SoC Implementation of Wildfire Infrared Monitoring and Early Detection System

Shalmon Hermesh[1], Shahnam Mirzaei[2], Ph.D., *Member, IEEE, California State University, Northridge CA 91330, USA, [1]shalmon.hermesh.29@my.csun.edu, [2]shahnam.mirzaei@csun.edu*

***Abstract*— This paper details the design and implementation of a wildfire infrared monitoring and early detection system (WIMEDS) on reconfigurable hardware. The proposed system aims to enhance the monitoring and detection of fires, with a particular focus on wildfires. Due to the complex nature of wildfire detection and classification, RGB imaging can be challenging. In contrast, infrared (IR) imaging simplifies this process, as fire captured by an IR camera displays a distinct pattern. This allows for more effective monitoring, detection, processing, and authentication of the images. The system is comprised of four primary components: image capture, software/hardware interface, image processing engine, and alarm module. The hardware module was implemented using the Vivado HLS Design Suite and targeted to the ZYNQ XC7Z020 SoC platform. The design efficiently utilizes FPGA resources, consuming less than 12% of Slice LUTs, 2% of memory LUTs, 5% of Slice registers, 8% of DSP blocks, and 4% of block RAMs. Implementation results demonstrate the system's inherent flame detection capabilities, effectively distinguishing hot flames from the surrounding environment.**

Keywords-component; Infrared; Image processing; SoC; HLS; Wildfire Early Detection; Wildfire Monitoring

I. INTRODUCTION

Early detection of wildfires has proven to be more critical than prevention, as most major wildfires are caused by human activities, lightning, or power lines, making them difficult to predict [1]. Early detection can help mitigate the significant risks and costs associated with these fires. For instance, California's largest and most destructive wildfire, the August Complex Fire in 2020, scorched over 1 million acres of land [2].

Common methods for wildfire detection include spotter planes, watchtowers, and satellites. In addition, ground-based systems utilizing both visual and non-visual technologies are also employed, such as visible light spectrum cameras, heat/flame sensors, and electronic lightning detectors [3]. Each detection method has its own set of advantages and limitations. For instance, human-operated methods like spotter planes are costly and limited by their line-of-sight coverage, which can be obstructed in low-visibility weather conditions. Similarly, visible light spectrum cameras are less effective in poor weather, while heat and flame sensors require a well-distributed network of devices and repeaters for optimal performance.

Infrared-based detection systems, in contrast, perform exceptionally well in low visibility conditions like smoke, fog, or rain. IR sensors can detect temperature variations as soon as a fire ignites, enabling quicker fire detection. When integrated with satellites or drone-based sensors, they can efficiently cover large areas. More importantly, IR sensors can identify heat signatures even when the fire is not visible to the naked eye. Long-wave infrared cameras, such as the Lepton sensor by FLIR Systems [4], are becoming increasingly common and are used in the system proposed in this paper.

In addition to IR sensors and cameras, effective early detection and response rely on an appropriate image processing system. This paper proposes a SoC-based real-time wildfire infrared monitoring and detection system that utilizes image processing techniques to track wildfires. The key contributions of this paper are the design and implementation of:

i. A wildfire detection circuit to process thermal images captured by an IR camera, enabling the identification of emerging hot spots.

ii. An image processing-based wildfire monitoring system to track the progression of wildfires.

The remainder of the paper is organized as follows: Section II provides the background, related work, available technologies, and design methodology. Section III outlines the system architecture, while Section IV details the implementation. Section V presents the analysis of the experimental results, and Section VI concludes the paper.

II. BACKGROUND

Alkhatib [3] surveys the most dominant fire detection technologies used for forest fire monitoring. Satellite-based wildfire detection faces challenges like high cost, limited coverage, poor resolution, and delayed detection due to weather conditions. Terrestrial visual systems, typically tower-mounted, rely on the visible light spectrum and assist firefighters during active firefighting. However, these systems are limited by visibility issues in adverse weather.

