Multidimensional Systems and Signal Processing, 8, 335–349 (1997)

Fast Implementation of 3-D Digital Filters Via Systolic Array Processors

B. G. MERTZIOS mertzios@demokritos.cc.duth.gr
Department of Electrical and Computer Engineering, Democritus University of Thrace, 67 100 Xanthi, Greece

A. N. VENETSANOPOULOS
Department of Electrical and Computer Engineering, University of Toronto, Toronto M5S 1A4, Canada

Abstract. In this paper a fast implementation architecture of three-dimensional (3-D) FIR or IIR digital filters via systolic VLSI array processors is described. The modular structure presented is comprised of similar processing elements in a linear cascade configuration with local interconnections. High speed throughput rates are attained due to high concurrency, which is achieved by exploiting both pipelining and parallelism. The considered 3-D FIR and IIR filters may be used for the processing of reconstructed 3-D images and in medical imaging applications.

Key Words: 3-D digital filters, systolic arrays, piplining, parallelism

I. Introduction

During the last decade we have witnessed a growing interest in the design and implementation of three-dimensional (3-D) and M-D digital filters [1]–[10], which find numerous applications in medical imaging [11]–[14], computer vision [15], 3-D TV video signals [16], image restoration and enhancement for geophysical and seismic data [17], and processing of time-varying images [18]. In particular, digital image processing is necessary for the processing of medical images in order to provide higher quality images for interpretation and diagnosis. In medical imaging applications there is usually a clinical need to examine sections of the human body along directions, where direct image acquisition cannot be attained [12]. This latter problem of representation of 3-D images is solved by combining a number of two-dimensional (2-D) sections and then filtering the 3-D reconstructed images. Reconstruction and processing of 3-D images are used in imaging magnetic resonance (MRI), medical imaging and in the reconstruction of the carotid vessel by echographic sections [14].

The need for fast processing of huge amount of data has led to the use of special purpose hardware architectures since the computer-based digital signal processing systems, which are designed with a general purpose structure and data processing philosophy, have certain features that prevent high throughput. The most prominent of the special purpose distributed processing structures are the *Array Processors* (APs). APs are ideal for the fast real-time implementation of complex digital signal processing algorithms. The aim of the APs design is defined as the choice of the best pipelined and parallel processing techniques and device technology, in order to meet satisfactory performance with low-cost. In particular, the VLSI APs are special purpose, locally interconnected computing networks that maximize concurrency by combining pipelining and parallelism. They are fully implemented by VLSI chips and are characterized by massive concurrency and regular data flow [19], [20]. The two most popular special purpose VLSI APs are the systolic and wavefront arrays, which exploit both pipelining and parallelism by using the concept of computational wavefront [20]–[22].

A systolic array is a network of elementary *Processing Elements* (PEs) that rhythmically compute and pass data through the system. A wavefront array is a systolic array plus data flow computing. Both kinds of arrays are algorithm oriented and they feature the desired properties of regularity, modularity, local communication, data and computational pipelining and highly synchronized multiprocessing. However, the systolic arrays are globally synchronized since the data movements are controlled by global timing-reference "beats", while the wavefront arrays are locally synchronized since the data movements are controlled by correct sequencing using handshaking. In addition to the classical implementation criteria of low-sensitivity with respect to finite word length effects, absence of overflow oscillations and limit cycles, the figures of merit in the VLSI implementation of digital signal processors are: (i). Concurrency, which is achieved by pipelining (data and computational pipelining), parallelism and multiprocessing, (ii). Repetitive, regular and modular structure, (iii). Local communication, which is the only permitted, (iv). Local synchronization and (v). Workload and flow distribution.

Since the communication in VLSI is very expensive and remains restrictive, only short local communication paths are used, while no shared buses are needed. VLSI APs have been used for the fast implementation of many matrix based algorithms (matrix decompositions, triangularization, matrix-vector multiplications) and signal processing algorithms (convolution and FFT techniques, estimation) [19], [20]. Moreover, recently fast implementation structures were presented for one-dimensional (1-D) [22]–[28] and 2-D digital filters [29]–[31]. The recursive algorithms, or generally the realizations which need feedback, are usually considered inappropriate for high speed implementation, due to the recursive bottleneck burden. The existence of feedback loops imposes a bound in the achieved throughput rate and usually results in nonlocalized communications and nonlocalized timing. Specifically, the maximum latency of the feedback loops determines the maximum allowed throughput rate. Fortunately, the recursive bottleneck may be overcome by *recasting* the algorithm using the principle of look-ahead computation, in order to increase the number of delays in the feedback loops and *retiming* to effectively pipeline the computation within the loops [25], [33], [34]. Other techniques for fast processing of recursive algorithms are the bit-level pipelining [35], [36] and the block processing [23], [29], [34], [37] (and the references therein). Also the use of internal local feedback loops, whenever this is possible, increases the throughput rate [23], [26], [33].

