

Fault-Tolerant Modular Convolvers

Luigi DADDA, Vincenzo PIURI

Politecnico di Milano, Dipartimento Elettronica e Informazione

P.za Leonardo da Vinci 32, I-20133 Milano, Italy

tel. +39 2 2399.3510; fax +39 2 2399.3411; email: dadda,piuri@ipmel2.elet.polimi.it

Abstract.

A preceding research, concerning the architecture of a convolver capable of accepting a sequence of groups of p samples presented simultaneously and of generating at p output ports the corresponding sequence of convolution groups, is expanded with a new approach. In this architecture the convolver is decomposed into a linear array of p identical "phase-convolvers" associated with an identical input networks, each producing one convolution every p . The resulting architecture has high fault tolerant features at different structural levels (the phase-convolvers, the sub-convolvers composing them, the multiplier-adders). Different strategies and alternatives are discussed to introduce different degrees of fault tolerance, suitable for different production and operational environments.

1. Introduction

In a number of mission-critical applications belonging to wide areas of digital signal/image processing, the use of convolvers having outstanding characteristics of regularity, modularity, performances and fault tolerance, becomes nowadays mandatory to allow an effective and efficient VLSI/WSI realization of complex systems.

In this paper, we consider the problem of architectural design of such units: our goal is the identification of a simple, modular approach to create regular structures capable of surviving to faults through reconfiguration and to achieve massive computational capabilities by partitioning the computation into parallel operations onto disjoint groups of input data. Dynamic adaptation of the basic structure is also considered to provide functional reconfigurability.

A *polyphase* network was firstly introduced in [1, 2]: it was defined as a set of all-pass filters having different phase characteristics (hence *polyphase*), followed by a summation device. The filtering process is obtained by amplitude addition in the pass-band and by amplitude cancellation in the stop band.

A similar structure was proposed in [3] for a generic serial-input FIR filter in order to obtain a null interval between subsequent samples: it is composed by a set of convolvers which are fed cyclically; no final summation is required. The intrinsic behavior of this structure is therefore inherently different from the one proposed in [1, 2]. A *p-phase convolver* is defined as a circuit generating one convolution every p input samples; a standard convolver is obtained by means of p *p-phase* convolvers properly interconnected. A first approach was based on a systematic analysis of the sample-weight products needed in each step for the computation of the convolutions to be generated: by using a suitably organized table, it was possible to determine the architecture of the whole convolver. A second approach was an algebraic one, proving among other things the formal correctness of the results obtained heuristically through the previous approach. The architecture obtained consisted in a network of phase convolvers, apparently, without any regular structure.

In this paper we propose a new approach to design multi-parallel polyphase convolvers which leads to a highly regular structure. Section II is directed to introduce the architecture

starting from the definition of the nominal operations. The obtained architecture is particularly suited to support efficiently several defect and fault tolerance techniques at different levels by providing spare sub-units which can easily replace the faulty ones. Section III will discuss defect and fault tolerance strategies. The same architectural supports for fault tolerance through dynamic redundancy provide also functional reconfigurability to allow adaptation of the same device to different operating conditions or application constraints, as discussed in section IV.

2. The basic scheme

To introduce our approach, we analyze first the convolution algorithm by considering the time diagram presented in fig 1 for the case of a 2-input convolver. It shows the data presentation at the convolver inputs for the sequence of time steps, with the corresponding convolution weights.

On the top of fig.1a, we represent the sequence of X_j samples grouped into 2-sample groups; in each group, samples are transmitted in parallel to the convolver. A time window composed of three adjacent groups (sub-windows) is shown at times $t-2$, $t-1$, t . The window contains the weights W_i ($i=0,1,2,\dots,5$) in the positions shown in the first row, and it slides in steps of two samples (i.e., of a sub-window) at each time step.