Several recent such as [5, 9] works have been developed addressing inference acceleration on FPGAs. However, they have mostly focused on using FPGA to accelerate CNNs. Garduno et al. [5] proposed a segmentation model for drone wildfire imagery using FPGAs and binarized neural networks (BNNs) for efficient edge computing in addition to CNNs. They optimized a U-Net model, reducing parameters by 90% and achieving 33.63 fps on a Xilinx Ultra96-v2, but did not provide details on FPGA resource utilization, IR camera type, or image resolution, making comparison with our work difficult. Their approach processes all images on the processor, with the FPGA acting as a hardware accelerator for BNNs. In contrast, our implementation integrates the entire system on FPGA, offering higher performance and reduced response time.

Jinghong et al. [6] designed a fire smoke detection system based on FPGA platform that includes image capture module, SDRAM data buffer module, image display module and smoke detection module. The authors have used the algorithms of frame difference, erosion and dilation to realize suspected smoke area. Methodologies in [6] and similar visible light spectrum-based schemes represent poor performance in undesired weather conditions and night vision and consequently they are not the best choice when it comes to wildfire monitoring and detection.

Thangavel et al. [7] discuss technologies using visible, infrared, multispectral, and hyperspectral sensors, focusing on hyperspectral imaging with neural networks. They note that preliminary information can be obtained from single bands, such as detecting smoke in the near-infrared (VNIR) bands and active wildfires in short-wave infrared (SWIR). Our infrared approach is the first SoC implementation with a complete image processing system in the FPGA for early wildfire detection and monitoring. By integrating high-performance image processing hardware in the Xilinx Zynq SoC, we significantly reduce response time. We use a long-wave infrared (LWIR) detection system with a FLIR Lepton thermal imager [4] to capture wildfire infrared images. We have used long-wave infrared (LWIR) based detection system using FLIR Systems 80x60 Lepton thermal imager [4] to capture wildfire infrared images.

III. System Architecture

The high-level system block diagram is shown in Figure 1. Image data collected via the serial interface from the Lepton camera is stored in the off-chip double data rate (DDR) memory. The data stream is then transferred to the Sobel operator using direct memory access (DMA). The resulting grayscale, edge-detected image is sent back to the DDR memory for further analysis at a later time. A dedicated algorithm running on the ARM processor handles the analysis of the reduced image data. To simplify the diagram, lower-level modules such as the timer and interconnect are omitted from Figure 1. Each module will be explained in more detail in the following sections.

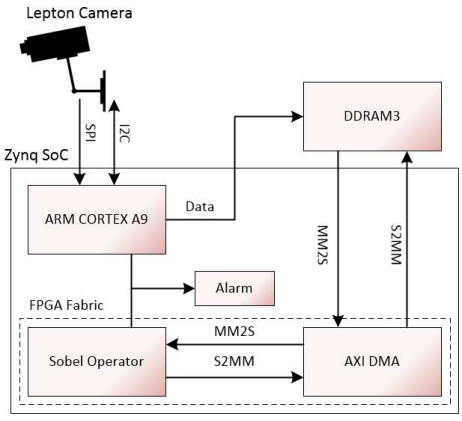

Figure 1. WIMEDS system block diagram

A. Camera Module and Infrared Image Frames

The IR cameras can be equipped with a radiometry mode, which influences the transfer function between the incident flux (scene temperature) and the pixel output. From an image quality perspective, for applications where the goal is to convert the camera's output signal into one that is proportional to the scene temperature, the radiometry mode is preferred. This is because the conversion remains constant across the entire operating temperature range of the camera, as illustrated in Figure 2 [4].

