Proceedings of the 6th World Congress on Intelligent Control
and Automation, June 21 - 23, 2006, Dalian, China

A DSP/FPGA -Based Parallel Architecture for Real-time Image Processing*

Luxin Yan
Institute of Pattern Recognition and Artificial Intelligence
Huazhong University of Science and Technology
Wuhan 430074 , China
yanluxin@gmail.com

Tianxu Zhang and Sheng Zhong
Key Laboratory for Image Processing and Intelligent Control
Huazhong University of Science and Technology
Wuhan 430074 , China
zs2971@gmail.com

Abstract-**A DSP/FPGA-based parallel architecture oriented to real-time image processing applications is presented. The architecture is structured with high performance DSPs interconnected by FPGA. Within FPGA a FIFO interconnection network and the specific data communication protocol are implemented, which interconnect 3 DSPs (TMS320C6414) effectively. The measured performances in the prototype with the proposed parallel architecture, including inter-DSP data communication performance and system computing capacity, show high data transfer bandwidth (up to 400 Mbytes/s) with low latency as well as high image processing performance, which achieve a good balance for parallel image processing.**

Index Terms-**Real-time image processing, parallel architecture, performance evaluation.**

I. Introduction

Real-time image processing often encounters the large volumes of data and the complex algorithms. Stated simply, image processing can be separated into three main levels. The first one, the lower level, deals with pixel-operation functions such as convolution and neighborhood filtering. At the intermediate level, like connected component labeling, motion estimation, segmentation and feature extraction or matching, necessitate a further global pixel analysis in order to extract the needed information. The upper level deals with interpretation, which usually requires artificial intelligence tools as well as previous knowledge of the environment. The lower level operations often are regular and fine-grain, but upper level, irregular and coarse-grain instead. Consequently the processing algorithms are complicated and require a high level of processing capacity, mainly in parallel processing.

A great deal of works on parallel system architecture has been carried out over the past decades. D. Crookes reviewed the architectures developed specifically for image processing [1]. Share bus-based systems are modular and scalable, but complicated and less predictable for real-time image processing. Distributed systems constructed by DSP (Texas C4X, ADI SHARC) with direct point-to-point communication offer the potential of real time image processing [1][2].

The new generation high performance digital signal processors, such as TMS320C6X (Texas Instruments) and TigerSHARC (Analog Devices, Inc.) families, have powerful processing capabilities and remarkable elevation of performance compared to C4X or SHARC. Some relatively simple parallel systems can be constructed with less numbers of such DSPs to achieve high performance, decreasing the difficulties of decomposing parallel algorithms as well as reducing the extra overheads of data communication between DSPs, consequently the system performance can be improved significantly.

Unlike C4X or SHARC, C64X has no high speed point-to-point communication ports or multiprocessor interface, so it is difficult to construct parallel system with C64X. McBSP is not competent for its low transfer bandwidth. Multiple C6xs connecting to a host controller via HPI (host-port interface) or expansion bus can structure multiprocessor system [3], but the peer-to-peer DSP communication should go through the host thus behaves with low efficiency, especially when the number of DSPs is large. Reference [4] describes a DSPs-based system in which C6201s are fully interconnected together via 32-bit bi-directional FIFOs physically, because of the wide data bus and high bus clock frequency the peer-to-peer DSP communication can achieve high transfer rates. However such architecture suffers, at least, from two main disadvantages. Firstly, there should be a FIFO between each DSP pair, which leads to the complex network topology and brings great difficulties in PCB routing. Secondly, the connection topology is fixed and lack of flexibility, furthermore, the fixed connection sometimes cannot be fully used, which caused resource waste.

In this article a parallel architecture of which C64Xs are interconnected via FPGA is presented. FIFO interconnection network and specific data communication protocol are implemented within FPGA, which ensure high data transfer rates for inter-DSP communication. Moreover, due to the FPGA, the system architecture can be conveniently reconfigured to adapt various parallelism approaches.

The article is organized as follows. Firstly, in Section II, the parallel architecture is introduced in general. The inter-DSP connection network implemented within FPGA is described in detail in Section III. An investigation into the system performance is presented in Section IV. Finally, conclusions are presented.

* This work was supported by the National Science Foundation of China under grant 60135020.

II. The Proposed Architecture

Figure 1 presents our new architecture, of which 3 DSPs (C6414) are interconnected via FPGA. Every DSP connects to the FPGA via EMIFA (external memory interface A) and exchanges data with the other two DSPs through the interconnection FIFO network implemented within FPGA. Also, there are local memories for each DSP. As the master, DSP0 takes charge of the data I/O of the system. Input and output interfaces are implemented in FPGA, in addition an extend interface is designed in FPGA for the convenience of expandability.