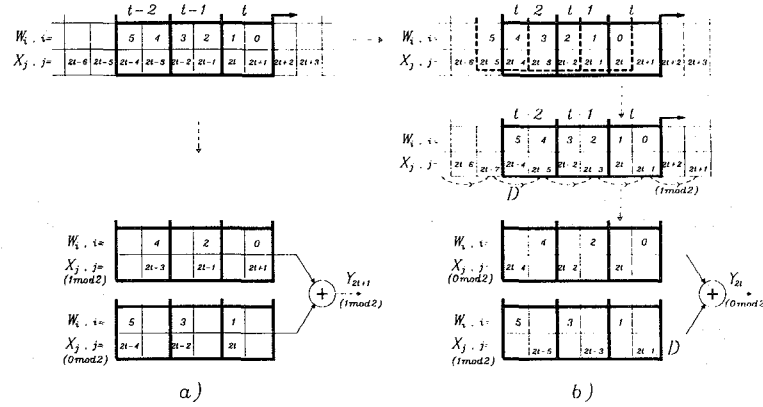


Figure 1 The operation of a 2-input convolver: a) for the generation of odd-indexed convolutions; b) for the generation of the even-indexed convolutions.

Convolution Y_{2t+1} (corresponding to sample X_{2t+1} - the rightmost in the outlined windows) is obtained by performing the weight-sample inner product, i.e. by executing the inner product of the weight vector multiplied by the sample vector of data contained in the considered window. The inner product can be decomposed into the sum of two sub-inner products (or sub-convolutions), as shown at the bottom of fig. 1a. The sub-convolvers are standard (i.e. 1-input) convolvers, since their input samples are ordered in time so that one sample at each time step (i.e., in each time window) is presented to each sub-convolver. The two standard sub-convolvers are thus characterized by even- and odd-indexed weights respectively. The outputs generated in a set of time steps corresponding to the kernel of the convolution (in our example, three steps) are then added to produce the odd-indexed convolutions Y_{2t+1} . This convolver is therefore a 2-phase convolver.

Generation of the even indexed convolutions Y_{2t} is presented in fig. 1b. Consider first the dotted window on the top of fig. 1b. It includes all samples needed to compute Y_{2t} , but it

cannot be directly implemented since samples required in each outlined window belong to two different dotted windows, i.e., they are available in two different time steps instead of within the same time step (as required by the convolution algorithm for the multi-parallel case).

However, we can implement such a sliding window if the original sample sequence is modified as shown in the middle of fig.1b, by shifting all even indexed samples two positions to the right, i.e. by delaying them into the subsequent sub-window. This can be obtained by a delay D as shown in the figure, so that all samples needed to compute even indexed convolutions Y_{2t} are included in the same time window.

Partitioning of these samples into two 1-parallel groups is shown at the bottom of fig. 1b, where each sample is associated with the proper weight.

Extension of our approach to the case of a 3-input convolver is shown in fig. 2. In fig. 2a we present the convolver structure for Y_{3t+2} , while in fig. 2b the use of a delay D allows to guarantee the suited alignment of samples in the sliding window. Finally, in fig. 2c two delays are necessary to achieve the expected positioning of data in the window. Generalization to a higher number of inputs is then straightforward.

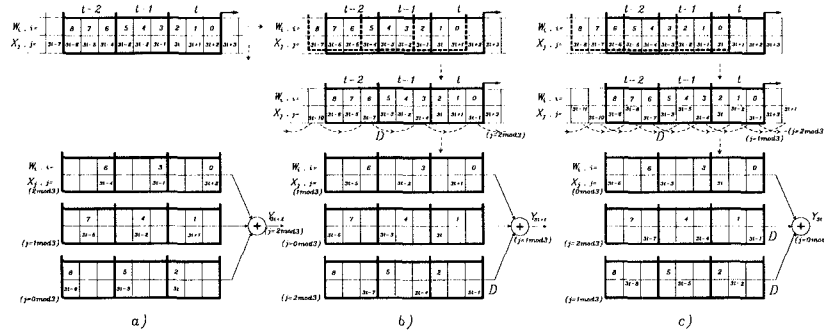


Figure 2 The operation of a 3-input convolver: a), b), c), for the generation of convolutions indexed $2\text{mod}3$, $1\text{mod}3$, $0\text{mod}3$ respectively.

