2023 3rd International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME) | 979-8-3503-2297-2/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICECCME57830.2023.10252728

Proc. of the International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME 2023)
19-21 July 2023, Tenerife, Canary Islands, Spain

New Architecture for Real-Time Image Computing Using Parallel Processing Based on DSP/FPGA

Aziz Oukaira
Electrical and Computer Engineering Department
Royal Military College of Canada
Kingston, ON K7K7L6, Canada
aziz.oukaira@rmc-cmr.ca

Ali Karime
Electrical and Computer Engineering Department
Royal Military College of Canada
Kingston, ON K7K7L6, Canada
ali.karime@rmc-cmr.ca

Vincent Roberge
Electrical and Computer Engineering Department
Royal Military College of Canada
Kingston, ON K7K7L6, Canada
Vincent.Roberge@rmc.ca

Mohammed Tarbouchi
Electrical and Computer Engineering Department
Royal Military College of Canada
Kingston, ON K7K7L6, Canada
tarbouchi-m@rmc.ca

***Abstract*—This paper introduces a parallel architecture designed for real-time image processing applications, utilizing a combination of digital signal processor (DSP) and field-programmable gate array (FPGA) components for optimal performance. The FPGA component features a first in, first out (FIFO) interconnection network and a specialized data communication protocol that enables effective interconnectivity between three DSPs (TMS320C6414). The performance metrics of the experimental prototype were impressive, as it exhibited both data, by leveraging its efficient image processing capabilities, the system achieves a harmonious balance in parallel processing.**

***Index Terms*—DSP, FPGA, image, optimal performance, parallel architecture, real-time**

I. Introduction

Image processing refers to the manipulation and analysis of digital images using various algorithms and techniques. It involves modifying or enhancing images to extract useful information or improve their visual quality. Image processing techniques are widely used in various fields, including computer vision, medical imaging, remote sensing, and multimedia applications. Performing image processing in real-time can present difficulties owing to the vast quantities of data and intricate algorithms required. The process of image processing can be categorized into three principal stages, namely the lower, intermediate, and upper levels. The involved processing algorithms can be complex and require significant processing capabilities, especially when it comes to parallel processing.

In recent decades, extensive research has focused on parallel system architecture, with notable contributions in the field of image processing. Crookes conducted a comprehensive study analyzing architectures specifically designed for this purpose [1]-[4]. In recent times, hardware implementations of image processing algorithms have emerged as the most effective solution for enhancing the performance of image processing systems [5]-[7]. The utilization of re-configurable devices and system-level hardware programming languages has further expedited the development of Digital Image Processing (DIP) in hardware. Field Programmable Gate Arrays (FPGAs) are commonly chosen as the implementation platforms for real-time image processing applications. FPGAs are programmable devices that allow end-users to directly configure the final logic structure. They consist of an array of flexible elements that can be interconnected or programmed in countless ways according to user specifications. FPGAs offer a valuable compromise between the flexibility of general-purpose processors and the digital signal processor (DSP). Moreover, their re-programmability and easy upgradability make them highly desirable for image processing tasks. The approach, described in [8]-[11], involves a DSP-based system where C6201s are physically interconnected through 32-bit bi-directional FIFOs.

This design allows high transfer rates for peer-to-peer DSP communication due to the large data bus and high clock rate of the bus, so this method is suitable for real-time image processing due to the speed, reliability of control, and ability to extract quantitative information from the images. However, this architecture has two significant drawbacks. First, a first-in-first-out (FIFO) buffer is required between each pair of DSPs, which results in a complex network topology and complicates the printed circuit board (PCB) layout. In addition, the connection topology is inflexible and fixed, which results in a loss of resources because the fixed connection cannot always be fully utilized. The tool we used for the implementation is Mirabilis Design.

The contributions of the present paper are: (1) propose a novel architecture for real-time image correction using DSP/FPGA-based parallel processing, (2) Validate the re-

sults by simulating the proposed architecture using high-performance DSPs interconnected by FPGA, and (3) measuring and comparing the performance between homogeneous and heterogeneous processors.

