Neural Network & Distributed Processing

Vol. 4555

PROCEEDINGS OF SPIE
SPIE—The International Society for Optical Engineering

Neural Network and Distributed Processing

Xubang Shen
Jianguo Liu
Chairs/Editors

22–23 October 2001
Wuhan, China

Sponsored by
SPIE—The International Society for Optical Engineering
Huazhong University of Science and Technology (China)

Cosponsored by
Université de Bordeaux III (France)
University of Pennsylvania (USA)
Wuhan University (China)
Chinese Academy of Sciences
Beijing Institute of Environmental Features (China)
Tsinghua University (China)

Supported by
National Natural Science Foundation of China
Ministry of Education of China

Published by
SPIE—The International Society for Optical Engineering

Volume 4555

Please use the following format to cite material from this book:

Author(s), "Title of paper," in *Neural Network and Distributed Processing*, Xubang Shen, Jianguo Liu, Editors, Proceedings of SPIE Vol. 4555, page numbers (2001).

ISSN 0277-786X
ISBN 0-8194-4283-6

Published by
SPIE—The International Society for Optical Engineering
P.O. Box 10, Bellingham, Washington 98227-0010 USA
Telephone 1 360/676-3290 (Pacific Time) • Fax 1 360/647-1445
http://www.spie.org/

Printed in the United States of America.

Symposium Committees

Symposium Honorary Chairs

Benjamin W. Wah, University of Illinois/Urbana-Champaign (USA)
Bo Zhang, Tsinghua University (China)
Thomas S. Huang, University of Illinois/Urbana-Champaign (USA)

Symposium Chairs

Anil K. Jain, Michigan State University (USA)
Deren Li, Wuhan University (China)
Jun Shen, Université de Bordeaux III (France)

Program Committee

Chairs

J. K. Udupa, University of Pennsylvania (USA)
Tianxu Zhang, Huazhong University of Science and Technology (China)

Members

Bir Bhanu, University of California/Riverside (USA)
J. M. Carazo, Universidad Autónoma de Madrid (Spain) and Universidad de Malaga (Spain)
Yair Censor, University of Haifa (Israel)
Dingchang Chen, China National Space Administration
Jacky Desachy, Université Paul Sabatier (France)
Jufu Feng, Peking University (China)
Weikang Gu, Zhejiang University (China)
Pheng-Ann Heng, Chinese University of Hong Kong
Bruce Elliot Hirsch, Temple University (USA)
Thomas S. Huang, University of Illinois/Urbana-Champaign (USA)
Anil K. Jain, Michigan State University (USA)
Dehua Li, Huazhong University of Science and Technology (China)
Deren Li, Wuhan University (China)
Jian Liu, Huazhong University of Science and Technology (China)
Zhiyong Liu, National Natural Science Foundation of China
Siwei Lu, Memorial University of Newfoundland (Canada)
Henri Maître, Ecole Nationale Supérieure des Télécommunications (France)
Stephen J. Maybank, University of Reading (UK)
Yuichi Ohta, University of Tsukuba (Japan)
Heping Pan, CSSIP (Australia)
Sharatchandra Pankanti, IBM Thomas J. Watson Research Center (USA)
Jiaxiong Peng, Huazhong University of Science and Technology (China)

Maria Petrou, University of Surrey (UK)
Jun Shen, Université de Bordeaux III (France)
Xubang Shen, Xian Institute of Microelectronics Technology (China)
Zhenkang Shen, National University of Defense Technology (China)
Pengfei Shi, Shanghai Jiaotong University (China)
Zhongzhi Shi, Institute of Computing Technology (China)
Tieniu Tan, National Key Laboratory of Pattern Recognition (China)
Zheng Tan, Xi'an Jiaotong University (China)
Jun Tian, Rice University (USA)
Qingxi Tong, Institute of Remote Sensing Applications (China)
Jayaram K. Udupa, University of Pennsylvania (USA)
Benjamin W. Wah, University of Illinois/Urbana-Champaign (USA)
Friedrich M. Wahl, Technische Universität Braunschweig (Germany)
Jinxue Wang, SPIE—The International Society for Optical Engineering (USA)
Patrick S. Wang, Northeastern University (USA)
Runsheng Wang, National University of Defense Technology (China)
Yuanmei Wang, Zhejiang University (China)
Chengke Wu, Xidian University (China)
Guangyou Xu, Tsinghua University (China)
Jie Yang, Shanghai Jiaotong University (China)
Jingan Yang, Hefie University of Technology (China)
Jingyu Yang, Nanjing University of Science and Technology (China)
Xingliang Yin, The Second Academy (China)
Baozong Yuan, Northern Jiaotong University (China)
Yehoshua Y. Zeevi, Technion—Israel Institute of Technology
Bo Zhang, Tsinghua University (China)
Tianxu Zhang, Huazhong University of Science and Technology (China)
Guangxi Zhu, Huazhong University of Science and Technology (China)
Yaoting Zhu, Huazhong University of Science and Technology (China)
Zhenfu Zhu, The Second Academy (China)