The global operation of the multi-parallel convolvers and the overall architectures are shown in fig. 3, for the case of 2-input and 3-input convolvers. Inputs are delivered in parallel to all sub-convolvers: each of them is associated with a specific phase and generates the corresponding output at the proper time steps. All output results are thus skewed in time according to their phase: the expected sequence of output convolutions can be created by collecting the sub-convolvers' outputs at the corresponding time steps.

The general schemes of the resulting architectures are shown in fig. 4, for the case of a 2-input convolver (fig. 4a), for the case of a 3-input convolver (fig. 4b), and for the case of a 4-input convolver (fig. 4c). The high regularity of the structures can be further enhanced by rearranging these schemes, which directly implement the previous algorithmic analysis, into the schemes presented in fig. 5 for the same cases. Application-specific design can be easily derived by identifying the number of phases which is the most suited to provide the required timing performances. In a p -phase convolver, the throughput is equal to p convolutions for each clock cycle; on the other hand, the clock cycle must be at least equal to the time τ_{cs} required to complete one convolution step. If convolution is performed on k samples, the convolution latency is $\tau_{cs} (1 + \text{ceiling}(k/n))$, by assuming that the time for adding partial convolutions is not higher than τ_{cs} .

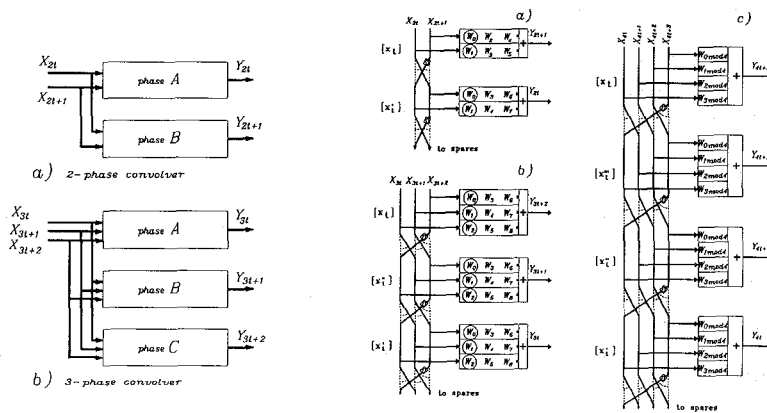


Figure 3 a) A 2-input convolver composed with two 2-phase convolvers; b) A 3-input convolver composed with three 3-phase convolvers.

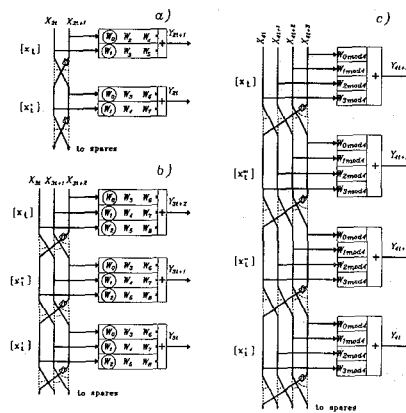


Figure 4 a) A scheme for a 2-input convolver; b) A scheme for a 3-input convolver; c) A scheme for a 4-input convolver.

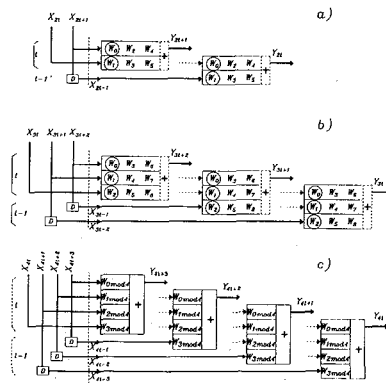


Figure 5 Schemes equivalent to fig. 4 schemes, showing their linear array structure with switches for fault confinement.