The paper is structured as follows: Section II outlines the design of parallel architecture introduced in this study. Section III provides a detailed description of the inter-DSP connection network implemented within FPGA. An analysis is conducted to evaluate the performance of the system in Section IV. The paper concludes in Section V by summarizing its main contributions.

II. The Proposed Design

Figure 1 illustrates the newly proposed architecture, which comprises three C6414 DSPs connected via an FPGA. In addition, individual digital signal processors (DSPs) in the system have access to dedicated local memories. The management of data input and output is handled by DSP 0, which functions as the master. The FPGA incorporates interfaces for both input and output, along with an extended interface that enables convenient expansion capabilities

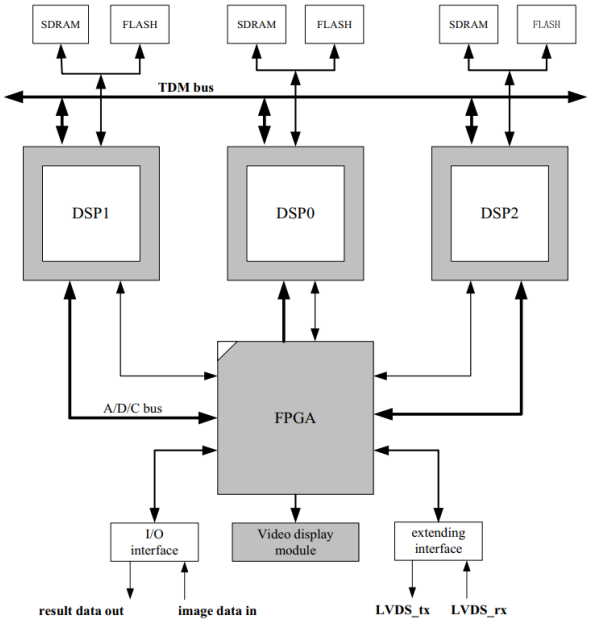

Fig. 1. The architecture that is being proposed for the parallel DSP system.

The DSP's McBSP operates in TDM mode to facilitate communication between the three DSPs, which enables synchronization operations in a multiprocessor system. In addition, the FPGA is connected to some of the general-purpose input-output (GPIO). These pins can be used for various purposes, such as controlling the FIFO and serving as flags. The operational clocks of three DSPs are synchronized since they are generated from the same clock buffer. As a result, it is essential to maintain uniform external memory interface A (EMIFA) configurations for all three DSPs to ensure that they read and write data to the FIFO at the same speed. This is critical for maintaining the accuracy of the data communication protocol between the DSPs.

III. The Connection Line Network

To support parallel image processing applications, it is crucial to facilitate the transfer of substantial amounts of image data between DSPs. This necessitates the establishment of the DSPs. In order to meet this requirement, the FPGA is equipped with inter-DSP, which consists of FIFO paths and data.

A. FIFO path connection networks

The FIFO interconnection network implemented in the FPGA is illustrated in Figure 2. The network is interconnected between two DSPs. Consequently, each DSP is connected to two FIFOs, enabling bidirectional data communication. To implement each FIFO with a depth of D_f, the FPGA's onboard memory was used. The FIFO interface is responsible for deriving the glue logic, including decoding FIFO addresses, adjusting write and read timing, and execution. This provides efficient communication between FIFOs and DSPs.

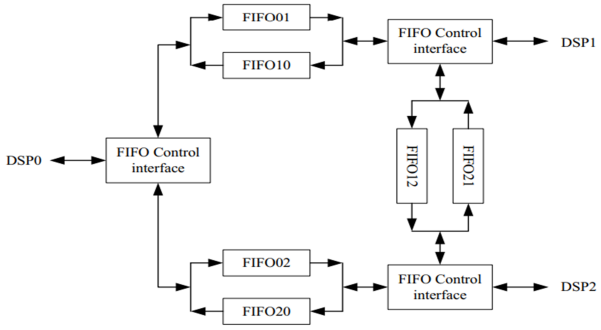

Fig. 2. Schematic of the FIFO cross-connect system implemented in an FPGA.

