IMPLEMENTATION OF OVERLAPPED BLOCK FILTERING USING SCHEDULING BY EDGE REVERSAL

Charles B. do Prado, Paulo S.R. Diniz and Felipe M.G.França

COPPE/Federal University of Rio de Janeiro
C.P. 68504, Rio de Janeiro, 21945-970,Brazil.
e-mail:bezerra@lps.ufrj.br, diniz@lps.ufrj.br, felipe@cos.ufrj.br

ABSTRACT

Implementation of overlapped block filtering using *Scheduling by Edge Reversal* (SER) is proposed in this paper. SER is a very simple and powerful synchronizer. It allows more efficient implementation of parallel structures. This technique is applied for the first time to FIR filters using the overlapped block digital filtering, and implemented on a parallel computer plataform. The results confirm the expected reduction in computation time.

1. INTRODUCTION

Block digital filtering has been largely used in many practical situations. Firstly, it allows that the frequency spectrum of the signal to be processed in narrower subbands. Since these frequency bands expose the distribution of signal power with more resolution, then specific filtering can be applied to the signal subbands with higher energy using filters with short length.

The basic block diagram of digital filter is shown in figure 1 in which the input sequence is converted into a series of contiguous blocks of length L by means of a serial-to-parallel converter. Each input block is processed simultaneously by an L-input, L-output block digital filter characterized by a transfer matrix $P(Z)$. Each L-output is then converted back into a serial format by means of a parallel-to-serial converter.

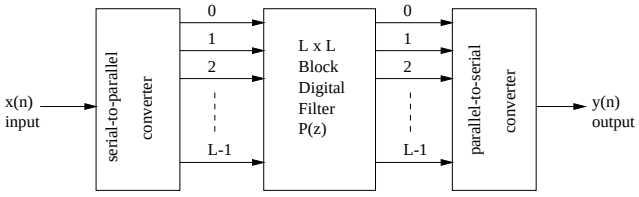

Figure 1: Conventional block digital filter

Another advantage of block processing is the possibility of using parallel structures to reduce the computational complexity and computation time of digital filtering systems. Lin and Mitra [1] have recently developed the block processing considering overlapped input and/or output blocks which can be implemented using parallel structures. Figure 2 illustrates the representation of overlapped block digital filtering, where L represents the input block size, N represents the output block size, and M is the down-sampling (up-sampling) factor. When $L = M$, the input blocks are not overlapped, and when $L > M$, the input blocks are overlapped. Likewise, when $N = M$, the output block are not overlapped, and when $N > M$, the output blocks are overlapped.

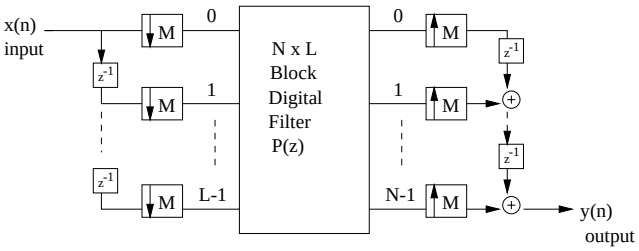

Figure 2: Multirate representation of an overlapped block digital filter

In this paper, we propose a parallel implementation for overlapped block digital filtering using *Scheduling by Edge Reversal* (SER) [2] [3]. SER is a powerful parallel and distributed graph-based algorithm developed for controlling concurrent operation amongst the elements of neighborhood-constrained systems of generic topologies.

The next section reviews the basic concepts of parallel block digital filtering. Section 3 introduces the *Scheduling by Edge Reversal* (SER) technique. Section 4 describes the proposed parallel implementation and results. The last section includes conclusions and suggestions for future works.

0-7803-5482-6/99/$10.00 ©2000 IEEE

2. PARALLEL BLOCK DIGITAL FILTERING

Lin and Mitra [1] have developed a parallel structure for implementing FIR filters using the overlapped block digital filtering. Using this framework, two fast FIR filtering algorithms were derived.

