PARALLEL COMPUTATION OF HIGHER ORDER MOMENTS ON THE MASPAR-1 MACHINE

John N. Kalamatianos *Elias S. Manolakos*

Communications and Digital Signal Processing Center
Electrical and Computer Engineering Department
Northeastern University, Boston, MA 02115
e:mail elias@cdsp.neu.edu

ABSTRACT

The design of efficient parallel processing implementations for speeding up the computationally intensive estimation of Higher-Order Statistics (HOS) has been recognized as an important task by the signal processing community [1]. In this paper we report on the synthesis of minimum running time (latency) *data-parallel* algorithms that can be employed to compute *all* moment lags, up to the 3rd or 4th-order, on the MasPar-1 *Single Instruction Multiple Data* (SIMD) parallel system. By construction the synthesized SIMD algorithms require constant memory per processing element (PE), thus allowing the processing of 1-D input data sequences with as many as $M = 2^{10}$ data samples. Simulation results are presented showing the gain in speedup and execution times, as compared to optimized versions of the serial estimation algorithm running in powerful workstations.

1. INTRODUCTION

Estimating Higher-Order Moments (HOM) from the data has become an increasingly important task in modern signal processing. Third and 4th-order moments play an important role in many applications but are also useful in computing the Bispectrum and Trispectrum [1]. It is well known that their estimation from the input data is a computationally intensive task, especially as the length of the sequence and the order of the desired estimates increases. In such cases, some form of parallel processing is needed if real-time performance is required.

In [2], [3] and [4], a systematic algorithm to architectures transformation approach was used to synthesize regular VLSI arrays for the real-time estimation of moments and cumulants, up to the 4rth-order. In this paper we extend the methodology proposed in [2] in order to be able to derive minimum latency SIMD algorithms for HOMs estimation on the widely availabe MasPar-1 parallel machine. The systematic formulation of efficient algorithms is achieved by allowing constraints related to the architecture of the target machine (MasPar-1) to drive the various stages of the mapping process. Although applied to specific class of algorithms here, *constraint-driven* synthesis of parallel implementations is a general enough methodology that can be used in extracting the parallelism available in a given nested-loop algorithm as long as the index space and data dependencies are known at compile-time.

This work is partially supported by a grant from the National Science Foundation, MIP-9309319.

2. THE ESTIMATION ALGORITHM

The "indirect estimation" procedure [1] was reformulated as a forward-order recursive algorithm in [2], in order to produce all moment estimates $\widehat{m}_k$ (up to order k) directly from M samples of the underlying 1-D process. Let,

$$\widehat{m}_k(i_1, i_2, ..., i_{k-1}) = \frac{1}{M} \cdot \sum_{i=s_1}^{s_2} x_i \cdot x_{i+i_1} \cdots \cdot x_{i+i_{k-1}} \quad (1)$$

where $s_1 = max(0, -i_1, -i_2, ..., -i_{k-1})$ and $s_2 = min(M-1, M-1-i_1, M-1-i_2, ..., M-1-i_{k-1})$. Taking advantage of symmetry properties [5], the domain of support becomes $\mathcal{R}_0 = \{(i_1, i_2, ..., i_{k-1}) : 0 \leq i_{k-1} \leq i_{k-2} \leq ... \leq i_2 \leq i_1 \leq M-1\}$. Then, if we define the k-th order product term $p_k(i_1, ..., i_{k-1}, i)$, where $(i_1, i_2, ..., i_{k-1}) \in \mathcal{R}_0$ and $0 \leq i \leq M-1-i_1$, as

$$p_k(i_1, i_2, ..., i_{k-1}, i) \equiv x_i \prod_{l=1}^{k-1} x_{i+i_l} =$$

$$= p_{k-1}(i_1, i_2, ..., i_{k-2}, i)\ x_{i+i_{k-1}};\ p_1 = x_i, p_0 = 1 \quad (2)$$

we notice that terms $\{p_l,\ l = 1, 2, \ldots k\}$ form the minimal set of products needed to compute the k-th order moments using:

$$\widehat{m}_k(i_1, i_2, ., i_{k-1}) = \frac{1}{M} \sum_{i=0}^{M-i_1-1} p_{k-1}(i_1, i_2, ., i_{k-2}, i) \cdot x_{i+i_{k-1}} \quad (3)$$