The McBSP of DSP works as TDM mode for passing messages among the 3 DSPs, which helps realize the synchronization operations in multiprocessor system. Furthermore, several general purposes I/O (GPIO) pins of DSP are connected to FPGA, which can be used to control FIFO, act as status flags for DSP consulting and pass simple message among DSPs for synchronization purposes.

All the work clocks of the 3 DSPs are derived from the same clock buffer, thus, as long as the configurations of the EMIFA of the three DSPs are uniform, the speed with which they read or write FIFO must be equal, what is more, it is the premise that the protocol of data communication between DSPs turn out to be right.

III. The Interconnection Network

In parallel image processing applications, mass images data should be transferred between DSPs, thus it is indispensable to construct high speed data communication channels between DSPs. Inter-DSP connection networks of FIFO paths and data communication protocol are implemented in FPGA to meet the demands.

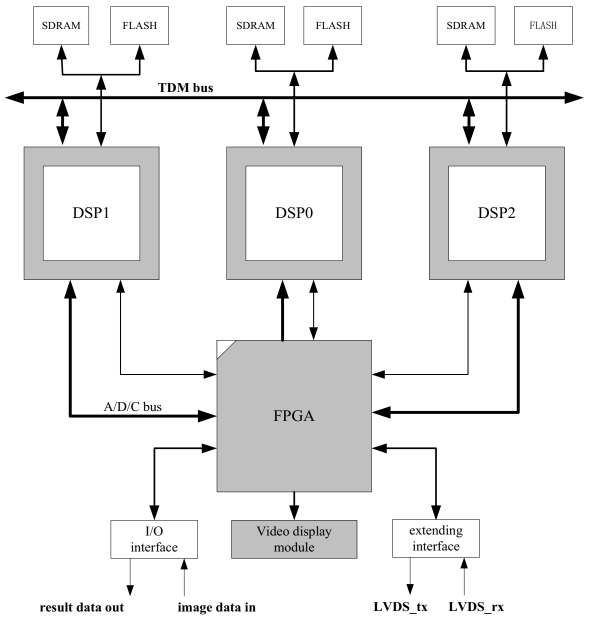


Fig. 1. The proposed architecture of DSP parallel system

A. Interconnection Networks of FIFO Paths

An interconnection network of FIFO paths implemented in FPGA is described as Fig. 2. There are three banks of FIFO, and each bank containing a transmitting and a receiving FIFO connects the DSPs by twos, so for each DSP, there are 2 banks of FIFO, which allows the bidirectional data communication with the other two DSPs respectively. Each FIFO is implemented using the embedded memory of Altera Stratix FPGA with D_f depth. The FIFO control interface carries out the glue logics, such as FIFOs' addresses decoding, adjusting write and read timing and communication handshake, helping the FIFOs interface to DSPs effectively.

B. Inter-DSP Data Communication Protocol

In the following, inter-DSP data communication protocol is introduced taking the case of DSP0 transferring an image to DSP1 as an example. DSP0 writes data into FIFO01 and DSP1 read out the data from FIFO01 to realize the data exchange between them.

1) Communication mode 1: The case that the size of data that need to be transferred is predetermined, for example, an image of size R×C. At the beginning, DSP0 drives a GPIO signal low to clear FIFO01 and then start writing data into it, when the FIFO01 comes half full the status signal will be asserted to interrupt the DSP1 to read data from it, at the same time, regardless of the status of FIFO01, DSP0 does not stall but continually write data into FIFO01 until the written data up to R×C. DSP1 responds the interrupt and then begin to read R×C data out from FIFO01 continually. Both write and read are performed by EDMA mode, because the write speed matches the read speed, though the FIFO01 depth is rather small, there are no exceptions of writing FIFO full or reading empty FIFO occur.

2) Communication mode 2: Communication mode 2 will be used while the size of data that need to be transferred is not fixed. First, DSP0 should inform DSP1 the data size via the TDM bus and following operation run in the same way as mode 1.

C. Bus synchronization

As the above said, the EMIFA of each DSP is shared by 4 different FIFOs, which could be treated as the share resource of the other 2 DSPs. It should be guaranteed that the receiver FIFO can take over the EMIFA bus before the two DSPs set up the data communication. Thus there should be some synchronization mechanisms to realize the resource mutual exclusion. Using TDM bus to pass messages can perform the bus synchronization.

D. Interface Signals and Timing of DSP and FPGA

In nature, the interface of DSP and FPGA is that of DSP and synchronous FIFO. Take the interface of DSP0 and FPGA as an instance, as a synchronous FIFO peripheral, FPGA are mapped into CE0 space of the DSP0. Though address signals are unnecessary to access FIFO, DSP0 decode address EA3 and EA4 to generate entrance addresses for the 4 different FIFOs.