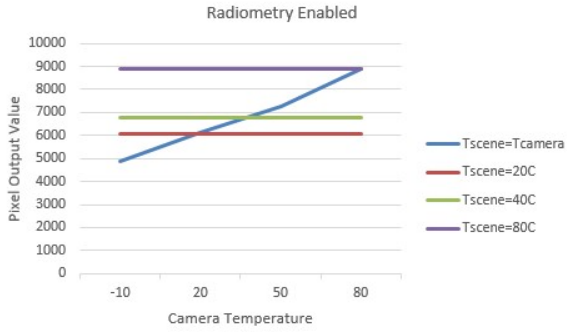

Figure 2. Hypothetical illustration of camera output pixel values vs. camera temperature in radiometry enabled mode

Color infrared transparency films consist of three sensitized layers that map infrared radiation to red, red to green, and green to blue. A typical infrared image captured by a FLIR device is shown in Figure 3 [4] on the top. The heat scale on the right margin of the image helps easily identify the fire and its surrounding areas, along with their corresponding temperatures. The processed infrared image, produced by our algorithm and shown on the right side of Figure 3 on the bottom, clearly highlights the enclosed fire area after the post-filtering stage. This would not be achievable using a visible light spectrum camera, as filtering the original fire images would not yield any meaningful patterns.

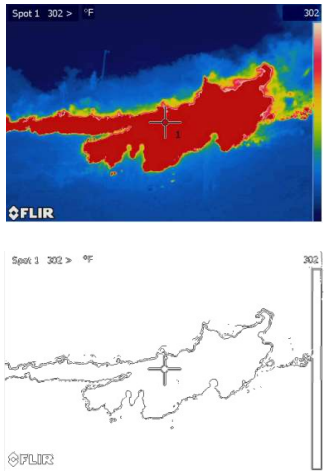

Figure 3. Top: Infrared spectrum fire image, Bottom: Filtered infrared spectrum fire image

The Lepton camera operates within the long-wave infrared (LWIR) spectrum and features a resolution of 80x60 pixels. With a size of less than one cubic centimeter, it is compatible with various handheld devices, including cameras and smartphones. The device offers different configurations and settings, including varying voltages, communication protocols, and lens aperture sizes. To establish communication with the Lepton device, a proper boot-up procedure is required. Specifically, the reset line must be deasserted after powering on and setting the serial clock. This is followed by a 950 ms waiting period. To determine if the camera is ready for operation, the camera status register, as shown in Figure 4, must be inspected. In particular, bit 0 (the camera busy bit) should read 0, and both the boot mode and boot status bits should be asserted.

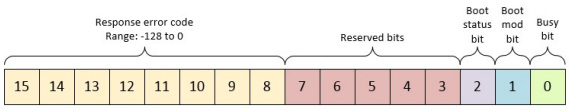

Figure 4. Lepton status register

The camera transmits 25.9 frames per second, with each frame consisting of 60 VoSPI packets, each 164 bytes. The first 4 bytes are for ID and CRC, while the rest are for the payload (Figure 5, top diagram). After boot-up, syncing with the camera is needed. Initially, the camera sends discarded and inter-frame packets marked with an ID of 0xF (Figure 5, middle diagram). A valid frame has an ID of 0x0000, followed by 59 packets with their ID and CRC (Figure 5, bottom diagram). The software verifies the validity of incoming frames during syncing.

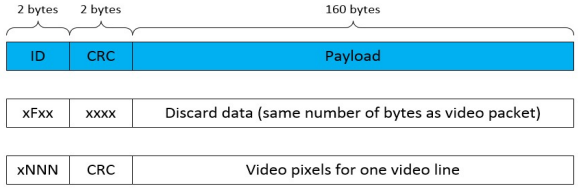

Figure 5. Top: Frame structure, Middle: Discarded frames, Bottom: Frame synching

B. Hardware/Software Interface

The camera supports two serial communication protocols: I2C for control and specific data transmission, and VoSPI for image transmission. The I2C protocol consists of a start condition, followed by an address packet that includes read/write information, slave acknowledgment bits, and one or more data packets, concluding with a stop condition. For transmitting the captured image or video stream, the VoSPI protocol is used, sending the captured data over the VoSPI bus for image transmission.

