A VECTOR ARCHITECTURE FOR HIGHER-ORDER MOMENTS ESTIMATION

José C. Alves, André Puga, Luís Corte-Real and José S. Matos

FEUP - Faculdade de Engenharia da Universidade do Porto
INESC - Instituto de Engenharia de Sistemas e Computadores
Praça da República, 93 - 4007 Porto CODEX - PORTUGAL

ABSTRACT

Higher-order statistics extend the analysis methods of non-linear systems and non-gaussian signals based on the autocorrelation and power spectrum. The main drawback of their use in real time applications is the high complexity of their estimation due to the large number of arithmetic operations. This paper presents an experimental vector architecture for the estimation of the higher-order moments. The processor's core is a pipelined multiply-accumulate unit that receives four data vectors and computes in parallel the moment taps up to the fourth-order. The design of custom cache memory organization and address generation circuits has led to more than 11 operations per clock cycle. The architecture was modeled and simulated in Verilog and is presently being implemented in XILINX field-programmable gate arrays (FPGAs) and one custom integrated circuit for the multiply-accumulate unit.

1. INTRODUCTION

Higher-order statistics (HOS) have been applied to a large set of engineering problems, for instance pattern recognition [1] and real-time image processing [2]. These statistics, namely the higher-order cumulants, are particularly useful when it is necessary to deal with non-linearities or non-gaussianities [3].

The 2nd, 3rd and 4th-order cumulants of a real one-dimensional, zero mean data process are defined as:

$$C_2^y(\tau_1) = m_2(\tau_1) \tag{1}$$

$$C_3^y(\tau_1, \tau_2) = m_3(\tau_1, \tau_2) \tag{2}$$

$$\begin{aligned} C_4^y(\tau_1, \tau_2, \tau_3) \;=\; & m_4(\tau_1, \tau_2, \tau_3) - \\ & m_2(\tau_1) \cdot m_2(\tau_2 - \tau_3) - \\ & m_2(\tau_2) \cdot m_2(\tau_3 - \tau_1) - \\ & m_2(\tau_3) \cdot m_2(\tau_1 - \tau_2) \end{aligned} \tag{3}$$

This work is supported by Junta Nacional de Investigação Científica e Tecnológica (JNICT), contract no. PBIC/TIT/2489/95 — HAREOS

where

$$m_2(\tau_1) = E[x(k) \cdot x(k+\tau_1)] \tag{4}$$

$$m_3(\tau_1, \tau_2) = E[x(k) \cdot x(k+\tau_1) \cdot x(k+\tau_2)] \tag{5}$$

$$m_4(\tau_1, \tau_2, \tau_3) = E[x(k) \cdot x(k+\tau_1) \cdot x(k+\tau_2) \cdot x(k+\tau_3)] \tag{6}$$

are the 2nd, 3rd and 4th moments. The estimation of the higher-order moments is the most complex task of the higher-order cumulants estimation algorithm [3], due to the large number of arithmetic operations required.

Two classic examples that show the usefulness of higher order cumulants are the following systems:

$$y_I(k) = x(k) - 0.85 \cdot x(k-1) + 0.175 \cdot x(k-2) \tag{7}$$

$$y_{II}(k) = x(k) + 0.85 \cdot x(k+1) + 0.175 \cdot x(k+2) \tag{8}$$

If a method of system identification based on the second-order cumulant is used, the systems cannot be distinguished because these cumulants are identical. However, system identification methods based on the 3rd-order cumulant can be successfully applied to identify both systems because the cumulants have different values for the two systems [3].

Some work has been done on the development of dedicated hardware architectures [4] and software implementation on parallel machines [5] for the real time estimation of higher-order moments. However, these implementations require large computation arrays and expensive parallel machines.

This paper presents ProHos, an experimental vector architecture for the estimation of the higher-order moments, suitable to be included in a low cost PC expansion board. The processor's core is a pipelined multiply-accumulate unit that receives four data vectors and computes in parallel the moment taps up to the fourth-order. The design of custom cache memory organization and address generation circuits has led to an average performance exceeding 11 operations per

clock cycle. A first version of the architecture has been simulated using a model written in Verilog hardware description language [6], that helped us to tune the control section of the processor [7]. The present prototype is being built using XILINX FPGAs [8] for the control part and one custom designed integrated circuit that implements the pipelined multiply-accumulate unit.