B. DSP-to-DSP interface protocol

In the next section, the DSP to DSP protocol is illustrated by DSP 0 transmitting images with DSP 1. In this scenario, DSP 0 writes the data into FIFO 01, and then DSP1 reads this data from FIFO 01, thereby enabling the transfer of data between the two DSPs.

1) Communication mode 1: involves transferring a predetermined size of data, such as an image with size R×C, between DSP 0 and DSP 1. At the start, DSP 0 clears FIFO 01 by driving a GPIO signal low and begins to write data into it. When FIFO 01 is half full, DSP 1 is interrupted by the status signal to start reading data. Meanwhile, DSP 0 continues to write data into FIFO 01 until it reaches R×C, regardless of the FIFO 01 status. EDMA mode is employed for both write and read operations and since the write speed matches the read speed, no issues arise even with FIFO 01 relatively small depth, preventing it from becoming full or empty.
2) Communication mode 2: is utilized when the data size to be transferred is not predetermined. In this scenario,

DSP 0 utilizes the TDM bus to notify DSP 1 of the data size, and the ensuing operations are carried out in a manner similar to mode 1.

C. Synchronization of the buses

As previously noted, each DSP's EMIFA is shared by four distinct FIFOs, which are regarded as shared resources by the other two DSPs. Therefore, synchronization mechanisms are required to achieve mutual exclusion of the resources. One approach to achieve bus synchronization is to use the TDM bus for passing messages.

D. FPGA and DSP synchronization interface signals

The interface between the module and the system is primarily a synchronized FIFO. For example, in the case of DSP 0 and FPGA, the FPGA is designated as a FIFO synchronous device and mapped into the CE0 slot of DSP 0. To access the FIFO, address signals are not required. However, the DSP 0 decodes the signals to generate input addresses for the four separate FIFOs. In our design, the DSP 0 handles interrupts to the receive FIFOs (FIFO 10 and FIFO 20), while ignoring the state of the transmit FIFO (FIFO 01 and FIFO 02). Although asynchronous interface signals are available as a contingency for timing problems, they were deemed unnecessary in practice. Figure 3 depicts the timing of the interface between the FPGA and DSP 0, including write and read operations.

The C6414 EMIFA is equipped with a programmable FWFT with a FIFO interface, in addition to the standard timing synchronous FIFO interface [12], [13]. In the present implementation, every FIFO operates in FWFT mode.

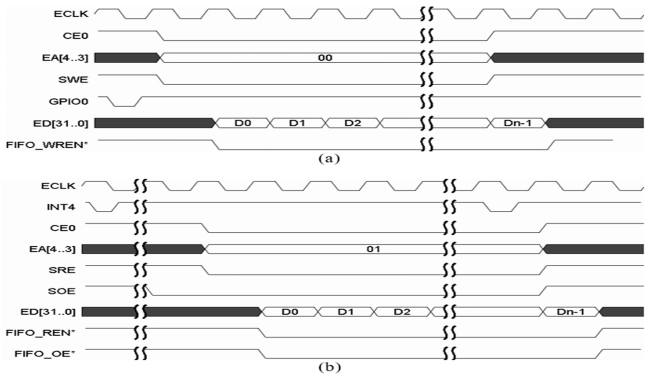

Fig. 3. Write and read times of the DSP on the FPGA interface: (a) write latency of one cycle; (b) read latency of one cycle.