3. Defect and fault tolerance

The high regularity of the architecture presented in the previous section is well suited to support strategies for defect tolerance, in order to increase the fabrication yield in VLSI/WSI implementations. On the other hand, our scheme has been studied also in relation to its possible use for mission-critical applications when very high fault-tolerance performances are mandatory or when maintenance is difficult or impossible; for example, in radiation-hard environments or in aerospace applications. Our approach has characteristics which are well suited for such applications and for traditional signal processing applications.

Consider first the problem of defect tolerance in an integrated implementation. To achieve system survival in the presence of production defects, it is necessary to introduce the capability of excluding the defective units from the active computation. The basic solution consists of hardware configuration of the interconnections between units [4]. This goal can be achieved by segmenting the interconnection paths and introducing configurable interconnections between segments. Permanent configuration may be realized through laser cutting, electrical programming of connecting fuses, or interconnection deposition. Dynamic configuration through programmable switches allows for a more flexible approach and for possible subsequent fault tolerance.

Application of these techniques implies the availability of a suited number of spare components which can be used to replace the defective ones. Configuration is successful if a fault-free convolver can be created by connecting fault-free components, according with the interconnection structure and the configuration approach.

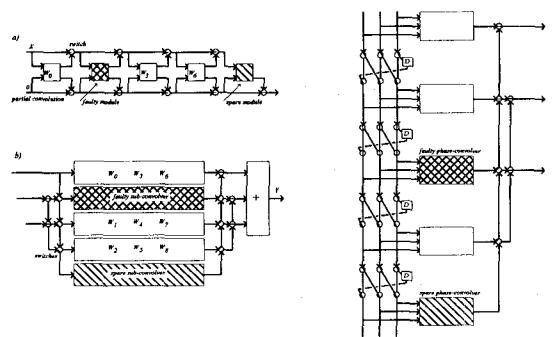


Figure 6 Schemes for defect and fault tolerance: at the component level (a), at the sub-convolver level (b), and at the phase-convolver level (c).

Three levels of defect confinement may be considered by referring to three different architectural levels; in fig. 6, an example is given for the case of switched-bus interconnections (as usual [4], bus and switches are assumed fault free).

- ♦ The *first* approach is concerned with the arithmetic units (namely, the multiplier and the adder which generates the partial convolution) within each sub-convolver (see fig. 6a). Spare multipliers are introduced in each sub-convolver: if a multiplier or an adder in the module is defective, it is excluded from the computation by configuring the interconnections within the sub-convolver so that the defective module is by-passed (e.g., by using the Diogene's approach [4]). Unused spares remain disconnected from the active computation. A sub-convolver is successfully configured if there are enough fault-free modules that can be connected; otherwise, the whole sub-convolver must be considered defective. Weights are stored in the active components of the sub-convolver to match the nominal algorithm.
- ♦ At the *second* level, we consider the sub-convolvers within each phase-convolver (see fig. 6b). A defective sub-convolver can be confined by excluding its output from the final addition which generates the convolver output. This goal can be achieved by-passing the output of the defective sub-convolver by means of a switched interconnection structure at the outputs of the sub-convolvers within the phase-convolver, before the final adder generating the corresponding phase output. Replacement of the defective sub-convolvers requires spare sub-convolvers within the phase-convolver; unused spare sub-convolvers remain unconnected. A phase-convolver is successfully configured if there are enough fault-free sub-convolvers within the phase-convolver to match the nominal algorithm; otherwise,

the whole phase-convolver must be considered defective. Weights are stored into the fault-free sub-convolvers to perform the nominal computation. A defect in the final adder of the phase-convolver is critical since cannot be tolerated by any configuration strategy at this second level.