2. QUANTIZATION ERROR

The hardware complexity of the processor depends mainly on the multiply-accumulate unit, namely on the width of the input data applied to the multipliers. However, a reduction of the number of bits used to represent the input data implies an increase of the quantization error. To evaluate the influence of this error on the numerical precision of the estimated cumulants, a set of simulations have been done using the systems described by equations (7) and (8).

Exponential distributed white noise with mean equal to 1 has been applied to each system and the second, third and fourth-order cumulants of the output signal have been estimated. The estimates have been compared with the cumulants obtained without quantization error, using the following precision measure:

$$Precision = 100\left(1 - \frac{\|\hat{C}^q - \hat{C}\|}{\|\hat{C}\|}\right)\%$$

where $\hat{C}^q$ is the estimate of the cumulant with quantization error, $\hat{C}$ is the estimate without quantization error, and the norm is the Forbenious norm.

Figure 1 presents the worst case of 10 independent runs for both systems of 1000 data samples each, obtained by quantizing the input data into words ranging from 8 to 16 bits. It can be verified that, to achieve 99.9% precision it will be necessary to use at least 12 bit words to represent the input data, and for these two systems the effects of quantization error on the final results are irrelevant when more than 16 bits are used.

3. PROHOS ARCHITECTURE

To compute the estimates of higher-order moments, a dedicated architecture was developed—ProHos (fig. 2). The processor implements the hardest task of the moment estimation algorithm, computing the moment taps for the several segments the data vector is divided into. The present architecture assumes that the input is a normalized one-dimensional data vector (zero mean, unit standard deviation), although a normalization procedure will be included in a future version. The computation core is a pipelined multiply-accumulate unit that processes four data vectors and computes in

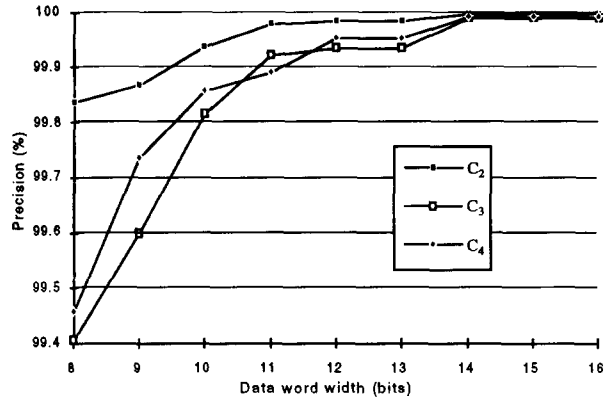

Figure 1: Variation of the precision of the cumulant estimation with the data word width

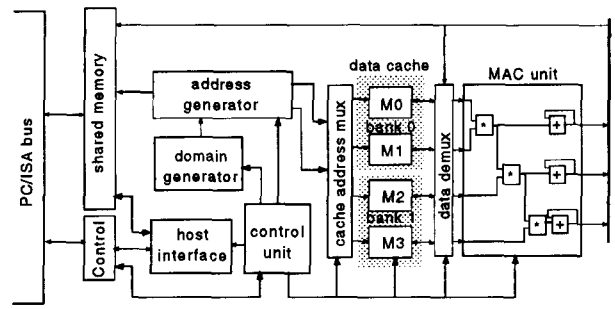

Figure 2: ProHos architecture

parallel the taps of the 2nd, 3rd and 4th-order moments. The input data vector is read from a RAM memory shared with the host processor, that is also used to pass some additional parameters. To overcome the limited bandwidth of this memory, four static RAMs are used as data cache to store one data segment at a time and supply their elements to the multiply-accumulate unit at the required order to compute each moment tap. The sequence of data samples delivered to the multiply-accumulate unit is generated by custom control circuits built around binary counters, that compute efficiently the addresses of the data cache memories, with little intervention of the main controller. The following sections present in detail the main blocks of ProHos.

3.1. Multiply-accumulate unit—MAC

The processor's computation core is the multiply-accumulate unit that computes the moment taps (fig. 3). The unit has three cascaded multipliers, with one accumulator at the end of each one.

