A Modular Pipelined Macro-Block for High-Speed Digital Filtering Applications

Frank K. Gürkaynak and Yusuf Leblebici
Worcester Polytechnic Institute
Department of Electrical and Computer Engineering
100 Institute Road, Worcester, MA 01609

***Abstract*—A programmable, fully pipelined macro-block (Aries), that can be used as a modular convolution engine to design digital FIR filters of any size with a minimum clock cycle of 20 ns. is presented. The proposed module is very compact and occupies an active silicon area of less than $1.5\,mm^2$ in a conventional $0.8\mu m$ digital CMOS technology, allowing a large number of Aries macro-blocks to be easily embedded in a larger design to realize digital filters of any dimensions.**

I. Introduction

Digital filters have become indispensable for a large number of applications, yet the modular VLSI realization of digital filters still remains a challenging task. Most of the digital filter architectures are based on calculating the sum of products of discrete data samples and filter coefficients. Using the conventional approach, an n dimensional convolution requires n multiplications of data points with their corresponding filter coefficients, and $n-1$ additions to accumulate the result. As the signal has to be processed continuously for real-time applications, the bandwidth of the signal puts strict constraints on the speed of operation. For the processing of video signals where 10 to 75 frames, (each made up of as many as 2 million pixels) have to be processed each second. This operation must be completed on the order of nanoseconds for real-time video processing applications.

The commonly used realizations of digital filters employ arrays of multipliers [1]. In the proposed Aries architecture, a bit-plane level distributed arithmetic method is used, that completely eliminates the multipliers. The data is broken down into its bit-planes and the data bits of the same order are grouped to form a RAM address. A pre-computed partial result is read from a RAM and the partial results corresponding to these data bits are then weighted and accumulated in an adder array to form the result. This structure results in a very compact realization of the basic weighted sum-of-products operation, which is the basis of the proposed macro-block.

II. Realization of the Sum of Products Operation

The majority of the digital filter implementations are based on computing the sum of products of data values corresponding to an input signal and fixed filter coefficients. It can be shown that any n dimensional filtering algorithm can be represented by a one dimensional sum-of-products using appropriate transformation of indices in the form of:

$$y_{i(n)} = \sum_{j=0}^{N(n)-1} c_j \cdot x_{(i(n),j)} \tag{1}$$

where $x_{(i(n),j)}$ is the $i(n)$th data value of an n dimensional signal, c_j is the corresponding filter co-efficient and $y_{i(n)}$ is the $i(n)$th result of the filter. This operation requires $N(n)$ multiplications and $N(n)-1$ additions for each value of $y_{i(n)}$, where the number N is the length of the one dimensional array that corresponds to the n dimensional filtering space. As multiplication is costly both in terms of speed and area for integrated circuit realizations, a number of methods have been proposed to simplify the computation of sum of products. In bit-plane filters [2], each input sample x_p can be expressed in a binary form as:

$$x_p = \sum_{q=0}^{m} x_{p,q} \cdot 2^q \ , \ (x_{p,q} \in \{0,1\}) \tag{2}$$

Substituting x_p in Eq. (1) we get:

$$y_{i(n)} = \sum_{j=0}^{N(n)-1} c_j \cdot \sum_{q=0}^{m} x_{(i(n),j),q} \cdot 2^q \tag{3}$$

Rearranging Eq. (3):

$$y_{i(n)} = \sum_{q=0}^{m} 2^q \cdot \sum_{j=0}^{N(n)-1} c_j \cdot x_{(i(n),j),q} \tag{4}$$

The inner sum in Eq. (4) represents the sum of products of a "bit-plane". As $x_{(i(n),j),q}$ has a 1-bit value, the calculation of this sum does not involve any multiplications. The outer loop can be realized using shift and add

0-7803-5491-5/99/$10.00 © 1999 IEEE

operations, thus eliminating the need for multiplications completely [3].