The first algorithm is based on a structure called Type A. In this structure the input blocks are not overlapped, whereas, the output blocks are overlapped. We can choose an input block size of $L = M$, and an output block size of $N = 2M - 1$. The block transfer matrix, $P_1(z)$, is represented below.

$$P_1(z) = \begin{bmatrix} H_{M-1}(z) & 0 & \cdots & 0 \\ H_{M-2}(z) & H_{M-1}(z) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ H_0(z) & H_1(z) & \cdots & H_{M-1}(z) \\ 0 & H_0(z) & \cdots & H_{M-2}(z) \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & H_0(z) \end{bmatrix} \quad (1)$$

The second algorithm is based on a structure called Type S. For a Type S structure the input blocks are overlapped, whereas the output blocks are not overlapped. We can choose a structure with $N = M$, $L = 2M - 1$. The block transfer function matrix, $P_2(z)$, is shown below.

$$P_2(z) = \begin{bmatrix} H_0(z) & H_1(z) & \cdots & H_{M-1}(z) & 0 & \cdots & 0 \\ 0 & H_0(z) & \cdots & H_{M-2}(z) & H_{M-1}(z) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & H_0(z) & H_1(z) & \cdots & H_{M-1}(z) \end{bmatrix} \quad (2)$$

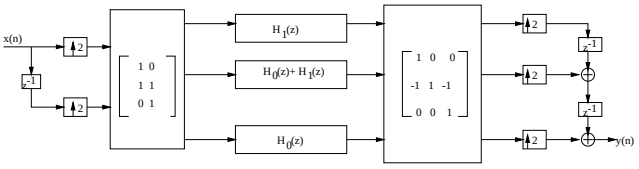

Figure 3: Example of Type A structure

Figure 3 shows a Type A structure for the case where $L = M = 2$, and $N = 3$. This structure was derived by decomposing matrix $P_1(z)$ as follows

$$\begin{bmatrix} H_1(z) & 0 \\ H_0(z) & H_1(z) \\ 0 & H_0(z) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & -1 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} H_1(z) & 0 & 0 \\ 0 & H_0(z)+H_1(z) & 0 \\ 0 & 0 & H_0(z) \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{bmatrix} \quad (3)$$

Each polynomial $H_i(z)$, for $i = 0, 1$, in (3) corresponds to a filtering operation with an FIR filter of about half the length of the original filter $H(z)$, being its polyphase components.

Figure 4 shows a Type S structure. This structure is exactly the transpose of Type A overlapped block structure. In both structures the number of subfiltering operations has been reduced from 4 to 3, reducing the computational complexity.

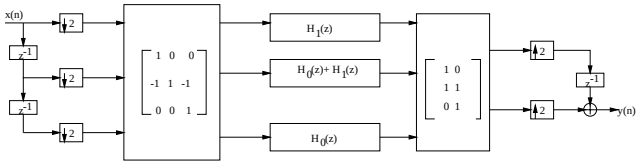

Figure 4: Example Type S structure

By using Type A structure presented, we will develop an parallel implementation for FIR filter using SER.

3. SCHEDULING BY EDGE REVERSAL -SER

SER [2] [3] is based on the idea of a population of processes executing upon access to shared atomic resources. Shared-resources systems are represented as a finite oriented graph $G = (N, E)$ where processes are the nodes of G and an oriented edge exists between any two nodes whenever they share a resource. A process is allowed to operate if it has all its edges directed to itself, a sink in G.

SER works starting from any acyclic orientation w on G. This means that there is at least one sink node. Consider, just for the purpose of this explanation, an ideal synchronous environment where according to the beginning of a clock pulse, only sinks are allowed to operate while other nodes remain idle. After operating and before the clock pulse ends, sinks reverse the orientation of their edges by sending messages to all their neighbors, each one becoming a source. It is easy to see that, again, another acyclic orientation, w', is formed and a new set of sinks can operate

at the next clock pulse. All subsequent orientations are also acyclic and the scheduling mechanism consists basically of consecutive sets of sinks being defined in G through time.

Some fundamental SER properties will now be presented.

Lemma 1 *Let ω and ω' be acyclic orientations of G such that ω'=g(ω). A sink in ω' has at least one neighbor that is a sink in ω [3].*

Let $m_i(q)$ define the number of times node $I \in N$ operates in q consecutive acyclic orientations under SER. The following theorem states a strong form of starvation freedom.