- ♦ The *third* level operates at the level of phase-convolvers (see fig. 6c). Confinement requires exclusion of the defective phase convolver and activation of a spare one. Configuration is successful if there are enough spare phase-convolvers to tolerate the defective ones. Exclusion of a fault phase-convolver is performed by configuring the interconnections between phase-convolvers so that inputs are propagated to the subsequent phase-convolver without position permutation; this operation may be performed by moving the permutation switches corresponding to the faulty phase convolver in the main bus to the dotted position. Outputs of the defective phase-convolver must be ignored so that erroneous results are not inserted into the global sequence of convolutions generated by the whole convolver. Since all phase-convolvers are identical and weights sets coincide, no specific action must be taken to provide the correct weight distribution on the whole architecture.

To increase the probability of successful configuration, a globally-configurable architecture could be derived by using an array of identical sub-convolvers and a suited interconnecting network for convolver's input presentation and for extraction of partial convolutions; the final convolutions could then be generated by an array of p -inputs adders. The switched input bus could provide the configuration ability for defect tolerance, i.e., for excluding the defective sub-convolvers: identical sub-convolvers are logically grouped (p sub-convolvers at a time) into a phase-convolver by positioning the interconnection switches. Another switched bus could route the partial convolutions generated by sub-convolvers, which belong to the same phase-convolver, to the corresponding p -inputs adder; this allows for excluding results generated by defective sub-convolvers and for by-passing the defective p -inputs adders. Finally, a switched output bus could exclude the defective p -inputs adders by connecting the active p -inputs adders to the p nominal convolver outputs.

This architecture guarantees a higher degree of survival since spare sub-convolvers are shared among all phase-convolvers, so that a higher number of defect distributions can be tolerated. Moreover, also defects in the final adders can be tolerated when enough spares are provided. On the other hand, the presence of switched busses could induce a larger hard core and must be carefully evaluated with respect to the expected defect tolerance capabilities.

The choice among the previous approaches depends on several factors, in particular on the expected defect distribution and on the desired yield. Besides, feasibility of some solutions is also related to the characteristics of the implementation adopted. For example, all solutions can be realized if weights can be programmed into the configured architecture (e.g., if they are stored in registers or in PROMs); conversely, if weights are fixed in ROMs or if arithmetic devices are tailored for such values, only the phase-convolver level can be adopted since no weight modification is allowed within the individual module or sub-convolver.

Note that no specific testing support has been included in the above architectures, even if some hardware redundancy required for defect tolerance can be exploited also for testing (e.g., the switched busses may be used to introduce additional test points).

Consider now the problem of fault tolerance. The basic step of any fault-tolerance strategy is detection of the errors induced by faults, since we operate at functional level to observe the behavior of the architecture. Then, location of the faulty unit is required to provide system survival through interconnection reconfiguration.

Different approaches can be adopted to achieve error detection at different system levels. Duplication with output comparison is simple in principle, but hardware redundancy is very high. Better solutions are based on the use of arithmetic codes [5] for concurrent error

detection: due to the characteristics of the nominal computation, AN codes [6] and residue codes [7] are particularly suited for such application.

In AN codes [6], nominal inputs are coded by a linear transformation; convolution operations are performed in the coded space and outputs are checked for correct divisibility. An error is detected whenever results are not divisible by the code generator A (A^2 must be considered if both operands are coded). Error checking can be performed:

- ♦ at the output of each arithmetic unit,
- ♦ on each partial sub-convolution, i.e., at the output of the two-input adder which adds the weighted input to the partial sub-convolution of the previous stage,
- ♦ at the output of each sub-convolver,
- ♦ at the output of each phase-convolver.

These different abstraction levels provide possibly-decreasing protection and decreasing localization capability at decreasing hardware costs. Possible error masking may in fact occur at the higher abstraction levels in the presence of multiple faults. Fault localization is high at the lowest levels; however, checking each arithmetic unit is generally not convenient since no reconfiguration capability can reasonably be considered for the single units.

