Published in 7th European Signal Processing Conference EUSIPCO 7, 1875-1878, 1994
which should be used for any reference to this work

A Low Power VLSI Architecture with an Application to Adaptive Algorithms for Digital Hearing Aids

Alexandre HEUBI, Sara GRASSI, Michael ANSORGE and Fausto PELLANDINI

Institute of Microtechnology, University of Neuchâtel, Rue de Tivoli 28, CH-2003 Neuchâtel-Serrières, Switzerland

Phone: ++41 38 301 479, Fax: ++41 38 301 845, E-Mail: heubi@imt.unine.ch

Abstract. A new architecture suitable for an effective low power VLSI implementation of a large class of digital signal processing algorithms is presented, which shows to be particularly well-adapted to fulfil the requirements of portable and autonomous microsystems. Starting with the precise specifications of the application algorithms, an appropriate scheduling method is first applied to optimize the data flow. The actual VLSI implementation is then performed, resorting to parameterized cell compilers for the automatic generation of the primary modules. As an example, the implementation of an adaptive spectral sharpening algorithm suitable for future all-digital hearing aids is discussed. The resulting silicon area is approximately 4 mm^2 for a 1.2 m CMOS process, and the estimated power consumption at a sampling rate of 8 kHz is about 4 mW at 5V (0.65 mW at 2V).

1. Introduction

In the fields of telecommunications, hearing aids, and electronic instrumentation, there is an important demand for new generations of high performance portable applications. The requirements of such applications are manifold, and can be expressed in terms of improved functionalities, enhanced miniaturization, and reduced power consumption to extend the system autonomy.

For flexibility and efficiency reasons, the signal processing facilities to be incorporated are more and more realized digitally on dedicated ASICs. However, in order to achieve optimal solutions with respect to the power consumption, the designs should be performed coherently and carefully at all levels, including the algorithmic, architectural, logical, and layout levels.

This paper proposes a contribution to the design and optimization of low power VLSI architectures suitable for effective implementations of various digital signal processing (DSP) algorithms such as FIR and IIR filters, adaptive filters, and Fourier transforms, with a particular strength on the power consumption which should be kept as low as possible.

The principles of the proposed architecture are discussed in Chapter 2, whereas Chapter 3 presents the primary functional modules. The design of these modules has been performed down to the layout level to get a precise estimation of the achieved performance. Hence, a set of functional modules has been developed, which, when properly assembled and parameterized, provide an efficient solution for the ASIC implementation of a broad variety of digital filtering algorithms.

Finally, Chapter 4 presents an application example for an adaptive spectral sharpening algorithm for digital hearing aids.

2. Processor architecture

For digital CMOS circuits, the power consumption is known to be essentially determined by the dynamic consumption, which can be expressed as [1] :

$$P = f \cdot C_{eq} \cdot Vdd^2 \qquad \text{(EQ 1)}$$

where f is the clock rate, Ceq the equivalent capacitance, and Vdd the supply voltage of the considered circuit. All three quantities should be kept as low as possible.

Substantial savings can be achieved with respect to the equivalent capacitance Ceq by strictly limiting the processing activities to the resources providing a direct contribution to the required computations. Hence, the next principles were applied :

a) Idle modules should be set into a power-saving STAND-BY mode.
b) The processor architecture and modules should be organized in a way to limit the overall data transfer to the strict minimum, local data traffic being preferred versus global traffic.
c) Larger memories should be split into a set of smaller memories, where a single one is active at a time.

Also, the structural regularity of most DSP algorithms should be utilized to simplify both the scheduling and the hardware implementation.

Finally, there is a global trade-off to be determined between f, Ceq, and Vdd in order to obtain the optimal power consumption. Indeed, reducing the supply voltage lowers the maximum achievable clock rate due to the increased signal propagation delay. Hence, in order to cope with the required computation throughput, it is usually necessary to extend the parallelism of the architecture [2], which in turn will have an effect on the equivalent capacitance! In this context, the architecture should be flexible enough to let the degree of parallelism be best fitted to the considered application.

2.1 Proposed solution

The data flow of most DSP algorithms features a sequential access to the data (e.g. filter coefficients and state variables), and it is therefore possible to recourse to sequential memories for the data storage. This solution drastically simplifies the data address computation and decoding, and the signal data updating required at each sampling period is merely achieved by virtual data shifts.