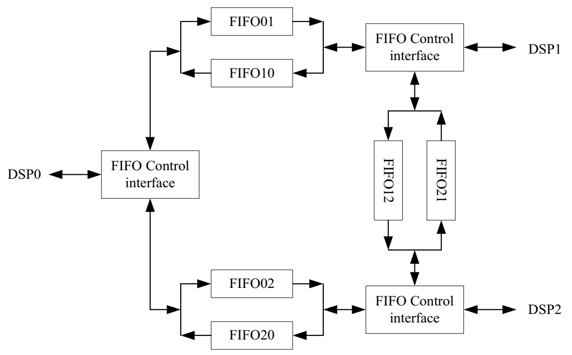
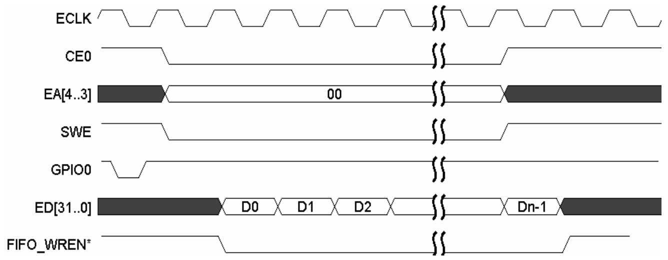
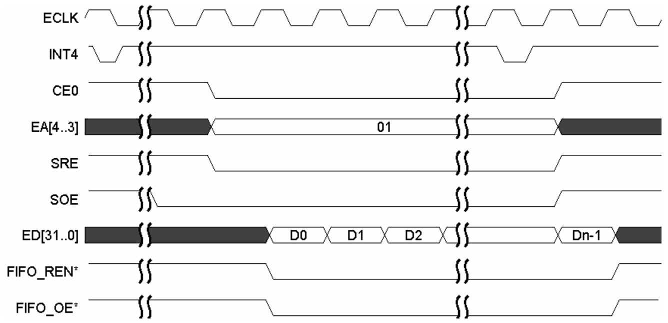


Fig. 2. Block diagram of FIFO interconnection network implemented within FPGA

DSP0 should respond the interrupts from the receiving FIFOs (FIFO10 and FIFO20) but not regard any status of the transmitting FIFOs (FIFO01 and FIFO02). Additional asynchronous interface signals are reserved for asynchronous interface timing when synchronous interface fails but these are found unnecessary in practice. The interface timing of FPGA and DSP0, including write and read operation timing, is illustrated in Fig. 3(a) and Fig. 3(b). The programmable synchronous mode of C6414 EMIFA supports both standard timing synchronous FIFO interface, and first word fall through (FWFT) FIFO interface [5]. In our design all the FIFOs are work in FWFT mode.



(a) DSP writes FPGA interface timing. 1 cycle write latency.



(b) DSP reads FPGA interface timing. 1 cycle read latency.

Fig. 3. The interface timing of DSP0 and FPGA. Signals signed * are internal signals within FPGA, derived from the glue logic of interface signals.

IV. System Performance Evaluation

A. Inter-DSP Data Communication Performance

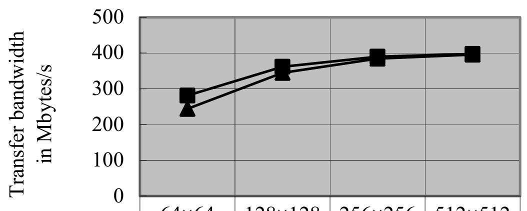
Inter-DSP data communication performance is the important factor that affects the system performance significantly, whereas data transfer bandwidth and transfer latency are the main parameters for measuring the data communication performance.

DSP accesses FIFO by synchronous mode with the theoretical bandwidth of B, which is equal to the bus clock frequency f times data width W .In the proposed system this means 100MHz×4 bytes that is equal to 400Mbytes/s. But B is not equal to B_f, the inter-DSP data transfer bandwidth, which is defined as the transferred data amounts in unit time. Figure 6 illustrates B_f from DSP0 to DSP1 for various image data sizes with different FIFO depth D_f .Measurements are made by starting the DSP0 timer at the beginning of sending operations and stopping it when all the data have been received at DSP1 and then returned to DSP0. The communication time is equal to the half of the total time in sending the data from DSP0 to DSP1 and receiving it back, in fact this is the transfer latency τ which consists of 3 sections of time: time τ_1 in the sender writing $D_f/2$ data into FIFO01, time τ_2 the receiver responding the interrupt and initiating the read EDMA, time τ_3 the receiver receiving all the data. τ_1 is the extra latency as a result of buffering the data using FIFO01, τ_2 and τ_3 are the indispensable overheads. Obviously, the smaller D_f is the smaller τ_1 and τ are, and the larger B_f becomes.