Hardware redundancy and computational delay are related to the checkers and to the increased size of the arithmetic units (due to the larger size of the coded operands). For single errors, the best results are achieved by using the 3N code.

To reduce the hardware overhead and the computational delay, we can use the residue codes [7] to represent the nominal operands. In this case, the codeword is obtained by cascading the residue of the nominal operand with respect to a given base to the nominal operand itself. Nominal operations are performed on the nominal operands, while module operations (corresponding to the nominal ones) are applied to the residues. Nominal results are then coded and compared to the results generated by the module operations: an error is detected whenever such residue are not equal. Also in this case, we can adopt different abstraction approaches, as in the AN codes. For the single error case, the best results are achieved by using the base 3.

System survival can be achieved by confining the faulty components through interconnection reconfiguration. Three approaches may be considered by referring to the same three architectural levels analyzed for defect tolerance.

The overall characteristics of these techniques coincides with the one discussed above. Weights must be redistributed within the new configuration to match the nominal algorithm according with the considered abstraction level of the reconfiguration policy. In the arithmetic unit case, the weights must be moved within the sub-convolver, while in the case of sub-convolver reconfiguration, all weights of the faulty sub-convolver must be transferred into the spare one; no weight update is required for the third level case since all phase-convolvers are identical. Similar updates are required also for the case of the globally-configurable architecture.

While concurrent localization of the faulty component can be achieved by exploiting the information generated by concurrent error detecting circuits, reconfiguration of the interconnection can be driven by an external host to minimize the hardware overhead and the hard core of the system. However, due to the linear structure of each component (namely, each sub-convolver is a linear array of modules, each phase-convolver is a linear array of sub-convolvers, and each convolver is a linear array of phase-convolvers), also on-board reconfiguration can be envisioned. At each abstraction level, the error signal generated by the corresponding component can be used to drive the reconfiguration elements (i.e., the switches) for such a level.

The choice of the most suited approach depends on several factors, in particular on the expected fault rate and on the expected system life. However, we must take into account also

the architectural costs in terms of hardware redundancy and computational delay to match the application specifications.

Feasibility of some solutions is also related to the characteristics of the implementation adopted. All solutions can be realized if weights can be updated (e.g., if they are stored in registers or in EPROMs); conversely, if weights are fixed in ROMs or if a custom design has been adopted for arithmetic devices, only the phase-convolver level is feasible since no weight modification is allowed within the individual module or sub-convolver.

In cases in which the convolver operates in the most severe environments, where a high level of protection is required, fault correction capabilities by fault confinement through reconfiguration can be improved by adopting more than one spare unit and by using more than one level of fault correction. The proposed architectures for the p -input convolvers are well suited to support such applications.

4. Functional adaptability

Modularity and regularity of the architectures presented above can be also exploited to support functional adaptability of the convolver. In particular, the configurable structure of the sub-convolvers, which is required for defect tolerance, can be directly used to customize the length the sub-convolvers (i.e., the number of samples considered in each convolution window) after production according with the specific implementation adopted. For example, laser cutting can be used both to exclude the defective modules in a sub-convolver and to define the length of the sub-convolver itself by cutting off also the remaining spare modules. In the case of switched connections, switches are programmed to include only the desired number of fault-free modules in the sub-convolver; the remaining modules are treated as faulty modules. Configuration is successful only if enough fault-free modules are available within each sub-convolver as they are required by the target application.

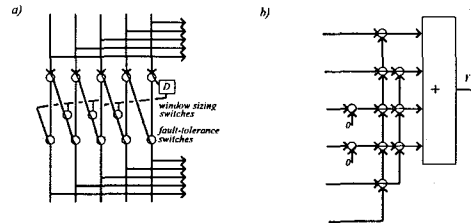


Figure 7 Schemes for functional adaptability: the input bus (a), the final adder (b).

