A METHODOLOGY FOR ASIC IMPLEMENTATION OF DIGITAL FILTERS

U. Sjöström, I. Defilippis, M. Ansorge, F. Pellandini

Institut de Microtechnique, Université de Neuchâtel, Rue A.-L. Breguet 2, CH-2000 NEUCHATEL, Switzerland

RESUME

Les remarquables progrès rencontrés dans les technologies VLSI ainsi que le nombre croissant des domaines d'application faisant appel aux signaux numériques ont créé des besoins pour de nouvelles méthodes de conception de *circuits intégrés spécifiques* (ASIC) dans le domaine du *traitement numérique du signal*. Cet article présente une nouvelle méthodologie sous la forme d'un système de conception à répartition verticale, avec un choix approprié des classes d'algorithmes de filtrage, un développement d'architectures spécifiques et des règles de traduction générales, simples et efficaces permettant de passer des algorithmes à leur implantation. Divers outils de *Conception Assistée à l'Ordinateur* (CAO) ont également été développés dans le cadre de ce projet.

SUMMARY

In recent years the great advance in VLSI technologies and the increasing number of application fields using digital signals have created a need for new methods for developing *Application Specific Integrated Circuits* (ASICs) in the field of *Digital Signal Processing* (DSP). In this paper a novel method, in form of a *vertically sliced synthesis system*, treating these tasks is presented. The emphasis of the methodology has been put on finding suitable filter algorithm classes, on developing a special purpose architecture, on establishing a simple, general and efficient translation of the filter algorithm into terms of hardware. Furthermore, some *Computer Aided Design* (CAD) tools have been developed during the project.

I. Introduction

Digital Signal Processing (DSP) presents many interesting advantages compared to analog processing. Problems such as tuning, ageing and temperature drift are totally avoided in the digital domain. A wide range of structures and algorithms are available, some with a high amount of flexibility.

DSP algorithms can be implemented using ASICs, which offer high density integration and possibility to realize large and complex filtering systems on a small number of chips. Furthermore, existing systems can evolve without complete system redesign, and ASIC designs can be well protected. All these aspects added to the advantages of digital signal processing render the field very attractive.

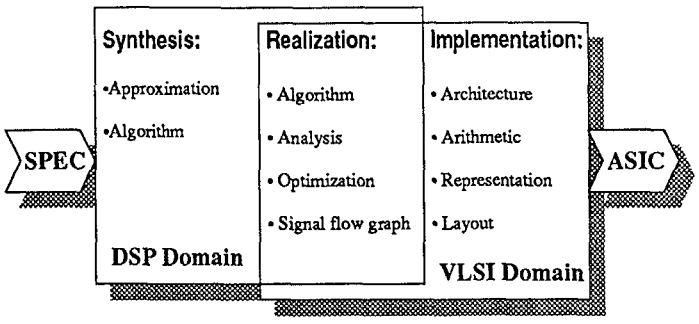

Figure 1 *The design task for digital filters*

Many different design strategies and methodologies can be found in this domain. Where these strategies differ most is probably on the target architecture and the mapping from the DSP algorithm. Some strategies described in LAGER [Raba85] and CATHEDRAL II [DeMa87] are based on microprocessor architectures. Other ones, MOVAL[Ligt86], FIRST [Deny85], CATHEDRAL I [DeMa87], are based on the *data-flow* concept.

A good design strategy is needed to reduce the design time, which otherwise can be prohibitive. It is essential to have a deep understanding of the design process of the DSP and VLSI fields, in order to be able to define a good methodology.

II. Design and Implementation Task

The task can be divided into three steps namely, *synthesis, realization and implementation*. First, some requirements and specifications given from the application should be fulfilled. The synthesis step consists of using a proper approximation algorithm to find a pole-zero configuration for a filter meeting these requirements. In the realization step, an explicit description of the filter algorithm is developed, i.e., a *signal flow graph* (SFG). Furthermore, optimization of word-lengths and certain properties are made in this step. This process is very interdependent on the final implementation step of the filter.