Theorem 1 *Consider nodes $i,j \in N$, and let the shortest path connecting them in G have r edges. Then $|m_i(q) - m_i(j)| \leq r$ for all $q \geq l$ [3].*

Theorem 1 shows that this simple dynamics is, consequently, free of deadlocks or starvation since there will be always at least one sink node and since the maximum amount taken for a node to become a sink is directly related to the longest directed path from it to a sink. Assuming that G is finite, eventually a set of acyclic orientations will be repeated. Derived from Theorem 1, SER's fairness is a very interesting property presented in the corollary below.

Corollary 1 *The number of times that a node becomes a sink in a period is m, the same for all nodes in G [3].*

As the number of such acyclic orientations is finite, eventually a set of orientations will repeat. This defines a period of length p of orientations. It is proved that inside any period every node operates exactly m times. The concurrency of a period is defined by the coefficient m/p. It is clear that the higher m/p is, less idle are every node through each period.

SER is a fully distributed algorithm. Its operation, after the initial reset, does not depend on any global signaling in order to work, as the decision that a node start or finish operating can be fully local. Another interesting property of this graph dynamics lies on its generality in the sense that any particular topology will have its own set of possible SER dynamics.

Figure 5 presents the equivalent graph G for Type A structure shown in Figure 3, where the node 1 is in black to represent one sink. Each node corresponds to one processor. Node 1 is responsible for decimation and creation of input vectors to sub-filters $H_1(z)$, $H_0(z) + H_1(z)$ and $H_0(z)$. These filters are represented by nodes 2, 3 and 4, respectively. Node 5 is responsible to receive the outputs of each sub-filter, interpolate them and create the output vector. Figure 6 shows us the graph G under SER.

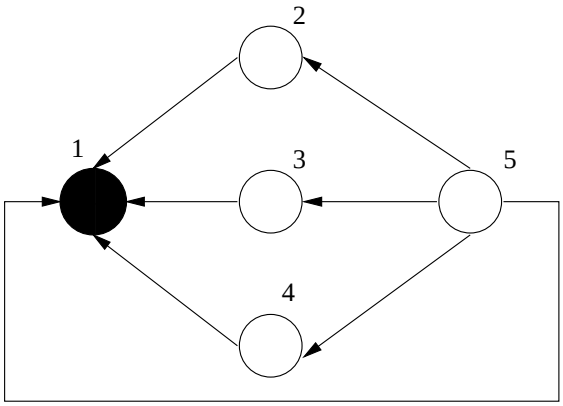

Figure 5: Graph G for the Type A structure

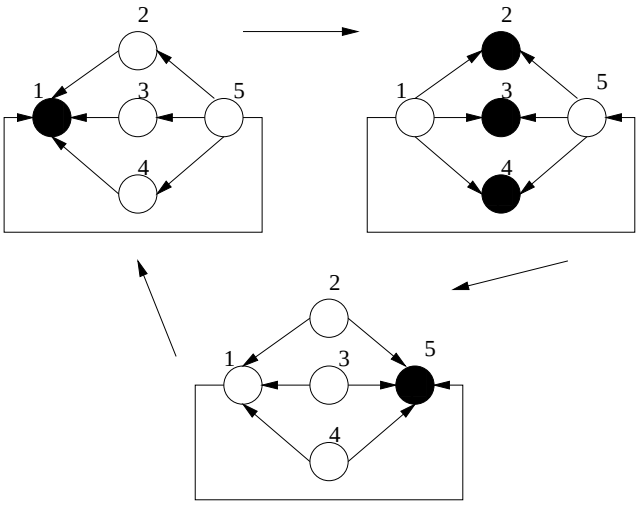

Figure 6: Graph G under SER, with $m = 1$ and $p = 3$, i.e., each node is a sink (in black) once in three synchronous steps

4. PARALLEL IMPLEMENTATION AND RESULTS

The structure presented in figure 3 was implemented on a parallel computer platform, the TN-310 system [4]. This is a 16 node Transputer/digital signal processor (DSP) based parallel machine with distributed memory. The machine allows all nodes to communicate to each other and, consequently, each node can access data held anywhere in the system (see figure 7).