0-7803-2431-5/95 $4.00 © 1995 IEEE

Equation (3) illustrates that every moment term $\widehat{m}_k$ can be computed using lower order product terms $\{p_{k-1}\}$ in $O(M)$ steps (we call this order-recursion). The motivation behind the reformulation of (1) as in (3) was the need to utilize lower order product terms effectively in computing subsequent order moments. This property makes the order-recursive algorithm as fast as possible (complexity in $O(M^k)$) at the expense of increased memory requirements (also in $O(M^k)$).

3. SYNTHESIS OF SIMD ALGORITHMS

In this section we address the problem of deriving systematically optimal SIMD algorithms for HOM's estimation. Due to lack of space we only sketch the synthesis methodology used. The target parallel machine chosen to implement the parallelized version of (3) is the SIMD MasPar-1 (MP-1). It consists of a square 64×64 toroidal grid (wrap-around mesh) of PEs, where each PE is connected to its eight nearest neighbors (xnet connectivity) [6]. Each one of the $p = 4096$ PE's is a 4-bit RISC microprocessor with 32 registers (each 32-bits wide) and only 64Kb of local memory.

One can model the HOM's estimation algorithm as a set of computations occurring at every point of an Index Space. This space is a multidimensional lattice of integer points (subset of Z^{n_a}), with the dimensionality $n_a = k+1$ corresponding to the number of nested loops in the serial algorithm's description. The synthesis methodology starts by first transforming the serial version into a suitable Locally Recursive Algorithm (LRA). The LRA may be represented by an acyclic Dependence Graph (DG) with dimensionality $n_a = k+1$, $k \geq 3$, where each node models a Multiply-Accumulate (MAC) operation and each arc a precedence relationship (data dependence) between two computations. By construction of the LRA all dependence arcs have length independent of the problem size. Then a linear space-time transformation is applied to the DG in order to derive a lower dimensional Signal Flow Graph (SFG) [7] that constitutes a behavioral model of the data-parallel algorithm's implementation.

The process of formulating a suitable LRA and choosing the appropriate space-time transformation involves two related challenging tasks: (a) the mapping of a higher dimensional DG into a 2-D SFG that may be embedded into the MP-1 interconnection network, (b) the choice of appropriate contours of propagation for certain data variables of the serial algorithm that minimizes interprocessor communications and local memory requirements after the space-time mapping. As a result of the systematic synthesis the MP-1 implementation uses only fast xnet type of communications. Furthermore, all MAC operations in (3) needed to produce an $\widehat{m}_k$ term are allocated to the same PE thus maximizing the amount of work/PE while using a *constant* amount of local memory, not depending on the sequence length M as long as $M \leq \sqrt{p} = 64$. It can be proved analytically that the parallel implementation attains the lower bound on parallel execution time for the given problem on a mesh with $O(1)$ local memory per PE.

We applied the aforementioned methodology to derive two data-parallel algorithms for producing *all* moment lags, up to the-3rd and 4th-order respectively, since these are the most commonly demanded in practice. The resulting array for the 3rd-order moments case is triangular $M \times M$ as shown in Fig.1, and was directly embedded onto the two-dimensional torus of MP-1.