Fig. 1 shows an overview of the design process. The realization step can be considered to be a part of both the DSP domain and the VLSI domain. The filter synthesis step has been developed over the last decades and can thus be considered as well established. The same can be assumed for the VLSI implementation domain. For these two steps the main point is to select feasible parts of existing strategies. For the realization step, however, new methodologies can be defined, since this step serves as a bridge between the DSP and the VLSI domains. A strict definition of the final filter algorithm description must be made, as well as a definition of the real implementation cost in terms of this description.

III. DSP Algorithms

Constraints must be introduced to limit the complexity of the methodology. There are some fundamental differences between adaptive and fixed filter algorithms, both when synthesizing them and implementing them. The methodology described here is basically limited to *Linear and Time Invariant* (LTI) digital filters. Only weakly programmable filters, i.e. filters with a finite set of predefined coefficients, are considered.

Classical digital filters can be divided into two main groups. First of all filters with *Finite duration Impulse Response* (FIR). Secondly *Infinite duration Impulse Response* (IIR) filters. They have their own specific properties [Oppe75]. For example, IIR filters can usually be realized more efficiently than FIR filters. On the other hand, stability is often a problem due to the recursive nature of IIR algorithms and due to the finite representation of numbers. For this reason they are very cumbersome to use in certain applications.

Fortunately, one class of IIR filters provides a solution for these problems, namely the so-called *Wave Digital Filters* (WDFs) [Fett86]. They are well suited due to their excellent numerical properties, to their simplicity, and also due to the generality of the theory developed around them. They have proven to provide a *forced response stability* even under looped conditions. All these arguments makes it natural to choose the WDF as the main filter class.

Different discrete transforms are usually main issues in the literature and conferences on DSP. The most popular are the *Discrete Fourier Transform* (DFT) and the *Discrete Cosine Transform* (DCT). From an implementation point of view, these transforms do not differ in any fundamental way from LTI filters. Thus, the same methodology can as well be applied in this domain [Defi89].

IV. Architecture

An almost infinite variety of choices exist for the VLSI implementation of filter algorithms. First of all on the architectural level, then on the representation level, on the bit level and finally on layout level. Choosing a fixed architecture will never be optimal for all possible applications but it will simplify both the realization and implementation step.

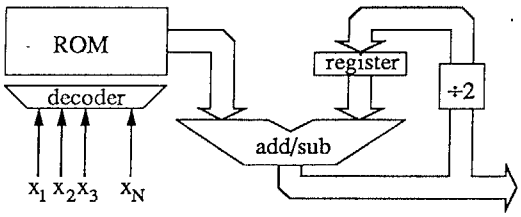

Figure 2 *Basic distributed arithmetic processor*

The target architecture described here is based on a bit-serial multi-processor structure of *data flow type* [Dinh84]. The basic processor performs a calculation of an *innerproduct*, i.e., a vector multiplication between a fixed coefficient vector and a data vector using the *distributed arithmetic* algorithm [Pele74, Siks81, Wanh81]. Using innerproduct operations rather than simply additions and multiplications, together with the multi-processor structure, provides a very powerful architecture from a computational point of view. Furthermore, this architecture can be efficiently used for many fixed filter and discrete transform applications. It also yields a modular and regular VLSI implementation.

The basic hardware requirements for performing this are, a ROM, an arithmetic unit consisting of an adder/substractor and an accumulator register, fig. 2. The basic processor is built-up by an extended serial-parallel multiplier (*shift-accumulator*) and a ROM *look-up-table*. A bit-sliced design style using dynamic registers and a pass transistor full-adder have been used to realize the processor unit. The processor has been implemented in a 2 µm technology cmn20a from VLSI Tech. Inc. (VTI) and an integration density of 3300 MOS/mm^2 was reached for the arithmetic part.