The distributed arithmetic method [4] stores the pre-computed result of the inner loop in $2^{N(n)}$ memory locations. As the memory storage requirements increase exponentially with increasing filter size, this method is feasible only for small filter sizes $N(n)$. For larger filter sizes the inner sum can be broken down into k summations of size T where $k = N(n)/T$. Eq. (4) becomes:

$$y_{i(n)} = \sum_{q=0}^{m} 2^q \cdot \begin{pmatrix} \sum_{j=0}^{T-1} c_j \cdot x_{(i(n),j),q} + \\ \sum_{j=T}^{T-1} c_j \cdot x_{(i(n),j),q} + \\ \cdots \\ \sum_{j=(k-1)T}^{kT-1} c_j \cdot x_{(i(n),j),q} \end{pmatrix} \quad (5)$$

It can be seen that the result for a $N(n)$ size filter can be calculated as the sum of k distributed sums of size T.

$$y_{i(n)} = \begin{pmatrix} \sum_{q=0}^{m} 2^q \cdot \sum_{j=0}^{T-1} c_j \cdot x_{(i(n),j),q} + \\ \sum_{q=0}^{m} 2^q \cdot \sum_{j=T}^{2T-1} c_j \cdot x_{(i(n),j),q} + \\ \cdots \\ \sum_{q=0}^{m} 2^q \cdot \sum_{j=(k-1)T}^{kT-1} c_j \cdot x_{(i(n),j),q} \end{pmatrix} \quad (6)$$

$$y_{i(n)} = y_{i(n),1} + y_{i(n),2} + \cdots + y_{i(n),k} \quad (7)$$

In this work, we propose a compact macro-block based on distributed arithmetic with a fixed small filter size ($T = 5$). The small filter size enables the use of a compact RAM lookup table for partial results. Eq. (6) and (7) show that a digital filter of any size $N(n)$ and dimension n can be realized by only using smaller filters of size T. The proposed architecture includes necessary hardware to combine the results of different Aries blocks to effectively realize digital FIR filters of higher complexity.

III. Description of the Architecture

The proposed macro-block core architecture is designed to compute the result of a one-dimensional digital FIR filter of size 5, for 8-bit data inputs. A full custom implementation of the Aries architecture can be seen in Figure 1. The pipelined architecture of the block enables operating speeds of up to 400 Megabits/second.

A five level shift register is used at the input to provide simultaneous access to all data values for computation. The output of the shift register is also provided and can be used as the input of another, cascaded Aries block to increase the filter size in a given dimension.

The core uses the distributed arithmetic method and stores the $2^5 = 32$ possible pre-computed results of the inner loop in a 32 location RAM. To achieve higher processing speed, Aries also uses the time multiplexing scheme

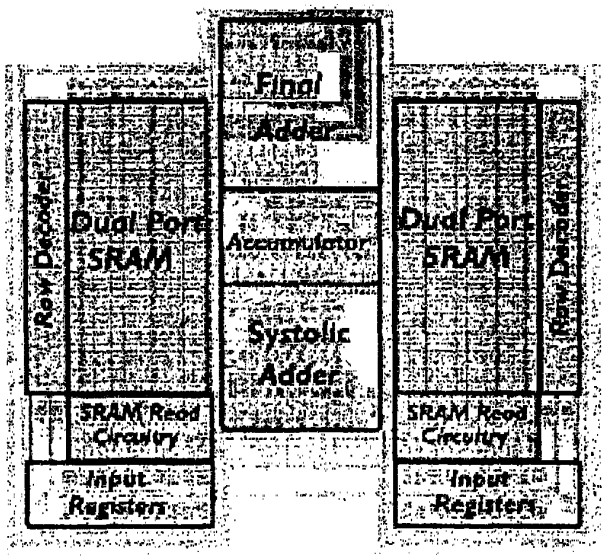

Fig. 1. Layout of the Aries macro-block. The layout occupies an area of $1.5\,mm^2$ in a conventional $0.8\mu m$ digital CMOS technology.

to operate only on 4 bit-planes at any given time. Two dual-port RAM arrays were used in the Aries architecture, to achieve a better area and speed trade-off. The programmable memory enables the filter coefficients to be changed during operation (albeit at a slower rate than the data input rate), which makes Aries suitable for adaptive filtering applications.

As a result of time multiplexing, the inner core of the pipeline operates at twice the speed of the input data rate. During the first cycle, the lower order 4 bits of the inputs are processed. The partial sums of are shifted and added together in a systolic adder structure to form an intermediate result. During the following clock cycle, the intermediate result of the higher order 4-bits are computed. Both results are then accumulated in a separate adder stage. Depending on the mode of operation the result may be passed to an additional adder stage, to be combined with the results of other Aries blocks; otherwise the result is made available at the output. Figure 2 illustrates the timing diagram of the entire pipeline architecture in detail.