IV. Evaluation of the System's Performance

A. DSP data performance communication

Figure 4 illustrates different image data sizes with various FIFO depths D_f, showcasing B_f from DSP 0 to DSP 1. The measurement process consisted in starting the timer on DSP 0 at the beginning of the send operation and stopping it once DSP 1 had received all the data and sent it back to DSP 0. The transfer latency τ corresponds to one-half of the total time required to send data from DSP 0 to DSP 1 as well as to receive it back. The transfer latency is made up of three sections of time: τ_1, the time taken for the sender to write $D_f/2$ data into FIFO 01; τ_2, the time taken for the receiver to respond to the interrupt and initiate the read EDMA; and τ_3, the time taken for the receiver to receive all the data. τ_1 represents the additional latency resulting from buffering the data using FIFO 01, while τ_2 and τ_3 are necessary overheads. It is evident that the smaller D_f is, the smaller τ_1 and τ are, and the larger D_f becomes. To ensure proper inter-DSP data communication, it is important to maintain the inequality $\tau_2 \prec \tau_1$. This means that FIFO 01 should not be full when the receiving DSP starts reading data from it. As the value of τ_2 is usually constant, the required condition for D_f can be expressed as follows:

Since $\tau_2 \prec \tau_1 = D_f W/2B$

Thus $D_f \succ 2B\tau_2/W$

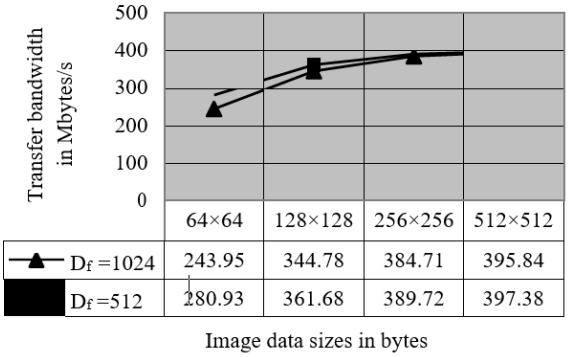

Fig. 4. The bandwidth of the data transfer, B_f, between DSP 0 and DSP 1 was measured for different image sizes.

According to Figure 4, with increasing data transmission size, the inter-DSP transmission bandwidth gradually approaches the theoretical bandwidth B (399 Mbytes/s). This is because the proportion of τ_1 to τ decreases as the size of the transmitted data is increasing. Figure 5 shows the transferred latencies for different image sizes with various FIFO depths.

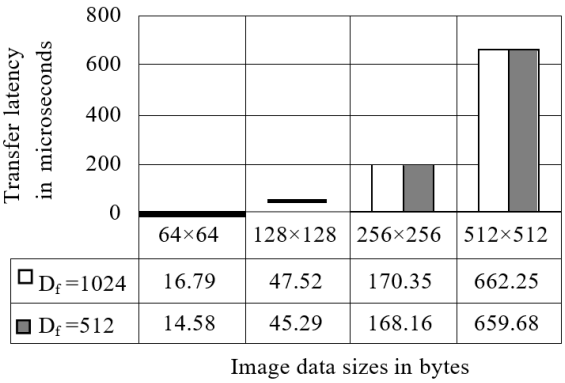

Fig. 5. The transfer latency, τ, between DSP 0 and DSP 1 was measured for different image sizes.

The additional latency resulting from data pulverization with FIFO 01, τ_1, is negligible when compared to τ. Consequently, achieving a consistently high-throughput, low-latency data transfer rate is essential for real-time parallel processing.

B. Computing performance of the system

The performance of the proposed parallel architecture on the prototype for 2D convolution is presented in Table I. Speed improves with the size of the image, as the computing speed is faster than the transmission speed. The performance of parallel processing is reasonable, exceeding 70%. Algorithms are currently developed in C, but further optimization is possible through manual implementation using machine language to reduce execution times. In conclusion, the results obtained show that the proposed parallel architecture is rational and very efficient compared to work [14], which makes it suitable for real-time image processing.

TABLE I.
2-D PERFORMANCE OF CONVOLUTION WITH A 3X3 MATRIX.