Simulations and test integrations have indicated that the processor can be clocked at 100 MHz, i.e. with a trough-put of 100 Mbit/sec. Thus, sampling rates in the range of 3 - 8 MHz can be obtained. Furthermore, a bit-serial system is very area efficient. Interconnections, both on chip and external, are easily and efficiently implemented in a bit-serial system. This architecture has shown to yield a very impressive trough-put/area performance. Moreover, the processor unit can easily be parameterized and generated automatically from a small set of basic cells.

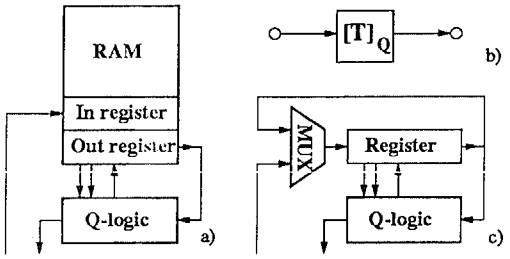

Figure 3 *Delay element, a) RAM realization b) Symbol c) minimum register realization*

To store the internal state variables, i.e. to realize the delay elements from the SFG, RAM or simply shift-registers are used dependent on the application, fig 3. Using one storage unit for each state variable minimizes the required memory band-width. Some special logic to perform different quantization and overflow characteristics completes the active part of the architecture. It has been shown to be advantageous to combine this logic with the storage units.

V. Mapping the DSP Algorithm into the target Architecture

Once a feasible filter class and an efficient high performance target architecture have been found, the remaining problem is the realization step. Here an efficient mapping of the filter algorithms into the architecture should be made. Given that the architecture is based on distributed arithmetic, it is clear that an innerproduct representation of the filter algorithms is desired.

a) $$\begin{aligned} w(n+1) &= A\,w(n) + B\,x(n) \\ y(n) &= C\,w(n) + D\,x(n) \end{aligned}$$

$$S = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$$

b) $$\begin{bmatrix} w_1(n+1) \\ w_2(n+1) \\ w_3(n+1) \\ \vdots \\ w_N(n+1) \\ y(n) \end{bmatrix} = S \begin{bmatrix} w_1(n) \\ w_2(n) \\ w_3(n) \\ \vdots \\ w_N(n) \\ x(n) \end{bmatrix}$$

Figure 4 *State-space representation.*

Transforming the filter algorithm into a *numerically equivalent state-space form* (NESS) [Oppe75, Wanh81], i.e., to describe the algorithm in a matrix form solves this, fig. 4. From a DSP point of view it is essential that the state-space transformation is numerically equivalent, otherwise some of the desired numerical properties might be lost. Using the compacted form, described in fig. 4b, the innerproducts are easily found and in fact all possible parallelism for the algorithm is extracted. The state-space description also reduces the quantization noise to a minimum. Moreover, a description of the algorithm in matrix form simplifies the use and the construction of CAD tools.

All LTI digital filters can be described in state-space form as

well as the discrete transforms, which shows that this approach is general and that it can be used uniformly for different applications. Here again, the choice of filter algorithm is important. Some filters give a sparse state-space matrix, e.g., the *Wave Digital Lattice Filters* [Gasz85].

Each row of the state matrix forms an innerproduct and can be directly mapped and implemented on a single basic processor. Using the correct number of processors makes it possible to calculate all new state variables concurrently. Dependent on the application, various configurations of processors can be used, e.g., in the case of a low speed application only one multiplexed processor can be used for calculating all the innerproducts. Also channel multiplexing can be employed.

VI. CAD Framework

A *Computer Aided Design* environment has been defined to support the described methodology [Sjös89]. It can be viewed as a vertically sliced synthesis system. Already existing filter synthesis tools are used for the synthesis step, some adaptations will be made to transform the output format to a hierarchical netlist describing the signal flow graph SFG in fig. 6.