IV. High Level Implementation of the Aries Architecture

A single Aries block is only capable of realizing a (5x1) digital FIR filter on a 8 bit input data stream. The strength of the architecture lies in the flexibility of its expansion capabilities. From Eq. (7), it can be seen that the result of a digital filter of size $N(n)$ can be computed by accumulating the results of $k = N(n)/T$ digital filters of size T. The final adder of the Aries architecture can

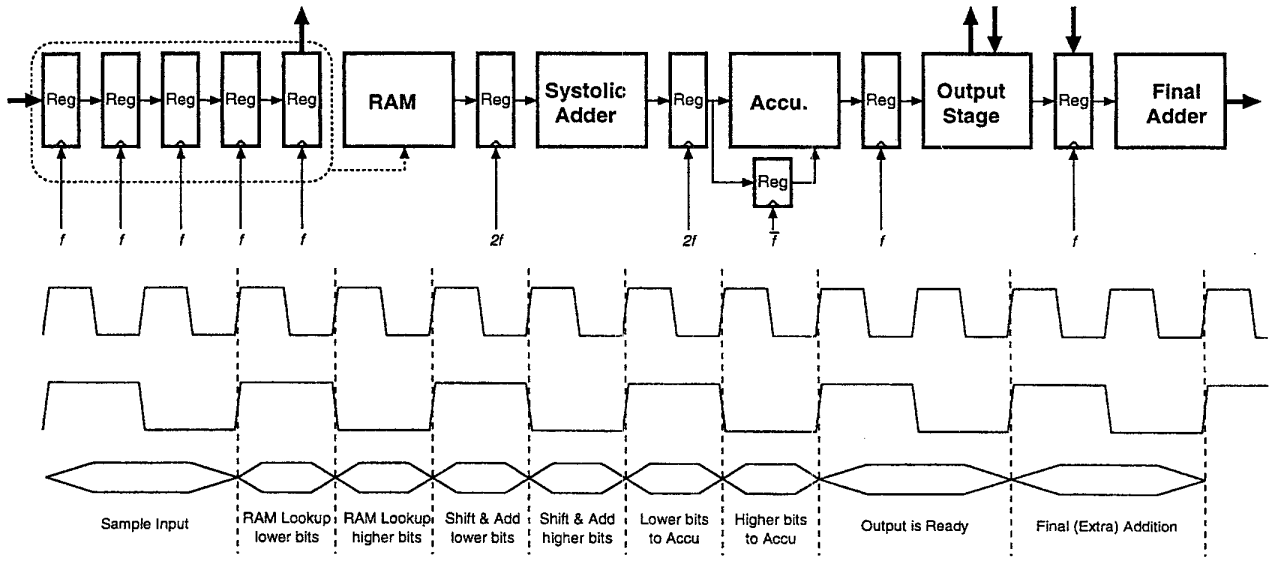

Fig. 2. Timing diagram of the Aries pipeline architecture.