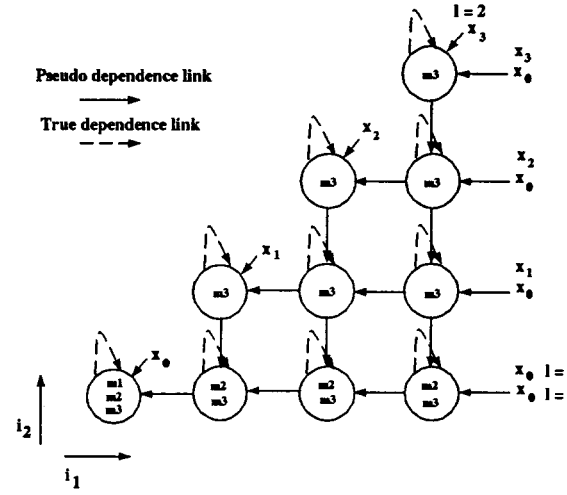


Figure 1: 3rd-order moments array structure, $M = 4$.

Each array node models k computations, $(k = 3)$ and is allocated onto an MP-1 PE. Arcs represent communications needed at specific time instants determined by the minimum latency linear timing schedule selected. Solid-lined arcs correspond to propagation/broadcasting of input data from one PE to another (*pseudo dependence links*), while dotted-lined self-loops model the reuse of partial results to produce new ones (*true dependence links*). By selecting an appropriate space transformation all product terms needed to compute a specific moment term are produced (in an order-recursive manner according to (2)) and accumulated (according to (3)) locally in the corresponding PE. Therefore there is no communication overhead associated with true dependencies. Furthermore, the communication overhead due to input data distribution (pseudo dependencies) is minimized by exploiting the fast *xnet* network of MP-1. At the end of the algorithm's execution $\widehat{m}_3(i_1, i_2)$ resides in PE_{i_1,i_2}, $\widehat{m}_2(i_1)$ in $\mathrm{PE}_{i_1,0}$ and $\widehat{m}_1$ in $\mathrm{PE}_{0,0}$. The timing schedule determines the arriving sequence of input data. All data

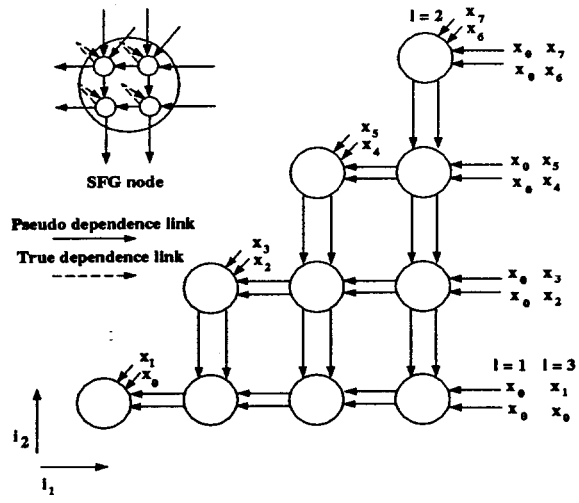


Figure 2: Partitioned algorithm for the 3rd-order moments estimation, $M = 8, K = 4$.

entering the array during the first schedule time period (first recursion) are shown explicitly in Fig.1; in this figure label $1 \leq l \leq k$ identifies the order of term $p_l(i_1, i_2)$ that needs the corresponding x token for its generation at PE_{i_1,i_2} based on (2). The array for the 4th-order moments has a similar structure but more links, more computations (work) per PE and more input data entering the array.

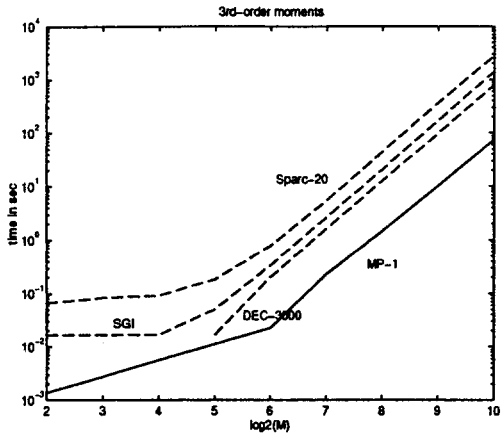
A method of *partitioning* (virtualization) was introduced so that problem sizes with $M > \sqrt{p} = 64$ could be processed on the 64 x 64 processors MP-1 machine. Let us define the *Virtual Processor Ratio* (VPR) in each dimension as VPR= $\frac{M}{K}$ where M is the input sequence length and $K \times K$ the physical PE array employed. The partitioned parallel algorithm for a VPR=2 may be represented by Fig.2. The major difference from Fig. 1 is that every physical PE now models VPR^2 nodes of the non-partitioned array. Therefore the work per PE is increased by a factor in $O(VPR^2)$ and the communication requirements per PE by a factor in O(VPR). Since the major source of overhead relative to the serial algorithm is due to interprocessor communications a better speedup is expected as the VPR factor increases.