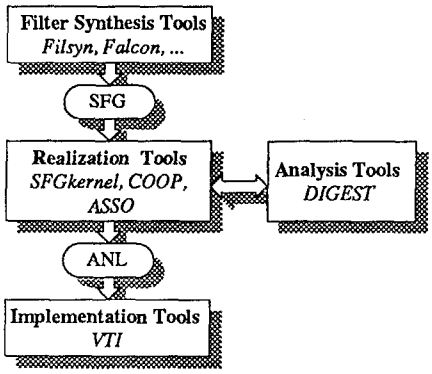

Figure 5 *CAD framework*

A special purpose software, under the name of SFGkernel, has been developed where the signal flow graph is entered using a graphical entry supporting hierarchy. Thus the filter can be given on the level of the *Wave Flow Graph* (WFG) rather than as a flattened SFG, this both saves time and eliminates the probability for errors. This software transforms the hierarchical SFG to state-space form where each state coefficient is given as an explicit expression of the initial filter coefficients. This allows the state-space *COefficient OPtimization* program (COOP) to optimize the filter coefficients such that minimum word-lengths for the state coefficients are obtained. COOP is based on the *Simulated Annealing* algorithm.

To tune different filter characteristics DIGEST [Clae84] is employed. DIGEST is also useful for different analysis and simulations of the DSP algorithm as well as to verify the correctness of the state-space transformation and the optimization.

A tool taking an optimized state-space description as input and generating an *Architecture Net List* (ANL), is under development, *Architecture Synthesis and System Optimization* (ASSO). The ANL should describe each needed module and all its parameters. ASSO will estimate the chip area and speed efficiency. Using this tool it is believed that a fast and accurate evaluation of different solutions can be made. The final intention is to incorporate this tool into the optimization step such that an even more optimal final chip layout is gained.

The ANL is passed on to the implementation step where different module generators, implemented using VTIvip, are used to generate the actual layouts as well as simulation models. In fact all the implementation step is based on the VTI system [VTI 87].

The CAD framework does not serve as a *silicon compiler*, but is rather a special purpose tool-box for the designer. Hence, human interaction still remains essential on each level.

VII. Example

To clarify the methodology a simple example is given. Consider a simple third order lowpass filter of Cauer type with a sampling frequency F_S. Different trade-offs depending on this frequency will be summarized later on. A WDF can be derived from a lumped element ladder reference filter. Many different choices and variations are available for the synthesis [Fett86], but this is not presented here. The final wave flow graph is an exact and detailed description of the filter algorithm.

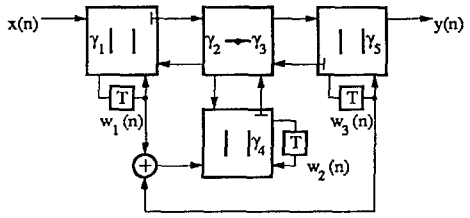

Figure 6 *Wave flow graph of a third order WDF*

The wave flow graph is entered using a graphical editor associated to the SFGkernel tool. Then the WFG is flattened to a normal SFG and a state-space transformation is performed. It is very important that it is made numerically equivalent, i.e., keeping the same state variables, otherwise the excellent numerical properties of WDFs are lost. Using COOP an *optimized state space description* (OSSD) can be obtained.

Scaling of the filter is made by computing different norms using DIGEST and by multiplying rows and columns in the state matrix by scale factors. The numerical properties of the original WDF are conserved if the scale factors are chosen to simple powers of two. For example, if it is found that the state variable i should be scaled down with a factor of 2, row i is multiplied by 2^{-1} and column i by 2.