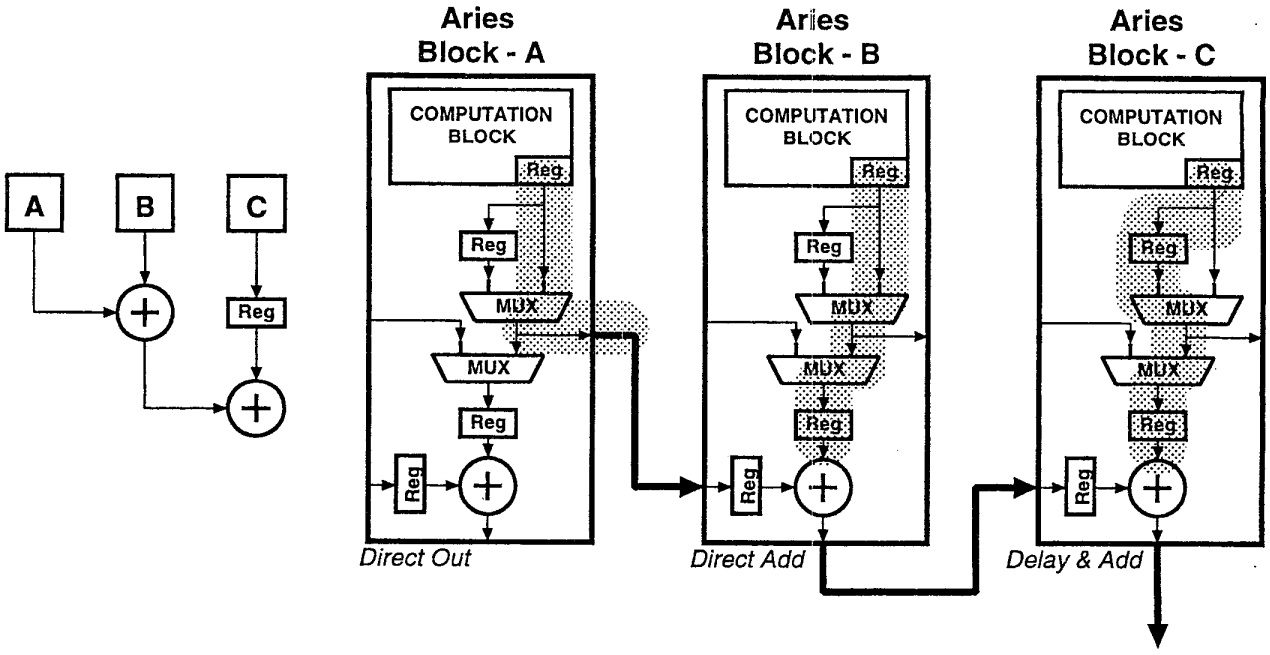

Fig. 3. Three modes of operation of the final adder stage: Direct output, direct addition and delayed addition.

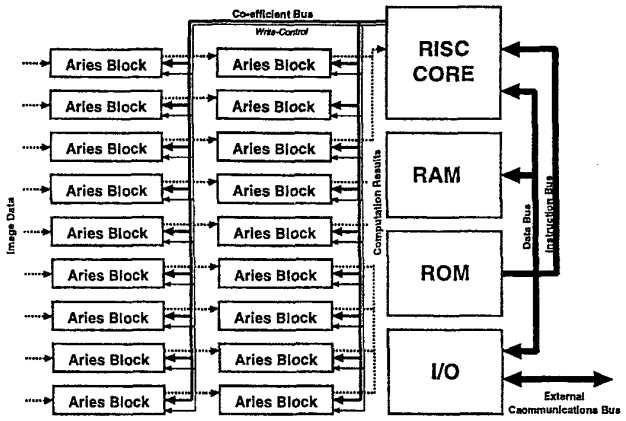

Fig. 4. Block diagram of a RISC based high level digital filter structure using multiple Aries blocks.

be configured to operate in one of three modes that are shown in Figure 3.

The final adder can add either the results of two separate Aries blocks, or accumulate the internally generated result with the result of another Aries block. To generate a fully pipelined adder chain that does not have exactly 2^n components, it is also necessary to introduce delay elements in the pipeline. An additional control signal is used to delay the internal result by one clock cycle. In Figure - 3, the result of Block C is delayed for one clock cycle to "wait for" the addition of results from Blocks A and B.

Using these three modes it is possible to create a pipelined adder tree with n Aries blocks that can sum up the results of these n blocks. Thus, the proposed modular architecture can be used to realize any FIR filter with arbitrary length and dimensions. For example, an array of 18 Aries blocks will implement a (9x9) 2-dimensional digital FIR filter. Using the same $0.8\mu m$ technology, the filter would occupy a core area of roughly $30\,mm^2$ and operate at 50 MHz data rate for a combined computation power equivalent to approximately 300 GOPS.

The capabilities of this modular architecture can be further enhanced using a small dedicated RISC core to control an array of Aries blocks. The block diagram for this implementation is shown in Figure 4. The RISC core in this case will be able to reconfigure the Aries blocks and provide and update the pre-computed coefficients. This scheme can also be employed for adaptive filtering applications. Another task of the RISC core would be to perform normalization operations on the results.

V. Conclusions

In this paper, we propose a compact modular core architecture that can be used to construct digital filters of arbitrary dimensions. The basic core realizes a (5x1) digital FIR filter of inputs, and occupies a silicon area of less than $1.5\,mm^2$. The results of any number of Aries blocks can be combined by forming a pipelined adder chain, utilizing the final adder stage of the module. The proposed modular architecture can operate on data rates of up to 50 MHz, enabling it to be employed even in the most demanding high-resolution image filtering applications.

References

[1] M. A. Bayoumi and E. E. Swartzlander, Jr., "VLSI Signal Processing Technology", Kluwer Academic Publishers, 1994.

[2] P. Pirsch, "Architectures for Digital Signal Processing," John Wiley and Sons, 1998.

[3] Y. Leblebici, A. Kepkep, F. K. Gürkaynak, H. Özdemir and U. Çilingiroğlu, "A fully pipelined programmable real-time (3x3) image filter based on capacitive threshold-logic gates," in Proc. IEEE Int. Symp. Circuits and Systems, 1997, pp. 2072-2075, 1997.

[4] C. S. Burrus, "Digital Filter Structures Described by Distributed Arithmetic," IEEE Trans. on Circuits and Systems, Vol CAS-24, No12, pp 674-680, Dec. 1977.