Organizing Committee

Chairs

Mingyue Ding, Huazhong University of Science and Technology (China)
Qingxi Tong, Institute of Remote Sensing Applications (China)
Yaoting Zhu, Huazhong University of Science and Technology (China)

Conference Committee

Conference Chairs

Xubang Shen, Xi'an Institute of Microelectronics Technology (China)
Jianguo Liu, Huazhong University of Science and Technology (China)

Program Committee

Chaoyang Chen, Huazhong University of Science and Technology (China)
Jyh Cheng Chen, National Yang-Ming University (Taiwan)
Yongji Wang, Huazhong University of Science and Technology (China)
Chengke Wu, Xidian University (China)

Contents

POSTER SESSION

Design of a parallel RISC image processor based on PCI bus

Jiang Xianyang[a,*], Shen Xubang[a], Zhang Tianxu[a]
[a]Image Processing & Intelligent Control Key Laboratory of Ministry of Education of China,
Institute for Pattern Recognition & Artificial Intelligence,
Huazhong University of Science & Technology, Wuhan, 430074,P.R.China

ABSTRACT

Low-level image processing operations usually involve simple and repetitive operations over the entire input images, thus image processor may communicate with the memory system or each other frequently, hence the image processor would provide high throughput rate. In this article we present an architectural design and analysis of a parallel RISC image processor. The processor was based on PCI bus to speed up a range of image processing operations. The other characteristic of the processor is that a new three-port hostbridge is integrated into the processor. The implementation of commonly used image processing algorithms and their performance evaluation are also discussed.

Keywords: Image processing, image processor, ARM core, hostbridge, parallel processing

1. INTRODUCTION

Low-level image processing operations usually involve simple and repetitive operations over the entire input images[1][2]. In real time applications, most of the input images are required to be processed at a rate of 25 frames per second. Hence, with each frame consisting of 256K pixels (512×512 resolution), tens of magabytes of image data or more must be processed at every second. To provide such a high throughput rate, parallel implementation has to be investigated. Moreover, in an image processor, source data are usually retrieved from the memory or supplied by external peripherals or other macrocells, via the bus, therefore, the bus (which even be multiplexed, for space reasons) is a bottleneck for the feed of data. As we all know now, processor architecture and software algorithm are both suitable ways to implement parallelism, and theoretically, the highest throughput rate will be achieved when a perfect match is met between the processor architecture and the algorithms[3]. Keeping this in mind, the design of our PCI bus based processor was undertaken. The processor was designed for many image processing applications.

We choose RISC architecture for it is simpler than CISC architecture and is easy to design. To speed up the design, we also reuse an ARM core and pay much attention to peripheral interface design.

2. ARCHITECTURE OF THE RISC IMAGE PROCESSOR

Parallelism is implemented by PCI bus. After originally developed in 1992, PCI bus successfully met most demands of the industry and is now the most widely accepted and implemented expansion standard in the world. By PCI bus, two of our designed processors can communicate each other with high throughput based on PCI bus, hence the high concurrency could be achieved.

Fig. 1 shows the block diagram of the image processor. There are seven main functional units that are connecting each other in the processor. The hardware units are listed bellow:

(1).ARM7TDMI core: This module is a reused intellectual property (IP) core. The ARM core takes care of OS and configuration and provides a workbench for programmer. There are various types of licensed ARM core, for our convenience, we chose the type of source code for we can modify the source code as we need, though such type of the core

* Correspondence should be sent to xianyangjiang@etang.com

Neural Network and Distributed Processing, Xubang Shen, Jianguo Liu, Editors,
Proceedings of SPIE Vol. 4555 (2001) © 2001 SPIE · 0277-786X/01/$15.00

is much more expensive.

(2).AHB-PCI hostbridge: This module is special for our processor. It is in charge of the protocol communication between advanced high-performance bus (AHB) of ARM core and PCI bus, and it's the first time to bridge the two buses. The hostbridge also integrates one DMA to deal with mass data for improving performance.