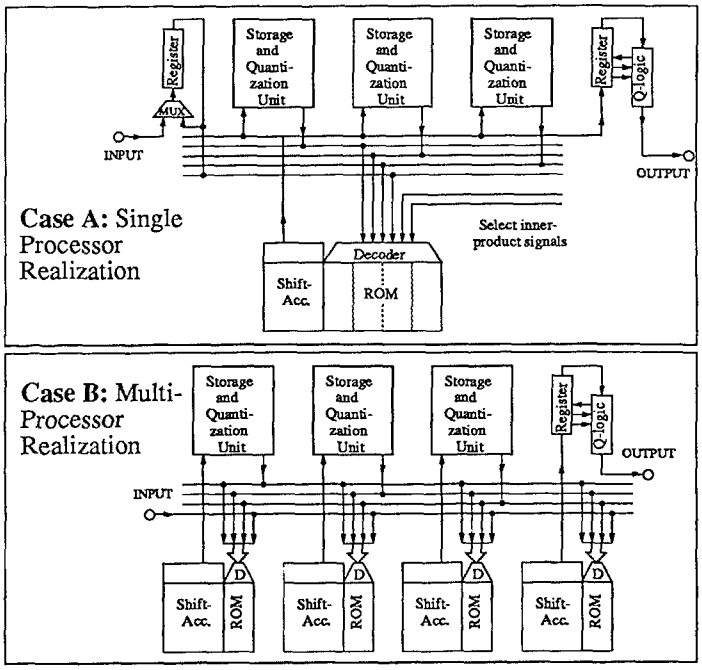

Figure 7 *Single- respectively Multi- processor realization*

Once the OSSD is found, it is transformed into hardware requirements in form of an architecture netlist using the ASSO tool. Many different solutions can be found dependent on the application. Two different possibilities are presented in fig. 7. Case A corresponds to a single processor realization, while case

B shows a realization using one processor per innerproduct.

Assume that the necessary data or signal word-length w_d was found to be 20 bits and that the ROM coefficient word-length w_c is equal to 10 bits everywhere. For a trough-put of approximately 100 Mbit/sec, a theoretical maximum sampling rate of 5 MHz could be obtained using a multi-processor implementation like in case B.

However, due to pipelining of the processors, the maximum sampling rate will be lower. The latency time for a processor is $w_c + w_d$ clock cycles. If quantization and saturation arithmetic is desired, a storage unit of length w_d bit is required. This results in a total cycle time of $2w_d + w_c = 50$ clock cycles. Thus, in case B the maximum sampling rate is reduced to 2 Mhz. In case A four different innerproducts are calculated on the same processor unit and the pipelining is no more a problem. Consequently, case A gives a maximum sampling rate of 1.25 MHz.

Other trade-offs can be made for multi-channel applications. The only difference is that the delay elements are implemented by RAMs instead of shift-registers (fig. 3) when the number of channels is sufficiently large. A diagram showing the maximum sampling frequency versus number of channels is shown in fig. 8a. When the number of channels is less than three, case B shows an achievable speed which is lower than the theoretical curve due to the pipelining problem. Case A and B give the lower and upper bounds for the obtainable performances of this architecture using different numbers of processors.

Area is an other important measurement and therefore a diagram is plotted in fig. 8b. The limit for using RAM storages instead of registers is at 7 respectively 9 channels for case A and B. The difference is smaller in area than in speed for the two cases. These diagrams are specific to the given example and must be evaluated for each application.

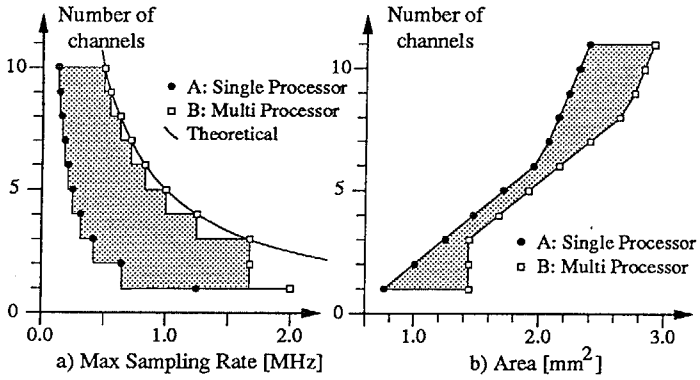

Figure 8 *Speed and Area versus number of channels*

It is clear that the architecture is especially well tailored for high speed applications and/or multi-channel applications. No complete system has yet been implemented using the methodology. However, a decimation filter for a HIFI audio application is under initial development. Furthermore, a 16-point DCT chip containing 95 000 transistors is soon ready for integration. This last design has shown to have some very interesting characteristics [Defi89].