4. EXPERIMENTAL RESULTS

Simulations were performed in order to determine the maximum input sequence size that could be processed on the MasPar-1 having only 64Kb local memory per PE. The serial algorithm was also coded in C and run on a Sun Microsystems Sparc20 (Sparc CPU), a an SGI Indigo-2 (MIPS R4400 CPU) and a Digital 3000 Workstation (Alpha AXP-800 CPU). All of them had 64Mb of available RAM memory space. The execution times measured are reported in Fig.3 for both the 3rd and the 4th-order moments cases. As we can see the MP-1 outperforms all workstations except the Digital 3000 on the 4th-order case and for small $M \leq 16$. We should note that the serial algorithm used for comparison is the fastest possible (order-recursive) at the expense of using extra memory. The superiority of the parallel implementation is evident for large records of data that usually lead to better moment estimates. The SIMD version can in general handle larger problem sizes than the serial one, since the memory requirements of the serial version are in $O(M^k)$ while for the data-parallel partitioned version only in $O(\frac{M^2}{p})$ per PE at the expense of introducing communications. This is so, due to the fact that the non-partitioned algorithms ($M \leq 64$) have been synthesized to have only $O(k)$ memory requirements per PE, where $k = 3, 4$ and does not depend on M. In order to be able to obtain results for $M \geq 2^7$ when $k = 3$ ($M \geq 2^5$ when $k = 4$) we had to use dynamic memory allocation for the serial algorithm.

According to our theoretical scalability analysis, that was also verified experimentally but only for small M due to the very small amount of local memory per MP-1 PE, the SIMD algorithm attain *linear speedup* i.e. in $O(p)$. Of more practical interest though is the "speedup" relatively to commonly used workstations that is plotted in Fig. 4 as a function of the input sequence length. It should be noted that the serial machines employed for these simulations use 32-bit and 64-bit microprocessors as opposed to the MP-1 using very simple 4-bit architecture PEs. In Fig.4 the 3rd-order moments curve presents an anomaly as we go from $M = 2^6$ to $M = 2^7$. The reason for it lies in the change of the version of the parallel SIMD algorithm used; $M = 2^6$ is the largest input size that can be used in conjunction with the non-partitioned SIMD version since the target parallel machine is a toroidal mesh with 64 x 64 PEs. So when $2^2 \leq M \leq 2^6$ we used the non-partitioned version, and when $2^7 \leq M \leq 2^{10}$ the partitioned one. Since the latter SIMD algorithm incurs an additional overhead for managing the data partitioning, there is a "knee" in that speedup curve at point $M = 2^6$. However, as the data size increases speedup approaches linearity. There is no such phenomenon observed in the 4th-order moments case, since the serial time is able to override the increase of the parallel time as we go from $M = 2^6$ to $M = 2^7$ step.