To compute $m_4(\tau_1, \tau_2, \tau_3)$, $m_3(\tau_1, \tau_2)$ and $m_2(\tau_1)$, four copies of the input data segment are sent to the inputs of the MAC unit, starting at samples 0, τ_1, τ_2

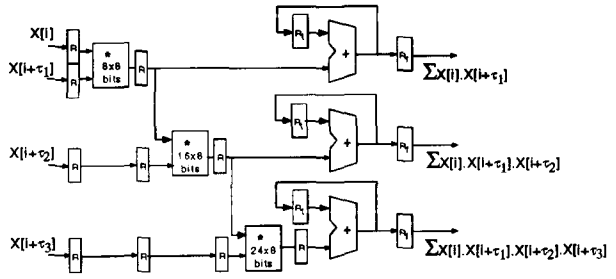

Figure 3: Multiply-accumulate unit

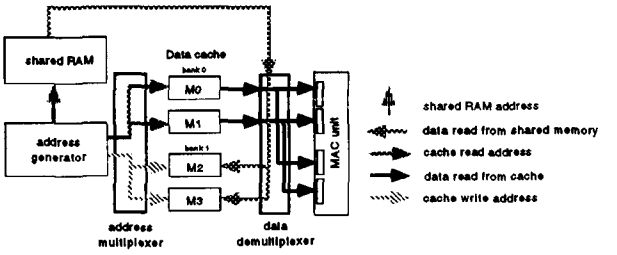

Figure 4: Operation of the data cache memories

and τ_3. As the unit computes the coefficients of the 4th-order moment, the 3rd and 2nd order moments are obtained as a side effect of this calculation. This way, the utilization of the multipliers in this unit is not optimized, because the lower-order moments are computed several times during the computation of the fourth-order moment. Besides the clock and reset lines, the whole unit is controlled by a single line, that enables the load of the output register of each acc mulator and u clears its temporary register. This line is asserted in the same cycle the last set of four data samples is sent to the unit, and three delay modules (not represented in fig. 3) introduce delays equal to the number of pipe stages of each input-output path. The present unit has 5 pipe stages and 3 combinational parallel multipliers that have been obtained by the ES2 module generator. To increase the clock frequency, pipelined multipliers can be used, what increases the pipe stages of the MAC unit. However, this has little impact in the overall performance, because once the unit is filled-up, it has to be emptied only at the end of the computation of the whole data vector, and the number of clock cycles needed for this is only a small fraction of the total execution time.

3.2. Data cache memories

If a single data memory was used, it should be read four times within each clock cycle, to send to the MAC unit the four data samples needed per clock cycle. This would impose a long clock cycle for the operation of the MAC unit, requiring a large and fast static RAM to store the whole data vector. In addition to this, it will be desirable to include a pre-normalization of each data segment that would be impractical if a single memory was used. By the moment, this task is left to the host processor.

To overcome this, a set of 4 static RAMs operate as data buffers, that accommodate one data segmentat a time. These memories are organized in two banks, each one holding two copies of one data segment. While the two memories of one bank are read twice in each clock cycle to send four data samples to the MAC unit, the other two memories are loaded with the next segment to be processed (fig. 4). To access each cache memory twice and place the four data samples at the inputs of the multipliers, address multiplexers and data demultiplexers are used to load the four registers that drive the inputs of the multiply-accumulate unit.

When the computation of the current segment ends, the function of the cache banks is switched and the next segment is loaded. Because each segment is kept in the cache memories during the computation of all its moment taps, the operation of loading a new segment is not time critical and will accommodate in the future the normalization procedure.

3.3. Cache address generation

The address generation unit produces the addresses needed to read and write the cache memories and to read the data samples from the shared memory.

The shared memory is always accessed in sequence and its address is the output of a binary counter initialized with zero. The sequence of addresses to write the data samples into the cache memories is the output of another binary counter, that is reset to zero before a new segment is loaded.

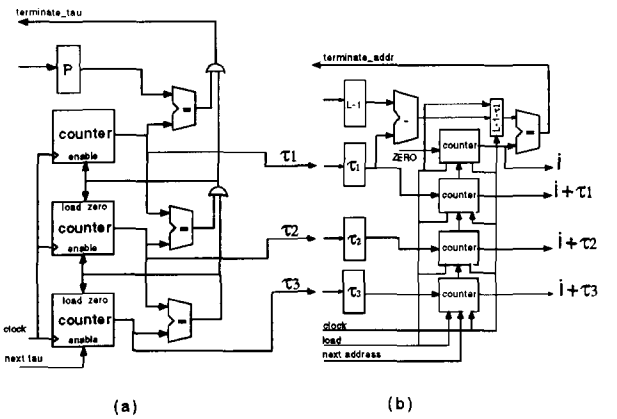

Figure 5: Logic circuits for the τ-generator (a) and cache read address generator (b)

The sequence of addresses needed to read the data

cache memories in the right order for the MAC unit is generated automatically by a dedicated circuit built with binary counters and comparators. This unit is divided in two separate sections: the generator of the domain of the moment functions (τ_1, τ_2, τ_3), and the generator of the read addresses for the data cache memories.