The present paper refers to the fast implementation of 3-D FIR and IIR digital filters via VLSI array processors. A systolic-like architecture is presented, which is comprised of similar PEs in a linear cascade configuration, with local communications. The resulting structures are modular, regular, with local interconnections. Concurrency is achieved by exploiting both pipelining and parallelism. The proposed VLSI implementation of the 3-D digital filters attains high throughput rates which meet the requirement of high sampling rates in the real-time processing applications.The FIR and IIR 3-D digital filters considered may be used for the processing of reconstructed 3-D images, such as medical or geophysical images.

II. Direct Form Realization of 3-D Digital Filters

A. 3-D FIR Digital Filters

The 3-D FIR linear digital filters may be described by the 3-D nonrecursive difference equation

$$y(n_1, n_2, n_3) = \sum_{i_1=0}^{N_1} \sum_{i_2=0}^{N_2} \sum_{i_3=0}^{N_3} a_{i_1,i_2,i_3} u(n_1 - i_1, n_2 - i_2, n_3 - i_3) \tag{1}$$

where $y(n_1, n_2, n_3), u(n_1, n_2, n_3)$ represent the input and output 3-D sequences respectively.

The *direct form realization* of the 3-D FIR filter (1) may be seen as an extension of the direct form realization of the 2-D FIR filter [38]; specifically now each coefficient of the 2-D filter is replaced by an 1-D polynomial in the third variable. The three independent variables n_1, n_2, n_3 are associated with three distinct *Unit Delay* (UD) elements UD_1, UD_2, UD_3, corresponding to the variables z_1, z_2, z_3, which appear in the 3-D filter's transfer function

$$H(z_1, z_2, z_3) = \sum_{i_1=0}^{N_1} \sum_{i_2=0}^{N_2} \sum_{i_3=0}^{N_3} a_{i_1,i_2,i_3} z_1^{-i_1} z_2^{-i_2} z_3^{-i_3} \tag{2}$$

B. 3-D IIR Digital Filters

The 3-D IIR linear digital filters may be described by the 3-D recursive difference equation

$$\begin{aligned} y(n_1, n_2, n_3) \; = \; & \sum_{i_1=0}^{N_1} \sum_{i_2=0}^{N_2} \sum_{i_3=0}^{N_3} a_{i_1,i_2,i_3} u(n_1 - i_1, n_2 - i_2, n_3 - i_3) \\ & - \sum_{i_1=0}^{M_1} \sum_{i_2=0}^{M_2} \sum_{i_3=0}^{M_3} b_{i_1,i_2,i_3} y(n_1 - i_1, n_2 - i_2, n_3 - i_3) \\ & (i_1, i_2, i_3) \neq (0, 0, 0) \end{aligned} \tag{3}$$

in analogy to the 2-D quarter-plane model [38]. Eq. (1) should be computable according to the selected scanning. There is not any restriction among the triples of indices (M_1, M_2, M_3) and (N_1, N_2, N_3), since causality in space is not a necessary condition for computability.

The 2-D transfer function, associated with (3), is given by the real rational function

$$H(z_1, z_2, z_3) \; = \; \frac{a(z_1, z_2, z_3)}{b(z_1, z_2, z_3)} = \frac{\sum_{i_1=0}^{N_1} \sum_{i_2=0}^{N_2} \sum_{i_3=0}^{N_3} a_{i_1,i_2,i_3} z_1^{-i_1} z_2^{-i_2} z_3^{-i_3}}{1 + \sum_{i_1=0}^{M_1} \sum_{i_2=0}^{M_2} \sum_{i_3=0}^{M_3} b_{i_1,i_2,i_3} z_1^{-i_1} z_2^{-i_2} z_3^{-i_3}} \quad (i_1, i_2, i_3) \neq (0, 0, 0) \tag{4}$$

The two known forms of direct realization exist also here, in analogy to the 1-D and 2-D ones. The *direct form I realization* results from the cascade configuration of the nonrecursive

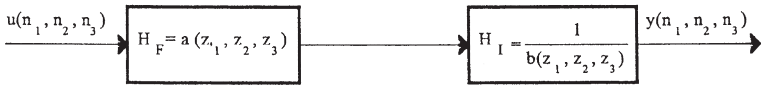

Figure 1. Block diagram of the direct form I realization of a 3-D IIR digital filter.

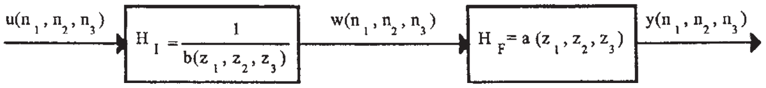

Figure 2. Block diagram of the direct form II realization of a 3-D IIR digital filter.