Functional adaptability is also available in the architectures designed to support fault tolerance during the life time of the convolver; the switched busses are exploited as in the case of defect tolerance. This approach can be used to design and implement classes of multi-parallel convolvers, characterized by the parallelism degree p . The convolver with the most suited value of p is then tailored on the specific application or on the specific operating conditions.

Functional adaptability at higher abstraction levels is related to the capability of adjusting the size of the convolution window. To introduce this feature, we have to deal with the structure of the bus providing the proper ordering of the input samples, and with the final adders generating the phase-convolver outputs. Adaptability of the bus can be achieved by modifying the interconnecting switches adopted for fault tolerance, as it is shown in fig. 7a. The maximum size of the window is fixed by the total number of samples that are propagated on such bus, while the actual size is given by the actual switch configuration which capture the rotated input from the rightmost side.

Adaptability of the final adders is more difficult. A solution is given in fig. 7b: the number of inputs is always equal to the maximum value, but undesired inputs are set to zero by the modified interconnection switches required to provide fault tolerance.

Functional adaptability can be used to introduce additional defect/fault tolerance capabilities by means of *functional graceful degradation* of the nominal behavior of the convolver. When there are not enough modules to configure all fault-free sub-convolvers required by the application, we may create a sub-convolver with reduced computational capabilities by reducing the size of the convolved samples' window. While in the basic approach discussed in section II the system is declared faulty, in this case we grant system survival even if performances are reduced with respect to the nominal ones. When further faults occur, we can apply the same approach by gracefully degrading the performances of the convolver. The system will be assumed faulty when the computational capability becomes lower than the minimum value that can be accepted for the specific application.

5. Concluding remarks

In this paper a novel approach to design multi-parallel convolvers has been proposed: the main goal was to achieve a highly-regular architecture which is well suited for integrated implementation and for defect/fault tolerance. The structure of the multi-parallel polyphase convolver has been directly derived by properly delaying the presentation of the input samples within the time windows on which the convolver components operate. Then, the regular structure has been created by rearranging the topology of the overall computational scheme. On the final architecture, the detailed analysis of the defect/fault tolerance characteristics has been performed and system survival has been achieved by using spare components and a suited reconfigurable interconnection structure. Functional adaptability has also been discussed by considering dynamic reconfiguration of the interconnections. Detailed evaluations of the hardware costs, of timing performances, and of survival capabilities will be given in the final paper.

6. Acknowledgement

We wish to thank Vijay K. Jain for suggesting the similarity with the polyphase network introduced by Bellanger, Bonnerot and Coudreuse.

7. References

- [1] Bellanger M.G., Bonnerot G., Coudreuse M., "Digital Filtering by Polyphase Network: Application to Sample-Rate Alteration and Filtering Banks", *IEEE Trans. on Acoustics, Speech, and Signal Processing*, vol. ASSP-24, no. 2, pp. 109-114, April 1976
- [2] Crochiere R.F., Rabiner L.R., *Multirate Digital Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1983
- [3] Dadda L., "A Polyphase Architecture for Serial-Input Convolvers", *Journal of VLSI Signal Processing*, vol. 2, 1990, pp. 17-27
- [4] Negrini R., Sami M.G., Stefanelli R., *Fault-Tolerance Through Reconfiguration of VLSI and WSI Arrays*, MIT Press, 1989
- [5] Rao T.R.N., *Error Coding for Arithmetic Processors*, Academic Press, NY, 1974
- [6] Piuri V., "Fault-tolerant array processors: an approach based upon AN codes", *IEEE Proc. Int. Conf. on Circuits and Systems (ISCAS88)*, Helsinki, Finland, June 1988.
- [7] Piuri V., "Fault-Tolerant Systolic Arrays: an Approach Based upon Residue Arithmetic", *IEEE Proc. Int. Symp. on Computer Arithmetic (ARITH8)*, Como, Italy, May 1987, pp. 230-238.