The Zynq SoC provides SPI and I2C interfaces as part of its serial communication capabilities. These interfaces must be initialized and configured through the software application. For I2C, the settings include a fast slew rate and a clock speed of 200 MHz. To ensure proper interaction between the Zynq and the Lepton camera over I2C, an I2C 3.3V to 2.8V level shifter was designed, though its details are omitted for simplicity. The Lepton's VoSPI interface supports clock rates ranging from 2 to 20 MHz. A serial clock speed of 150 MHz was selected and divided by 32 to achieve an SPI transmission frequency of approximately 4.6875 MHz. Equation (III-1) calculates the frame transmission time, while equation (III-2) computes the delay between frames. Since each packet consists of 1312 bits (164 bytes) and the Lepton sends 60 packets per frame, the time required to transmit ~25.9 frames per second at a clock rate of 4.6875 MHz is approximately 0.453 seconds.

t(frames/sec) = bits per packets/clock frequency (Eq. III-1)
*=164 *8/(4.6875*10e6)=0.0167 sec*

t(delay between frames) = (Eq. III-2)
*= (1 – frame rate * timer per frame)/frame rate*
*= (1 – 25.9 * 16.7e3)/25.9 = 21.1 ms*

IV. Implementation

Our implementation followed the design flow in [8], which included defining requirements, developing system architecture, partitioning software and hardware, and testing. Software development involved creating the board support package, application development, debugging, and establishing the software/hardware interface. Both hardware and software were developed using Vivado Design Suite and Vivado HLS tools. The design progressed through four stages: developing HLS IP for Sobel edge detection, firmware for the Zynq/Lepton interface, DMA transfer implementation, and system integration. Key components are explained in detail below.

A. Software Development

The software development process consists of five key parts: system initialization, Lepton camera syncing, data acquisition, Sobel execution, and analysis of the results. After successfully initializing the ARM processor, its peripherals, and fabric IPs, the camera's status is verified by sampling the boot status and mode bits. Once the syncing process is complete, the data acquisition and analysis proceed as shown in functional flowchart in Figure 6.

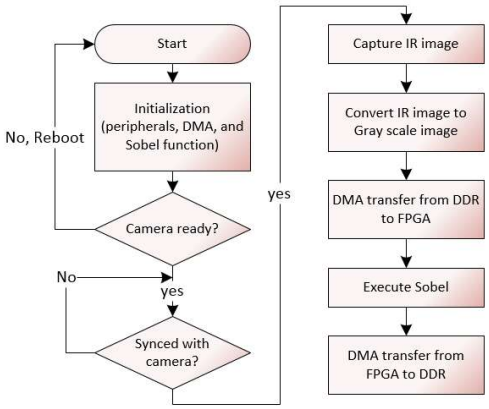

Figure 6. Software application functional flowchart

B. *Hardware Development*

HLS converts software algorithms in C/C++ into digital hardware described by HDL, accelerating RTL development with high-level programming languages. The Sobel operation, a computationally inexpensive differentiation operator, computes image gradients by convolving with horizontal and vertical filters. The HLS-implemented Sobel function uses two 2-D kernels, threshold operators, and an eight-bit data stream, performing a multiply-accumulate (MAC) operation with a sliding window. This is implemented using the HLS-built line buffer, as shown in Figure 7.

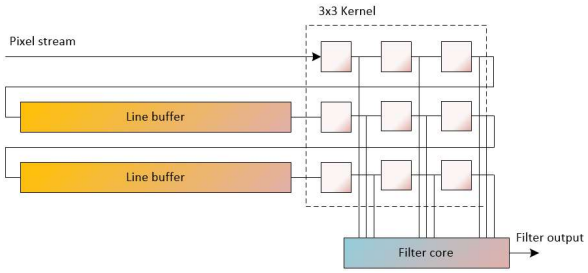

Figure 7. Illustration of the edge detection filter operation

The Sobel high/low threshold settings are used to differentiate between weak and strong image edges. If the accumulated value over the sliding window exceeds the high threshold, it is assigned a value of 255 (white), while values below the low threshold are assigned a value of 0 (black). The top-left image in Figure 8 shows the original image captured by the FLIR IR camera [4]. The top-right image displays the grayscale version of the original image. The bottom-left image applies high/low thresholds of 200/100, while the bottom-right image uses settings of 75/25. Lower threshold settings help eliminate noisy pixels. The thresholds chosen by the authors were determined through trial and error. A Direct Memory Access (DMA) engine was integrated to reduce the processor load. The DMA is responsible for transferring data stored in the DDR memory to the Sobel filter and then sending the resulting data stream back to the DDR memory, all without requiring any intervention from the processor.