This concept can also be applied to more complex algorithms such as adaptive filters and Fourier transforms, adjusting the memory addressing scheme when required.

Moreover, a serial-parallel arithmetic has been chosen for the realization of the arithmetic unit, since it provides a processing rate which is high enough for most filtering applications in the audio domain (sampling rates in the range of 10 to 50 kHz), while keeping the implementation complexity at an appropriate level.

Finally, the scheduling has been hierarchically organized to limit the processing rate of each module to the strict minimum [3].

3. Realization

The main functional modules of the proposed architecture to be discussed are the sequential memory, the arithmetic unit, and the sequencer.

3.1 Sequentially accessed memory

Basically, sequential memories can be realized either using shift register banks, or using arrays of memory cells, in which case all cells of a given row are simultaneously addressed by a sequential register. Contrary to the first solution, the second one does not require any global data movement.

Comparing both solutions, one observes that for higher storage capacities (number of words · number of bits/word), sequential array memories are more efficient with respect to area and power consumption [4]. Conversely, shift register banks are more economic for small memory sizes.

3.1.1 Sequential addressing scheme

The classical memory addressing scheme relies on a binary counter followed by an address decoder. For sequential memories, a better solution from an area and power consumption point of view consists in using a shift register which is directly embedded in the memory structure. Figure 1 shows the principle of the retained solution, which was further refined to avoid any addressing conflict by ensuring that a single data row is selected at a time [4].

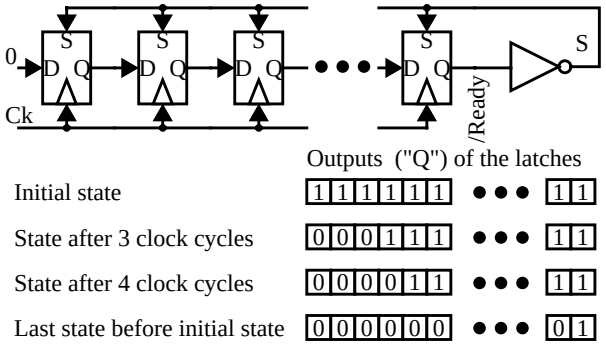

FIGURE 1. Principle of sequential addressing

3.2 Arithmetic unit

The scalar product is certainly one of the most fundamental operations in digital signal processing. Accordingly, the arithmetic unit was optimized for this purpose.

The filter coefficients are handled using Booth's recoding scheme to reduce at least by a factor of two the number of relevant partial products occurring in a multiplication. The chosen solution provides a recoding of two bits at a time, with a coding overlap of one bit [5].

The resulting arithmetic unit illustrated in Figure 2 computes scalar products in full precision and was extensively pipelined to increase the throughput (for instance the multiplication of two 20 bit numbers is performed in 6 clock cycles) [4]. The number of memory accesses was also minimized.

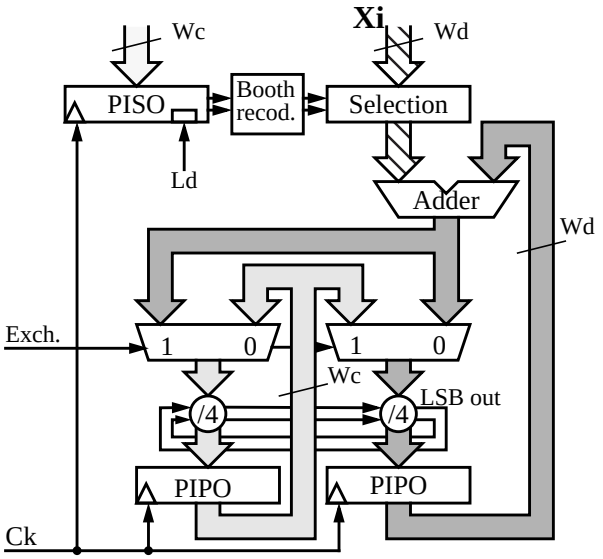

FIGURE 2. Arithmetic unit with auxiliary registers

3.3 Sequencer

The sequencer was designed to support a hierarchical scheduling in order to minimize the processing rate of all resources. The sequencer is realized as a ROM-based finite state machine [3, 9].

3.4 Performance evaluation

The layout of the modules was designed in the COMPASS CAD environment (1.2 m5V CMOS CMN12 process from VLSI Technology, Inc.). All modules can be automatically generated using parameterized cell-compilers.