(3).Channel Access Controller: It's used to control access to different memory systems.

(4).JTAG and TAP Controller: It's used to test the processor.

(5).System Control Unit: it consists of timer, reset, interrupt control unit and watchdog unit.

(6).The Independent DMA Module: This module is in charge of the data exchange between different memory areas. In the memory system, there are several blocks that can only be accessed by different master respectively, thus data exchange between them should be considered.

(7).Basic Communication Module: The module includes serial channel and parallel I/O controller.

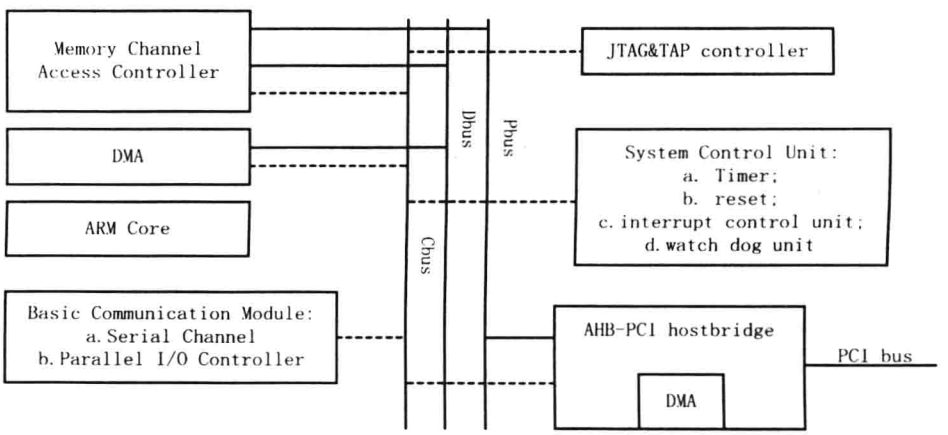

Fig. 1 Top level view of the RISC image processor

Though ARM has developed a bus architecture called AMBA, the use of which will improve expandability, greatly ease system design, and help testing—particularly in applications that require multiple bus masters, we change to not to use the slow advanced peripheral bus (APB) and directly bridge AHB to PCI bus by AHB-PCI hostbridge for improving efficiency of data exchange. Because low speed bus APB is not implemented in the processor, the design is simpler. There are three system buses in the processor, i.e., Pbus, Cbus, Dbus, Which in normal run mode can only be accessed by AHB-PCI hostbridge, ARM core and the independent DMA respectively. Setting three system buses is to solve bus usage competence between independent DMA and PCI bus. The three system buses architecture makes it possible that internal data exchange in the system can do at the same time to data exchange between the chip and peripheral devices. Furthermore, the three system buses architecture makes the hostbridge get a three-port structure which is different from other hostbridge.

3. SOME DETAILS OF THE DESIGN

3.1 INTEGRATION OF ARM7

ARM7 is a small, 32-bit RISC core with very low power consumption and good code density. Using the Dhrystone 1.1 Benchmark, the ARM7 code size is only 23% larger than that of i386. Overall, ARM7 achieves an average CPI (Clock Cycles Per Instruction) of around 1.8. When with 0.35 CMOS process, typical power consumption is 1.5mW/MHz. Our image processor takes these advantages of the ARM core, and the processor gets the speed more than enough for the cost-sensitive applications.

The integration of the ARM core includes building and configuring the runtime (and real time) deployed platform, configuring each module, and assigning interrupts and memory addresses for its subsystems. Because the source code of the core is independent with implementation technology, we can edit the source code as we need before the mapping.

3.2 AHB-PCI HOSTBRIDGE

The block diagram of the AHB-PCI hostbridge is shown in figure 2.