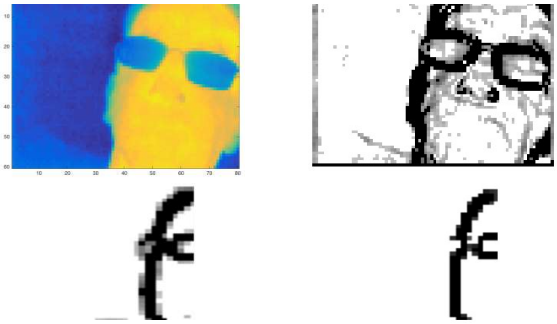

Figure 8. Top left: Original infrared selfie image, Top right: Grayscale processed selfie image, Bottom left: Processed selfie image with high/low thresholds 200/100, Bottom right: Processed selfie image with high/low thresholds 75/25

V. EXPERIMENTATL RESULTS

The WIMEDS was implemented on Zynq SoC. Table 1 shows the resource utilization of the reprogrammable area. As can be seen, the Sobel function consumes a small area of the FPGA fabric (20% of the design or 2% of the FPGA fabric). The difference between the numbers in two rows of the resource utilization table is consumed by the glue logic such as AXI interfaces and the DMA controller logic.

Table 1: WIMEDS FPGA resource utilization

Module	Resource Utilization				
	Memory LUTs	***DSP Blocks***	***BRAMs (36Kbits)***	***LUTs***	***FFs***
Sobel Function	30	18	2	1202	1083
Total	350	18	5	5723	5174

VI. CONCLUSION AND FUTURE WORK

While visible light-based fire detection systems are effective for wildfire monitoring, our infrared solution significantly improves detection and response. By integrating an infrared camera with high-performance image processing on an SoC platform, we enhance fire monitoring and reduce response times, saving lives, protecting nature, and lowering wildfire costs. Ongoing efforts aim to improve the camera's capabilities, including better image transmission via Bluetooth 5 and developing a flame pattern detection algorithm. We are also exploring a multicast wireless sensor network (ANT network) to enable communication with satellite fire detection systems.

REFERENCES

[1] https://www.fs.usda.gov

[2] https://www.fire.ca.gov/incidents/2020/8/16/august-complex-includes-doe-fire

[3] A. Alkhatib, “A review on Forest Fire Detection Techniques”, International Journal of Distributed Sensor Networks, Volume 2014, Article ID 597368

[4] FLIR Lepton Long Wave Infrared (LWIR) Camera http://www.flir.com

[5] Guarduño-Martinez, E., Ciprian-Sanchez, J., Valente, G., Rodriguez-Hernandez, G., Palacios-Rosas, A., Rossi-Tisson, L. and Ochoa-Ruiz, G., 2023. An FPGA smart camera implementation of segmentation models for drone wildfire imagery. arXiv preprint arXiv:2309.01318.

[6] L. Jinghong, Z. Xiaohui, W. Lu, “The design and implementation of fire smoke detection system based on FPGA”, Control and Decision Conference (CCDC), 2012, China

[7] Thangavel, K., Spiller, D., Sabatini, R., Amici, S., Sasidharan, S.T., Fayek, H. and Marzocca, P., 2023. Autonomous satellite wildfire detection using hyperspectral imagery and neural networks: A case study on australian wildfire. Remote Sensing, 15(3), p.720.

[8] Taylor, A., Binnun, D. and Srivastava, S., 2021. A Hands-On Guide to Designing Embedded Systems. Artech House.

[9] Briley, A.A. and Afghah, F., 2024, May. Hardware acceleration for real-time wildfire detection onboard drone networks. 2024-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS) (pp. 01-06). IEEE.