VIII. Conclusion

It has been shown that appropriate methodologies are essential for fruitful ASIC implementation of DSP algorithms. The methodology presented in this paper describes a vertically sliced synthesis system specially tailored for efficient implementation of digital filters and discrete transforms. The concept of the mapping between the DSP domain and the VLSI domain is not only efficient and general, it even improves the numerical properties and facilitates the construction of software tools. The architecture has shown to be very area and speed efficient, but it suffers from flexibility, i.e., it is only weakly programmable. It is believed that the field of applications where this methodology can be used will grow enormously in the next few years.

IX. Acknowledgements

This work was supported by FSRM, *Fondation Suisse pour la Recherche en Microtechnique*, under grant CS 85/7.

X. References

[Clae84] L. Claesen, H. J. De Man, and J. Vandewalle : "DIGEST: A Digital Filter Evaluation and Simulation Tool for MOSVLSI Filter Implementations", IEEE J. of Solid-State Circuits, Vol. SC-19, No. 3, June 1984.

[Defi89] I. Defilippis, U.Sjöström, M. Ansorge, F. Pellandini : "A 2-Dimensional 16 Point Discrete Cosine Transform Chip for Real Time Video Applications", GRETSI-89 Symposium, Juan-Les-Pins, June 1989.

[DeMa87] H. De Man, J. Rabaey, P. Six, and L. Claesen : "Computer Aided Synthesis Systems for Digital Signal Processing", Proc. Journées d'électronique, Lausanne, Switzerland, October 1987.

[Deny85] P. Denyer, and D. Renshaw : *"VLSI Signal Processing: A Bit-Serial Approach"*, Addison & Wesley, VLSI Systems Series, USA, 1985.

[Dinh84] F. Dinha, B. Sikström, U. Sjöström, and L. Wanhammar : "LSI Implementation of Digital Filters - A Multi- Processor Approach", Proc. Int. Conf. on Computers,Systems and Signal Processing, Bangalore, India, Vol. 3, pp. 1316-1320, December 1984.

[Fett86] A. Fettweis : "Wave Digital Filters: Theory and Practice", Proc. IEEE, Vol. 74, No. 2, pp. 270-327, February 1986.

[Gazs85] L. Gazsi : "Explicit Formulas for Lattice Wave Digital Filters", IEEE Trans. on Circuits and Syst., Vol. CAS-32, No. 1, pp. 68-88, January 1985.

[Ligt86] A. Ligtenberg, M Vetterli, and J.H. O'Neill : "MOVAL: A Framework for turning Digital Signal Processing Algorithms into Custom VLSI", Signal Processing, Vol. 11, No. 2, pp. 119-132, 1986.

[Oppe75] A. V. Oppenheim, and R. W. Schafer : *"Digital Signal Processing"*, Prentice-Hall Inc., Englewood Cliffs, N.J., 1975.

[Pele74] A. Peled, and B. Liu : "A New Hardware Realization of Digital Filters, IEEE Trans. Acoust. Speech, Signal Processing", Vol. ASSP-22, No. 6, pp. 456-462, December 1974.

[Raba85] J. Rabaey, S. Pope, and R. Brodersen : "An Integrated Automated Layout Generation System for DSP Circuits", IEEE Trans. on CAD, Vol. CAD-4, pp. 285-296, July 1985.

[Siks81] B. Sikström, L. Wanhammar : "A Shift-Accumulator for Signal Processing Applications", Proc. European Conf. on Circuit Theory and Design., ECCTD'81,The Hague, The Netherlands,1981.

[Sjös89] U.Sjöström, I. Defilippis, M. Ansorge, F. Pellandini : "CAD Environment for Digital Filter Design and Implementation", ISSSE-89, Paper No. 105, Erlangen, September 1989.

[VTI 87] VLSI Technology Inc. : *Diverse Manuals*, VTI, 1987.

[Wanh81] L. Wanhammar : *"An Approach to LSI Implementation of Wave Digital Filters"*, Linköping Studies in Science and Technology, Diss. No. 62, Linköping University, Sweden, April 1981.