3-D FIR filter

$$H_F(z_1, z_2, z_3) = a(z_1, z_2, z_3) \tag{5}$$

with the recursive 3-D all-pole IIR filter (Fig. 1).

$$H_I(z_1, z_2, z_3) = 1/b(z_1, z_2, z_3) \tag{6}$$

On the contrary, the *direct form II realization* of the 3-D filter (3) results from the cascade configuration of the subfilters $H_F(z_1, z_2, z_3)$, $H_I(z_1, z_2, z_3)$ in reverse order (Fig. 2). The space-invariance property [39] of the filter considered ensures that the transfer function remains unchanged in both realizations.

The direct form II realization of a 3-D IIR filter is described by the equations:

$$w(n_1, n_2, n_3) = u(n_1, n_2, n_3) - \sum_{i_1=0}^{M_1} \sum_{i_2=0}^{M_2} \sum_{i_3=0}^{M_3} b_{i_1, i_2, i_3} w(n_1 - i_1, n_2 - i_2, n_3 - i_3) \\ (i_1, i_2, i_3) \neq (0, 0, 0) \tag{7a}$$

$$y(n_1, n_2, n_3) = \sum_{i_1=0}^{N_1} \sum_{i_2=0}^{N_2} \sum_{i_3=0}^{N_3} a_{i_1, i_2, i_3} w(n_1 - i_1, n_2 - i_2, n_3 - i_3) \tag{7b}$$

where $w(n_1, n_2, n_3)$ is an intermediate variable. Here the delays UD_1, UD_2 in the directions of n_1 and n_2, corresponding to the variables z_1 and z_2 of the transfer function, may be shared by the subfilters $H_I(z_1, z_2, z_3)$ and $H_F(z_1, z_2, z_3)$.

Consider the *Row by Row, Plane by Plane (RRPP) scanning*, in a 3-D frame of size $J_1 \times J_2 \times J_3$, where the inputs are processed sequentially along the three rectangular axes; then the mapping of the spatial pile (n_1, n_2, n_3) to the lexicographic index is determined by the index mapping

$$I(n_1, n_2, n_3) = n_1 + J_1 n_2 + J_1 J_2 n_3 \tag{8}$$

The adoption of the RRPP scanning, implies that the *Z-Transform* (ZT) operator is associated to the unit delays UD_1, UD_2, UD_3 (denoted as simple delays by 1 in the corresponding equations), according to the following relations:

$$z_1^{-1} ZT[x(n_1, n_2, n_3)] = ZT[x(n_1 - 1, n_2, n_3)] \tag{9a}$$

$$z_2^{-1} ZT[x(n_1, n_2, n_3)] = ZT[x(n_1, n_2 - 1, n_3)] \tag{9b}$$

$$z_3^{-1} ZT[x(n_1, n_2, n_3)] = ZT[x(n_1, n_2, n_3 - 1)] \tag{9c}$$

which show the correspondance of the variables z_1, z_2, z_3 with the unit delays UD_1, UD_2, UD_3.

III. Fast Implementation of 3-D Filters Via Systolic Arrays

In this section we describe implementation structures of the 3-D FIR and IIR filters via VLSI array processors, which are based on the nonrecursive direct form realization and on the recursive direct form II realization respectively.

A. 3-D FIR Digital Filters

The systolic arrays implementation of a 3-D FIR digital filter consists of a 2-D array of *Processing Units* (PUs), which are locally interconnected (Fig. 3a). Each PU is formed as a cascade configuration of N_1+1 elementary PEs (Fig. 3b). There are in total $(N_2+1)(N_3+1)$ PUs denoted as $PU(i, j)$, $i = 0, 1, \dots, N_2$, $j = 0, 1, \dots, N_3$. Moreover, each $PU(i, j)$ operates with a separate input the sample $u(n_1, n_2 - i, n_3 - j)$. Thus the delays D_2, D_3, which due to the structure of the RRPP scanning correspond to large delays ($D_2 = J_1 D_1$, $D_3 = J_1 J_2 D_1$), do not have to be implemented.

The structure of each PE is shown in Fig. 3c. The output of the PE is given by

$$v_{i_2,i_3} = \sum_{i_1=0}^{N_1} a_{i_1,i_2,i_3} w(n_1 - i_1, n_2 - i_2, n_3 - i_3) \tag{10}$$

Considering that the delay of local communication is negligible, the delay UD_1 is considered to be equal to the time T needed to execute the operations in a PE, i.e.

$$UD_1 = T = Mu + Ad \tag{11}$$

where Mu and Ad denote the time needed to execute one multiplication and one addition respectively.

The whole structure operates with *column block pipelining* [40]. The maximum allowed *throughput rate* is therefore determined by

$$R \leq \frac{1}{T} \tag{12}$$

Thus considering $Mu = 115$ ns and $Ad = 19$ ns for the 16 bit multipliers and adders [41],

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS

Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS

Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS

Sync your system to PACER to automate legal marketing.

WHAT WILL YOU BUILD? | sales@docketalarm.com | 1-866-77-FASTCASE

fastcase® Smarter legal research.