For the correct inter-DSP data communication, the inequality should keep true : $\tau_2 < \tau_1$, that means to say, FIFO01 is not still full when the receiver DSP begins to read data from it. Since τ_2 is often constant (90~100 core cycles for C6414), the conditional expression of D_f can be concluded as:

$$\text{Since } \tau_2 < \tau_1 = D_f W / 2B$$

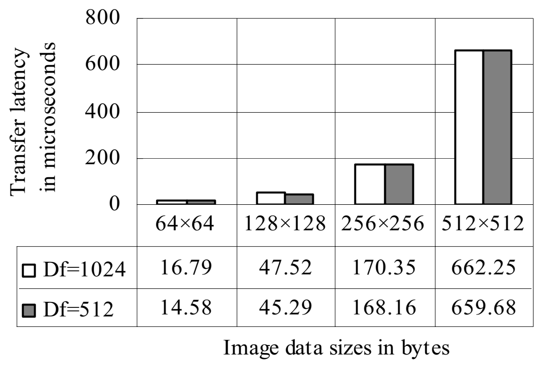
$$\text{Thus } D_f > 2B\tau_2 / W$$



	64×64	128×128	256×256	512×512
▲ Df=1024	243.95	344.78	384.71	395.84
■ Df=512	280.93	361.68	389.72	397.38

Image data sizes in bytes

Fig. 6. Measured data transfer bandwidth B_f between DSP0 and DSP1 for various image data sizes with different FIFO depth D_f .



	64×64	128×128	256×256	512×512
□ Df=1024	16.79	47.52	170.35	662.25
■ Df=512	14.58	45.29	168.16	659.68

Fig. 7. Measured data transfer latency τ between DSP0 and DSP1 for various image data sizes with different FIFO depth D_f .

From Fig. 6 it can been seen that inter-DSP transfer bandwidth gradually approaches up to B (400Mbytes/s) with the increase of transferred data size. That is because the proportion of τ_1 to τ decreases as the size of the transferred data increases. Figure 7 illustrates the transfer latency for various image sizes with different FIFO depth. τ_1, the extra latency as a result of buffering the data using FIFO01 is insignificant compared with τ . So both high data transfer rate and low latency are achieved, which are important for parallel real-time processing.

B. System Computing Performance

2D convolution computation is common in image processing, which computes each output pixel as a weighted average of several neighboring input pixels. The computation pattern is regular but with large numbers of operations. For example, convolution with a 3×3 mask on the 512×512 image need perform 2,340,900 MACs (multiply-accumulates) operations. Table I lists the 2D convolution performance on the prototype with the proposed parallel architecture. The speedup increases with image size, because the calculation time increases at a faster rate than the communication time for larger images. The efficiency of the parallel architecture is reasonable at over 75%. Now the algorithms are implemented in C language and optimized by the compiler, and can be implemented by hand in assembly language to further reduce execution times. The performance shows the parallel architecture is rational and of high efficiency, so it is suitable for real-time image processing.

TABLE I
2D CONVOLUTION PERFORMANCE WITH 3×3 MASK

Image size in words	Execution time (ms)		Speedup	Efficiency (%)
	1 C6414	3 C6414s		
128×128	1.438	0.619	2.32	77.3
256×256	5.835	2.388	2.44	81.3
512×512	23.508	9.405	2.50	83.3

V. CONCLUSIONS

Standard and glueless DSP interfaces, modifiable FPGA implementation make the parallel architecture easy to implement and has a very good performance. In our prototype systems, up to 400 Mbytes/s data transfer bandwidth with low latency as well as powerful computing capacity is achieved with standard, commercial FPGA. Furthermore, the interconnection scheme is suitable for homogeneous processors to construct parallel systems but for heterogeneous processors it may be not competent any longer, because the specific data communication protocol requires the both processors in communication to interface FPGA with the same configuration, maybe which is not fulfilled in heterogeneous processors system.

ACKNOWLEDGMENT

The authors would like to thank Sheng Zou, Jiao Wang, Wei Zheng and Lin Duan for their assistances to this work.

REFERENCES

[1] Danny Crookes, “Architectures for high performance image processing: The future,” *Journal of Systems Architecture*, vol. 45, pp.739-748, 1999.

[2] David M. H, Shirish P. K, C. Allan H, “Low cost scaleable parallel image processing system,” *Microprocessors and Microsystems*, vol. 25 pp.143-157, 2001.

[3] Zoran Nikolic, TMS320C6000 expansion bus: multiple DSPs to i80960Kx/Jx microprocessor interface, Texas Instruments, 2001.

[4] Hunt Engineering, Hepc8 full length PCI heron module carrier USER MANUAL, www.hunteng.co.ku, site visited 25/06/2005.

[5] TMS320C6000 Peripherals Reference Guide, Texas Instruments, 2001.