The main characteristics are summarized in Table 1 for the case where all data (coefficients and state variables) are represented with the same fixed point precision (Wc = Wd = W).

	Read/Write memory	Arithmetic unit	Sequencer
Max clock rate	25 MHz (32 · 64 bits)	20 MHz (32 · 32 bits)	25 MHz (32 · 64 bits)
Power consumption (energy, given in pJ)	(per read + write cycle) 8 + nbwrds·0.75 + W·14.5 + nbwrds·W·0.7	(per multiplication) 110 + W·62 + W·W·7.8	(per read cycle) 8 + nbwrds·0.75 + nbbits·1.75 + nbwrds · nbbits· 0.1
Area (λ = 0.6 µm for the CMN12 technology)	(120λ + W·23λ) · (280λ + nbwrds·28λ)	1200λ · (300λ + W·35λ)	(215λ + nbbits·12λ) · (70λ + nbwrds·12λ)

TABLE 1. Main characteristics of the constituent modules

The indicated power consumption corresponds to typical (i.e. mean) values achieved either from simulations based on a set of random inputs, or from an estimation of the transition probability of each electrical node.

4. Application example

As an application example, the implementation of an adaptive speech processing algorithm suitable for future all-digital hearing aids will be discussed. The considered algorithm aims at improving the speech intelligibility/quality by spectral sharpening and can be applied to perform either a noise reduction or a speech enhancement by slightly modifying the structure [6].

In the sequel of the paper, the discussion will be focussed on the spectral sharpening algorithm for noise reduction.

4.1 Spectral sharpening for noise reduction

Figure 3 presents the block diagram of the noise reduction algorithm where the core of the algorithm is given by the adaptive decorrelator together with the analysis and synthesis filters. In the scope of this work, the elementary first-order highpass filter was not considered for the VLSI implementation.

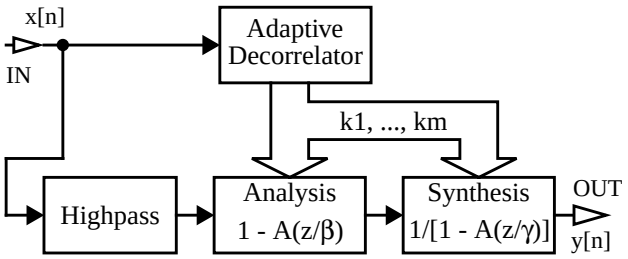

FIGURE 3. Spectral sharpening for noise reduction

Figure 4 shows the signal flow graph of the core of the algorithm. The upper part corresponds to the decorrelator which was realized using an adaptive gradient lattice filter.

The decorrelator is characterized by a modular structure and a local adaptation scheme where the output signals provided at each stage are featuring a decreasing autocorrelation. The partial correlation (Parcor) coefficients k1, ..., km generated in the decorrelator and updated at each sampling interval are transferred to the analysis and synthesis filters.

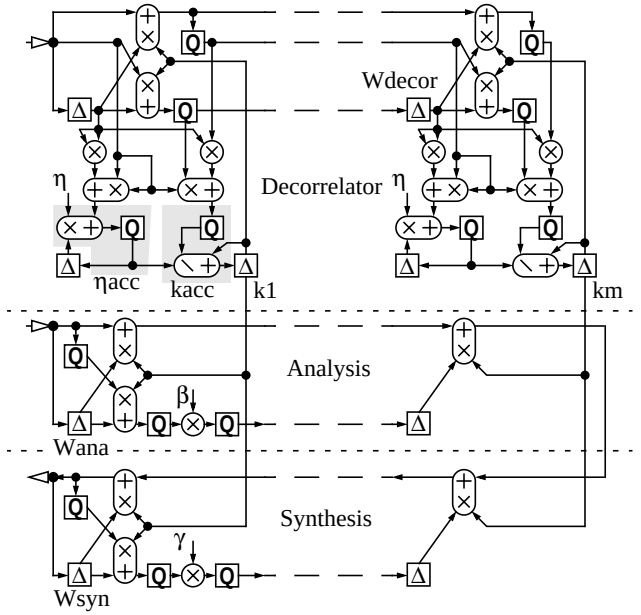

FIGURE 4. Signal flow graph of the core of the algorithm

These filters are processing the speech signal to actually achieve the spectral sharpening by applying an overall transfer function $H(z) = [1 - A(z/\beta)] / [1 - A(z/\gamma)]$ with $(0 < \beta < \gamma < 1)$.