In our simulations we performed 64-bit double precision operations (for greater accuracy) that are slow on the MP-1 PEs having only a 4-bit floating point unit per PE, as opposed to the workstation CPUs that operate directly on 64-bit operands. Use of single precision floating-point



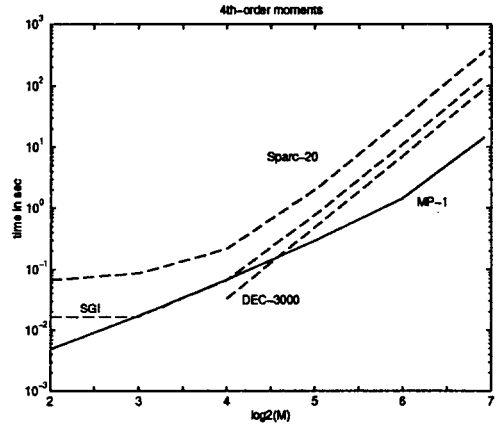
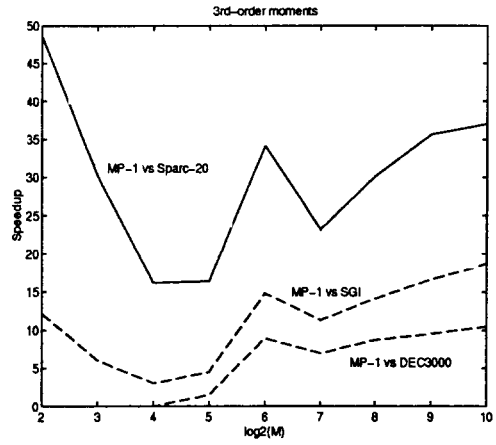


Figure 3: Execution times comparison vs. $\log_2 M$.



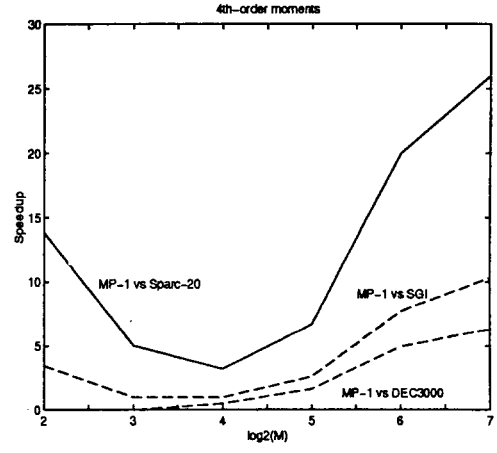


Figure 4: Speedup results vs. $\log_2 M$.

32-bit operands would not change the situation radically but would improve the speedup results in favor of the MP-1.

5. CONCLUSIONS

Data-parallel algorithms were systematically synthesized for the estimation of all moment lags up to the 3rd or the 4th-order and evaluated on the SIMD MP-1 parallel machine. They exhibit minimum latency since their design was driven by the machine's operating model and real characteristics. Performance tests showed not only improved execution times compared to powerful workstations, but also the ability to estimate HOM's of longer data sequences. We are currently investigating the systematic derivation of optimal HOS estimation algorithms suitable for Multiple Instruction Multiple Data (MIMD) massively parallel machines.

6. REFERENCES

[1] C.L. Nikias and J.M. Mendel. Signal Processing with Higher-Order Spectra. *IEEE Signal Processing*, 10(3):10–37, 7/1993.

[2] H.M. Stellakis. *"The Systematic Synthesis of Parallel Architectures for the Estimation of Higher Order Statistics"*. PhD thesis, Northeastern University, 8/1993.

[3] H.M. Stellakis and E.S. Manolakos. "An Architecture for the Real-Time Estimation of Cumulants". In *Proceedings of IEEE Int'l Conference on Acoustics Speech and Signal Processing, (ICASSP'93)*, IV·220–IV·223, 4/1993.

[4] H.M. Stellakis and E.S. Manolakos. "An Array of Processors for the Real-Time Estimation of the Fourth and lower Order Moments". *Signal Processing, (*special issue on Higher Order Statistics*)*, 36(3):341-354, 3/1994.

[5] C. L. Nikias and M. R. Raghuveer. "Bispectrum Estimation: A Digital Signal Processing Framework". *IEEE Proceedings*, 75(7):869–891, 7/1987.

[6] MasPar Computer Corp. *"MasPar-1 Architecture Specification"*, 1992.

[7] S.Y. Kung. *VLSI Array Processors*. Prentice Hall, 1989.