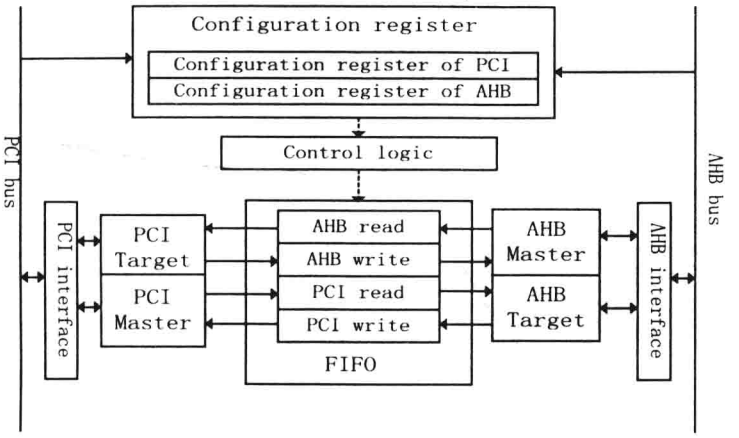

Fig. 2 Block diagram of AHB-PCI hostbridge

The hostbridge takes care of protocol communication between AHB and PCI bus, hence based on the PCI bus, two of our designed processors can communicate each other parallelly with high throughput. In order to make the image processor to work either in master mode or in slave mode, we design four state machines in the hostbridge and the hostbridge can be configured in either bus.

Table 1 Subset implementation of PCI command

C/BE[3::0]#	Command Type	PCI MASTER	PCI TARGET
0000	Interrupt Acknowledge	×	×
0001	Special Cycle	×	×
0010	I/O Read	√	√
0011	I/O Write	√	√
0100	Reserved		
0101	Reserved		
0110	Memory Read	√	√
0111	Memory Write	√	√
1000	Reserved		
1001	Reserved		
1010	Configuration Read	√	√
1011	Configuration Write	√	√
1100	Memory Read Multiple	√	√
1101	Dual Address Cycle	×	×
1110	Memory Read Line	√	√
1111	Memory Write and Invalidate	×	×

To speed up the design, we only implemented a subset of PCI command given by PCISIG, the subset is show in table 1, where × denotes not support of the command, √ denotes support of the command. Also to simplify the protocol communication, we modify some of the transaction ordering rules given by PCI specification. Our rules are shown in table

2. Where PMW denotes Posted Memory Write, DR denotes Delayed Request, DC denotes Delayed Completion. For our specific system, deadlock shouldn't occur, that is due to the following reasons: (1) Internal memory always responses in certain time, DR forward to internal CPU can always complete in certain time; (2) When CPU is the slave device of PCI bus, PW forward to CPU can pass Read forward from PCI master and complete earlier, thus the PW forward from CPU can always arrive at the final destination; (3) The Read forward from PCI master to CPU can always return result.

3.3 PIPELINE OF THE PROCESSOR

Like traditional RISC processors, the process in image processor is decomposed into a three-stage instruction pipeline (Fig. 3): instruction fetching (IF), instruction decoding (ID), and instruction execution (EX). The overlapping execution among the three stages is a characteristic with ARM core, with this characteristic, the performance of the processor could be improved.

Table 2 Implementation of transaction ordering

Row pass Col.?	PMW	DR	DC
PMW	No	No	YES
DR	No	No	YES
DC	No	YES	YES

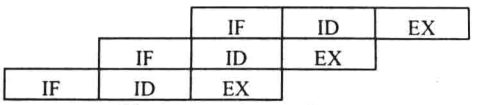

Fig. 3 Instruction pipeline

4. COMPARISON WITH OTHER IMAGE PROCESSORS

In order to evaluate the architecture and performance of the image processors, we have conducted extensive experiments and have compared the performance of our image processor with a number of processors, namely Intel 80i860, TMS320C40 DSP and T800 Transputer. These processors are the most commonly used processors for real time applications. Figs. 4 and 5 show the comparative performance of the processors in implementing the same dilation and convolution algorithms over the same 512×512 test image given in paper[4], respectively. For convenience, we also only discuss the two algorithms in this paper. As we can see, the proposed architecture is more powerful than the commonly used processors.

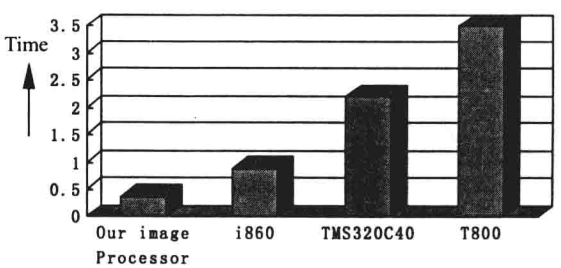

Fig. 4 Dilation algorithm execution times of processors

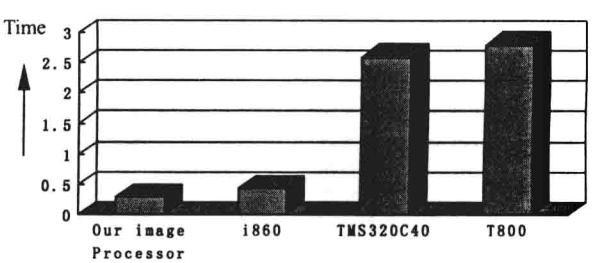

Fig. 5 Convolution algorithm execution times of processors

5. CONCLUSIONS

In this article, we have presented the architectural design and the results of an image processor based on PCI bus. An important characteristic in the design of the image processor is its integration of one AHB-PCI hostbridge. As a result of this hostbridge, concurrency between processors is significantly improved. Our experiments show that it is more powerful to implement various image-processing algorithms Therefore, the proposed processor could be a good candidate for real-time image processing applications.