The required filter order is typically m = 8..10 for speech signals sampled at 8 kHz. In this study, we have chosen m = 8.

4.2 Implementation specifications

An important issue to be taken into account at both algorithmic and implementational levels are the nonlinear effects (overflow and quantization) resulting from the finite precision data representation. These effects were thoroughly analyzed and simulated to determine the minimum data wordlength and related scaling factors, to select the type of quantization operation to be used, and to possibly simplify the division occurring in the decorrelator. A companion paper presents this study in detail [7], and the obtained results are summarized in Table 2.

Sampling frequency		8 kHz
Filter order		m = 8
Data wordlengths:	Power estimation (ηacc on Figure 4)	Wηacc = 32 bits (norm = 64)
	Coarse division and k accumulation (kacc on Figure 4)	Wkacc = 16 bits
	Decorrelator (48 mac)	Wdec = 16 bits
	Analysis (22 mac)	Wan = 16 bits
	Synthesis (22 mac)	Wsyn = 16 bits
Quantization operators		Sign magnitude truncation

TABLE 2. Overall implementation specifications ("mac" = multiply and accumulate)

In particular, the operations marked as ηacc and kacc in Figure 4 could be simplified. Indeed, the optimal value of the parameter η shows to be insensitive to the input signal and is about η_{opt} 0.98, which can be approximated by $(1 - 2^{-6} - 2^{-8})$. Hence, the corresponding product can be processed with only two additions.

Furthermore, the division occurring in kacc can be replaced by a coarse division without any significant degradation of the output signal [7]. This is achieved by approximating the power estimation (divisor corresponding to ηacc) by a power of 2 and by shifting the cross power (dividend at the input of kacc) accordingly [7].

Basically, four kind of operations should be performed for the execution of the algorithm: multiply and accumulate (mac), ηacc, coarse division and accumulate (kacc), and the quantization.

We have chosen to implement the algorithm on time-multiplexed resources. To achieve a sufficient computational throughput, the mac and quantization operations are executed on two (identical) arithmetic units to be implemented with the developed cell-generators, whereas the ηacc and kacc operators are realized separately using a low power standard cell library.

Finally, six different types of resources are required for the implementation of the noise reduction algorithm, namely four computation operators (mac, ηacc, kacc and quantization), memories and data buses.

4.3 Scheduling

Once the needed resources have been identified, the scheduling of the algorithm can be performed. For this purpose, we used an adapted version of the optimization technique known as "TABU

search" [8] which is particularly well suited for the scheduling of DSP algorithms at the macro-cycle level (multiplication time).

In order to obtain a regular scheduling and to localize the data transfers, one mac unit is bound with the operations of the decorrelator, whereas the other mac unit is bound with the analysis and synthesis filters. The obtained scheduling has an iteration period of 48 macro-cycles with a computational load of 100 % for the first mac, and of about 92 % for the second one.

The fine grain scheduling performed at the level of elementary clock cycles, the resource allocation and the splitting of the memories into smaller units are currently made by hand. Finally, the sequencer is designed accordingly.

4.4 Results

The different modules and the floorplan of the core of the noise reduction algorithm were designed in the COMPASS CAD environment for the 1.2 m 5 V CMN12 CMOS process from VLSI Technology. The resulting silicon area is approximately 4 mm^2.

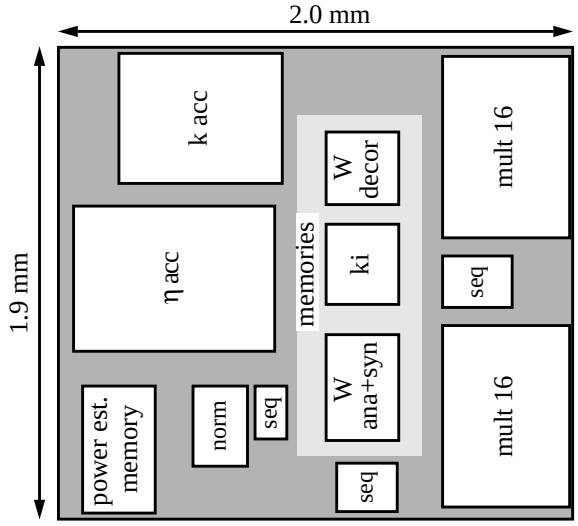

FIGURE 5. Floorplan of the ASIC implementation

The power consumption was estimated for Vdd = 5 V using Table 1 together with complementary simulations for the specialized modules. As can be observed in Table 3, the major part of the power consumption, which is about 4 mW at a sampling rate of 8 kHz, is absorbed by the multiplication operations.