The τ-generator produces the sequence of the (τ_1, τ_2, τ_3) that represents the non-redundant moment taps being computed [7]. Although this operation is not time critical, this is accomplished in the present version by the circuit shown in fig. 5-(a), with the advantage of requiring a single control line for its operation.

To compute $m_4(\tau_1, \tau_2, \tau_3)$, four copies of the data samples $x(i)$ must be sent to the MAC unit, starting at addresses 0, τ_1, τ_2 and τ_3. Because the samples are accessed in sequence, these addresses can be produced by four counters. For each new point (τ_1, τ_2, τ_3), these counters are loaded with the initial values 0, τ_1, τ_2 and τ_3 (figure 5-(b)). The number of iterations is $L - max(\tau_1, \tau_2, \tau_3)$ what simplifies to $L - \tau_1$ because $\tau_3 \leq \tau_2 \leq \tau_1$, where L is the size of data segment being computed.

4. RESULTS AND CONCLUSIONS

The architecture presented here was validated with a mixed behavioral/structural Verilog model. Presently a prototype is being built with XILINX FPGAs and one custom integrated circuit, that will operate as a co-processor of a PC, connected to the ISA bus. The control section (address generators, finite state machines, address/data multiplexers and four small static RAMs with 64 bytes included just for test purposes) has been implemented in one XILINX FPGA. It occupies approximately 70% of the programmable resources of a XC4010-4, (10K gates-equivalent, -4 speed grade) and can be clocked at 16MHz.

The multiply-accumulate unit was designed as a custom integrated circuit using the Cadence DFII system and the ECPD07 (0.7μ) technology from ES2. The multipliers were obtained automatically by the ES2 module generator. The unit receives four 8-bit inputs and produces three results with 26, 34 and 42 bits, respectively the 2nd, 3rd and 4th-order moments. This guarantees no overflow for data segments smaller than 1024 samples. First results of active chip area stay below $7mm^2$, and the circuit can run with a 25MHz maximum clock frequency.

With a 16MHz system clock, limited by the FPGA implementation of the control part, the processor can compute the moment estimates up to the fourth order of a 4000 data samples segment in 55ms. In this example, 285 different coefficients were computed, performing more than 8.6 millions of operations. A sequential implementation, coded in C and optimized for speed runs in 660ms in a PC/Pentium120MHz, what represents 12× the execution time of ProHos.

REFERENCES

[1] Michail K. Tsatsanis and Georgios B. Giannakis. Object and texture classification using higher-order statistics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(7):733–750, jul 1992.

[2] John M. M. Anderson and Georgios B. Giannakis. Image motion estimation algorithm using cumulants. *IEEE Transactions on Image Processing*, 4(3):346–357, mar 1995.

[3] C. L. Nikias and J. M. Mendel. Signal processing with higher-order spectra. *IEEE Signal Processing Magazine*, 10:10–37, 1993.

[4] Haris M. Stellakis and Elias S. Manolakos. An array of processors for the real-time estimation of fourth- and lower-order moments. *Signal Processing*, (36):341–354, 1994.

[5] John N. Kalamatianos and Elias S. Manolakos. Parallel computation of higher-order moments on the maspar-1 machine. In *Proceedings of the International Conference on Acoustics Speech and Signal Processing*, may 1995.

[6] D. Tomas and P. Moorby. *The verilog Hardware Description Language.* Kluwer Academic Publishers, 1991.

[7] J. C. Alves, A. Puga, L. Corte-Real, and J. S. Matos. An efficient co-processor for the estimation of higher-order moments—prohos-1. In *Proceedings of the XI Design of Integrated Circuits and Systems—DCIS'96,*, pages 457–462, Nov 1996.

[8] *XILINX—The Programmable Logic Data Book,1995.*