ACKNOWLEDGEMENTS

The author would like to thank Dr. Liang Songhai for his helpful suggestions and Dr. Liu huayu for their cordial encouragements.

REFERENCES

1. C.J.Turner, V.C.Bhavsar, "Parallel implementations of convolution and moments algorithms on a multi-transputer system," *Microprocessor and Microprogramming*, pp.283–290,1995.
2. P.A. Navaux, A.F. Cesar, "Performance evaluation in image processing with GAPP array processor," *Microprocessor and Microprogramming*, pp.71–82, 1995.
3. V. Cantoni, C. Guerra, S. Levialdi, *Towards an Evaluation Of An Image Processing System, in: Computer Structures for Image Processing*, pp.43-56, Academic Press, New York, 1983.
4. M.Y.Siyal, M.Fathy, "A programmable image processor for real-time image processing applications," *Microprocessors and Microsystems*, **23**, pp.35–41, 1999.

An Efficient Method for Hardware Based DCT/IDCT Implementation*

Sun Xiantao, Wu Chengke
ISN National Key Laboratory,
Xidian University, Xian, 710071, China,

ABSTRACT

Discrete Cosine Transformation (DCT) and Inverse Discrete Cosine Transformation (IDCT) are important parts of many image and video compression system. Unfortunately these operations are extremely computation-intensive in a coding system, it consumes a large amount of resources for computation, especially in a real-time video coding system. In this article an efficient method for hardware based DCT/IDCT implementation is proposed. We combine the vector processing with parallel processing using Distributed Arithmetic. At the same time the processing elements are pipelined to increase the processing speed and reduce the computation latency, which can also reduce the resource requirement and thus enhance the efficiency.

Keywords: DCT, IDCT, Distributed Arithmetic, row-column decomposition, parallel processing

1 INTRODUCTION

The 2-D DCT and IDCT are core parts of a variety of video and image coding standards including H.261[1], H.263[2], MPEG1[3] and MPEG2[4] etc. Their performance and efficiency have a great influence on the performance of the whole coding system. Recently increasing interests have been focused on DCT and IDCT transform due to the explosive growth of the real-time video and multimedia services in communication area.

Various algorithms and implementation techniques have been proposed to accommodate different applications. The conventional technique to implement the DCT and IDCT is to decompose the 2-D transform into two 1-D transform stages. One is a row-wise transform, the other is a column-wise transform [5]. Another technique is to use the polynomial transformation. This algorithm is hard to implement in hardware due to its irregular structure and data flow path. So the first method is more suitable for hardware-based implementation. When it is combined with Distributed Arithmetic, it can simplify the design at the same time increase the efficiency. The method proposed in this paper is based on this technique. And we modify the data flow path of the implementation and reduce the resource usage and computation latency.

This paper is organized as follows, in section 2, the row-column decomposition algorithm and the Distributed Arithmetic for its implementation is introduced. In section 3 our FPGA based architecture is proposed and its performance is analyzed. In section 4, we conclude this article.

* This work is sponsored by Defence Advanced Research Project and Defence Advanced Research Foundation of China

Neural Network and Distributed Processing, Xubang Shen, Jianguo Liu, Editors,
Proceedings of SPIE Vol. 4555 (2001) © 2001 SPIE · 0277-786X/01/$15.00

2 DISTRIBUTED ARITHMETIC FOR DCT AND IDCT

Distributed Arithmetic [5][6] is a technique that allows the hardware implementation of a sum of products without using multiplier. By storing first a finite number of immediate results in a lookup table, a sum of products can be obtained through repeated addition and shifting operations. This can greatly reduce the complexity of the circuits and increase their processing speed. This technique is ideal for the multiplied-by constant algorithm such as FIR filters and DCT/IDCT implementation.