Image size in words	**1C6414 (ms)**	**3C6414 (ms)**	**Efficiency (%)**
128 × 128	1.438	0.619	77.30
256 × 256	5.835	2.388	81.30
512 × 512	23.508	9.405	83.30

V. CONCLUSIONS

The implementation of the parallel architecture is straightforward, using standardized and bondless DSP platforms, as well as a configurable FPGA architecture, resulting in an excellent performance. Our prototype systems have achieved data transfer bandwidths of up to 399 megabits/second, ultra-low latencies, and powerful computational capabilities using commercially available FPGAs. However, the interconnection scheme may not be suitable for constructing parallel systems with heterogeneous processors. This is mainly related to the fact that the particular database communication protocol requires that both processors are interfaced with the FPGA using the identical configuration, which may not be possible with heterogeneous processors on the system. This method can be applied to heterogeneous processors in the case of a task-frequency scheduler (TFS), which would improve the stability of task-frequency assignment for tasks with stable behavior.

REFERENCES

[1] C. Danny, "Architectures for high-performance image processing: The future", Elsevier Journal of Systems Architecture, vol. 45, n. 10, 2023, pp. 739–748.

[2] R. Aswathy and S. Harini, "Acceleration of Image Processing and Computer Vision Algorithms", IGI Global Handbook of Research on Computer Vision and Image Processing in the Deep Learning Era, 2023, pp. 1–18.

[3] C. Shuang and Z. Yang, "A review of convolutional neural network architectures and their optimizations", Springer Artificial Intelligence Review, vol. 56, n. 3, 2023, pp. 1905–1969.

[4] R. Daniel, G. Dennis and D. Jack, "Reinventing High-Performance Computing: Challenges and Opportunities", arXiv preprint arXiv:2203.02544, 2022.

[5] Z. Xiangming, "Application and Analysis of Computer Vision Algorithms in Graphics and Image Processing", International Journal of Informatics and Information Systems, vol. 6, n. 1, 2023, pp. 7–13.

[6] S. Xabier, N. Javier, B. Beñat, P. Iker and Bringas, and G. Pablo, "An Objective Metallographic Analysis Approach Based on Advanced Image Processing Techniques", Journal of Manufacturing and Materials Processing, vol. 7, n. 1, 2023, pp. 1–17.

[7] E. Ouafaa, O. Aziz, A. Mohamed, H. Ahmad, N. Morteza, S. Yvon, and L. Ahmed, "A real-time thermal monitoring system intended for embedded sensors interfaces", MDPI Sensors, vol. 20, n. 19, 2020, pp. 5657.

[8] C. Zhenkun, Z. Qihui, Y. Xiao, Z. Da, S. Xiang, Z. Chenguang, C. James and K. George, "DSP: Efficient GNN Training with Multiple GPUs", Proceedings of the 28th ACM SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming, 2007, pp. 392–404.

[9] O. Aziz, E. Ouafaa, T. Mohamed, T. Shamsodin, and L. Ahmed, "Simulation, Validation and FPGA Implementation of a Ring Oscillator Sensor for Thermal Management and Monitoring", Elsevier Procedia Computer Science, 2019, vol. 155, pp. 83–88.

[10] B. Antonio, S. Fanny, G. Raffaele, and F. Fabio, "An FPGA-Based Hardware Accelerator for the k-Nearest Neighbor Algorithm Implementation in Wearable Embedded Systems", Springer Applied Intelligence and Informatics: Second International Conference, 2023, pp. 44–56.

[11] L. Alexander, H. Philipp, P. Simon, M. Ralf, and R. Marc, "Most Resource Efficient Matrix-Vector Multiplication on FPGAs", IEEE Access, 2023.

[12] J. Xiaonan, J. Hongxu, X. Chaosheng, and W. Yuanpeng, "Software FIFO based interconnection between DSP and FPGA in video encoding system", IEEE International Congress on Image and Signal Processing, vol. 8, 2010, pp. 3699–3702.

[13] E. Mahyar, B. Endri, W. Janneck, and L. R James, "Auto-Partitioning Heterogeneous Task-Parallel Programs with StreamBlocks", Proceedings of the International Conference on Parallel Architectures and Compilation Techniques, 2022, pp. 398–411.

[14] K. Sourabh, and M. C. Andras, "Improving Effectiveness of Simulation-Based Inference in the Massively Parallel Regime", IEEE Transactions on Parallel and Distributed Systems, 2023.