For task involving signal processing, the TN-310 system has a fast and powerful DSP (ADSP-21020) coupled to each processing node. The DSP acts as a coprocessor to the T9000 (Transputer) and can be programmed from the T9000 through C runtime library calls. The TN-310 system can be accessed through a host machine, which, in our case,

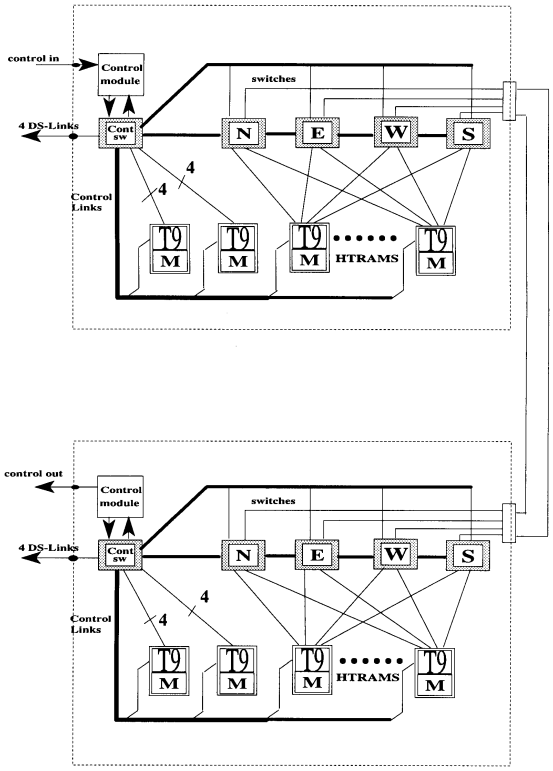

Figure 7: The architecture of the TN-310 system and the interconnection of the two boards of processors. Extracted from [4] .

is an IBM PC.

Developing an application in this environment comprises two phases. In the first, the user configures the available hardware according to the target application. This phase describes the parallel application in terms of number of processors, amount of memory for each processor, interconnection matrix of processors, the use of cache memory and control signal. In the second phase, the user develops high-level codes (in C language) according to the resources available from the hardware configuration of the first phase.

In this TN-310 environment we implemented Type A structure to FIR filters using SER (figure 5). Table 4 shows us the computation time of the parallel structure as compared with computation time of the direct form implementation of the same transfer function. It lists the overall computer time needed to calculate 10 000 output samples for two filters of different lengths.

5. CONCLUSIONS

The implementation parallel overlapped block filtering using SER was developed. The structure presented in Figure 5 was implemented in the TN-310 system. The computation-time ratio obtained between direct form and overlapped block implementation using SER was about 0.3, therefore the total computation time was significantly reduced.

Table 1: Comparison of Computer Times: M=2 Case

filter order	30	100
overlapped block	271804	597243
direct form	792830	1901807
speed-up	2.9	3.1

For type A structure presented in this paper, theoretically, the ratio between the computation times of the fast algorithm and the direct implementation should approach 75% when the filter length increases, because the number of sub-filtering operations have been reduced from 4 to 3. However, using SER in the structure with 5 processors, we obtained the ratio of approximately 30% which is better than the expected computation-time ratio. This proves that SER is very powerful in parallel implementation applications such as the one presented in this paper.

Even though SER has been applied to a particular implementation, it can be used in many other applications in which it is possible to employ a parallel structure. SER, for example, can be used in the implementation of parallel adaptive filters [5]-[6].

6. REFERENCES

[1] LIN, I. S., MITRA, S. K., "Overlapped Block Digital Filtering", *IEEE Trans. Circuits Syst*, v. 43, n. 8, Aug 1997.

[2] BARBOSA, V. C., GAFNI, E., "Concurrency in heavily loaded neighbourhood-nontrained systems", *ACM Transactions on Prog. Languages and Systems*, v. 11, n. 4, Oct 1996.

[3] BARBOSA, V. C., *An Introduction to Distributed Algorithms*. MIT Press, 1996.

[4] TELMAT MULTINODE, *TN 310 System Manual and Training Set*. France, 1995.

[5] DINIZ, P. S. R., *Adaptive Filtering: Algorithms and Practical Implementation*. Norwell, MA, Kluwer Academic Publishers, 1997.

[6] R. MERCHED, P. S. R. D., PETRAGLIA, M. R., "A New Delayless Subband Adaptive Filter Structure", *IEEE Trans. on Signal Processing*, v. 47, pp. 1580–1591, Jun 1999.