The DCT/IDCT operation can be represented as follows in forms of matrix multiplication,

$$F = CfC^T \tag{1}$$

Where C is the cosine coefficient matrix and f is the input.

Using row-column decomposition, the multiplication can be decomposed into 2-stage multiplication:

$$Y = Cf^T \tag{2}$$

$$F = CfC^T = C(Cf^T)^T = CY^T \tag{3}$$

In this way a 2-D DCT/IDCT can be computed serially using a 1-D DCT transformation module.

For N=8, a 1-D DCT/IDCT can be written as

$$Y_{k,l} = \sum_{m=1}^{N} c_{m,l} f_{k,m} \qquad l,k = 1,2,3.....,N \tag{4}$$

So the 1-D DCT/IDCT requires eight eight-point inner products. However due to the inherent regularity of the DCT/IDCT coefficients matrix, the equation above can further be decomposed into two four-point inner products [7]. Consider the computation of the inner products below:

$$y = \sum_{m=1}^{N} a_m x_m \tag{5}$$

Where x_m are input and a_m are constant coefficients. Assuming each x_m is a two's complement binary number with B-bits precision and $|x_m|<1$, then x_m can be expressed as

$$x_m = -x_m^{(0)} + \sum_{j=1}^{B-1} x_m^{(j)} 2^{-j} \tag{6}$$

In equation (6), x_m^j is the j th bit of x_m.

Substituting (6) into (5), we get

$$y = \sum_{j=1}^{B-1} \left[\sum_{m=1}^{N} a_m x_m^{(j)} \right] 2^{-j} - \sum_{m=1}^{N} a_m x_m^{(0)} \tag{7}$$

Let

$$F_N(a,x^{(j)}) = \sum_{m=1}^{N} a_m x_m^{(j)} \tag{8}$$

Substituting into equation (7), yields

$$y = \sum_{j=1}^{B-1} F_N(a,x^{(j)})2^{-j} - F_N(a,x^{(0)}). \tag{9}$$

Since $x_m^{(j)}$ may only be either 0 or 1, $F_N(a,x^{(j)})$ can take only 2^N possible values. These values can be computed in advance and stored in a lookup table. Then we can obtain y by the lookup table, a shifter and an accumulator. The diagram of the lookup-table based accumulator is shown in figure 1.

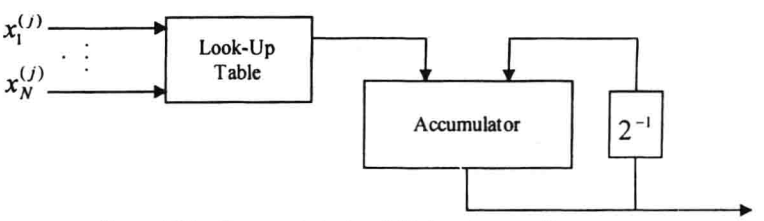

Figure 1 The diagram of a lookup table based accumullator

3. DCT and IDCT architecture for Hardware based implementation

3.1 Architecture Overview

The DCT/IDCT architecture is show in figure2. It contains 1) two 1-D DCT/IDCT transform modules, which do the row-wise and column-wise transform respectively. 2) a dual port RAM which stores the immediate results of the row-wise computation. 3) an address generation and access control module which controls the two 1-D DCT/IDCT transform module to access the dual port RAM and at the same time transpose the immediate results matrix and feed it to the column-wise transform module. 4) a serial-parallel transform module and a parallel-serial transform module which reduce the I/O port requirement of the device.

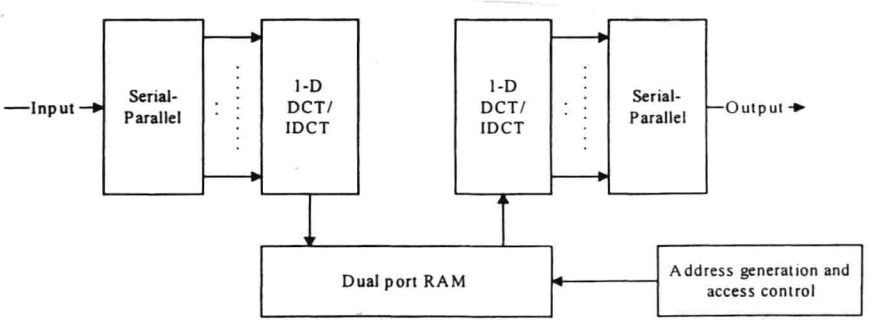

Figure 2 Block Diagram of the DCT/IDCT Architecture