Resources	Number of proc. cycles	Energy per cycle (in nJ)	Total energy (in nJ)
Mac 16·16 bits	92	3.1	285
η acc	8	2.5	18
k acc	8	0.5	4
All memories	–	–	50
Sequencer	–	–	26
Data transfers	–	–	50
Control signals	–	–	24
TOTAL	–	–	< 500

TABLE 3. Power consumption estimation per sampling interval (Vdd = 5 V)

Since the computational margin of the circuit is higher than 10, it is possible to lower the supply voltage down to 2 V, and the power consumption can be reduced by a factor of 6, resulting in P 0.65 mW and Idd 0.33 mA. Further verifications should be performed at that level.

5. Conclusion

This paper proposes a new architecture for the low power VLSI implementation of DSP algorithms. This architecture is in particular intended to be used for portable and autonomous microsystems, and was applied as an example to the implementation of a spectral sharpening algorithm for digital hearing aids.

In the short term, the future extensions of the project will be focussed on improving the data flow optimization (fine grain scheduling, resource allocation), including an automated generation of the sequencer. It is further previewed to extend the module library with several essential functionalities needed for the realization of complete microsystems, i.e. other digital operators, AD / DA converters, and mixed-mode modules.

Acknowledgments

This project was supported by the Swiss National Found for Scientific Research (FNRS) under Grant 20-33997.92. Also, the authors would like to thank CSEM SA, Neuchâtel, Switzerland, for the low power standard cell library CSEL_LIB.

References

[1] C. Piguet, V. von Kaenel, J.-M. Masgonty, J.-F. Perotto, R. Klootsema, "Basic Design Techniques for both Low-Power and High-Speed ASIC's", Proc. EUROASIC'92, Paris, France, 1992, pp. 220-225.

[2] A. P. Chandrakasan, S. Sheng, and R. Brodersen, "Low-Power CMOS Digital Design", IEEE J. of Solid-State Circuits, Vol. SC-27, no. 4, April 1992, pp. 473-484.

[3] C. Piguet, E. Dijkstra, A. Theys, M. Stauffer, J.-F. Perotto, "A Design Methodology of Micro-programmed Controllers for Custom IC's", Proc. EUROMICRO'87, Portsmouth, UK, Sept. 1987, pp. 463-470.

[4] A. Heubi, M. Ansorge, F. Pellandini, "Architecture VLSI faible consommation pour le traitement numérique du signal", Proc. GRETSI'93, Juan-les-Pins, France, Sept. 1993, pp. 1083-1086.

[5] H. Sam, A. Gupta, "A Generalized Multibit Recoding of Two's Complement Binary Numbers and Its Proof with Application in Multiplier Implementations", IEEE Trans. on Computers, Vol. C-39, no. 8, Aug. 1990, pp. 1006-1015.

[6] A. Schaub, P. Straub, "Spectral Sharpening for Speech Enhancement / Noise Reduction", ICASSP'91, Toronto, Canada, May 1991, pp. 993-996.

[7] S. Grassi, A. Heubi, M. Ansorge, F. Pellandini, "Study of a VLSI Implementation of a Noise Reduction Algorithm for Digital Hearing Aids", Proc. EUSIPCO'94, Edinburgh, UK, Sept. 13-16, 1994.

[8] F. Glover, M. Laguna, E. Taillard, D. de Werra, "Tabu Search", Annals of Operations Research, Volume 41, Ed. J. C. Baltzer AG, Basel, Switzerland, May 1993, ISSN 0254 5330.

[9] A. Theys, E. Dijkstra, C. Piguet, "Discrete Transforms on a Chip: Design and VLSI Implementation", Proc. Int'l Symp. on Applied Control, Filtering, and Signal Processing '87, IASTED'87, Geneva, CH, June 1987, pp. 241-245.