AN APPLICATION-ORIENTED ANALYSIS OF POWER/PRECISION TRADE-OFF IN FIXED AND FLOATING-POINT ARITHMETIC UNITS FOR VLSI PROCESSORS

Francesco Pappalardo and Giuseppe Visalli
Advanced System Technology
ST Microelectronics Catania, Italy
Email: Francesco.Pappalardo@st.com
Email: Giuseppe-ast.Visalli@st.com

Mirko Scarana
Department of Electronic Engineering
University of La Sapienza
Rome, Italy
Email: Scarana@die.uniroma1.it

ABSTRACT
Floating and fixed-point formats represent the most used representation in VLSI DSP systems. Fixed-point formats are often preferred for lower power and faster operation, when the algorithm permits the appropriate scaling factors. Floating-point formats require additional hardware and power, but guarantee better precision performance. This work compares the energy dissipated by the key arithmetic units in DSP systems, addition and multiplication, in both floating- and fixed-point scenarios, also considering reduced precision and power-aware architecture solutions. The evaluation has been performed referring to STMicroelectronics 180nm,130nm and 90nm CMOS processes. The results show that specialized floating-point formats can be the best energy-efficient solution in low power DSP.

1 Introduction

For the past ten years, the decision about the fixed or floating point support for real number calculations in DSP involved engineers, researchers and silicon foundries [1]. The latter tackled the problem from the view point of the implementation cost. The engineers required a better implementation both in terms of area and timing. Finally, algorithm and application researchers required high precision computations. Nowadays, the problem of low-energy real number calculations represents a common point for the silicon foundries, engineers and application researchers: the problem of a DSP real number support with minimal energy dissipation, depends on several and important factors at different abstraction levels: technology, circuits, architecture and algorithms [2]. This work faces this problem at circuit and algorithm levels, addressing application specific VLSI circuits for signal processing, where additions and multiplications are the most power consuming operations. According to real application data, signal processing algorithms require an average 40% multiplications and 60% additions (Table 1).
In general, floating-point support requires additional hardware and power, using the IEEE standard representation [3]; existing studies suggest the use of reduced floating-point formats as the best approach for low-energy math operations [4]. The goal of this research is the energy comparison, executing different representative signal-processing benchmarks, of different floating- and fixed point formats. Since the floating-point multiplier is responsible of the major power dissipation in a FPU, we considered the truncated multiplier, which drastically reduces the area and the power, increasing the percentage precision error. The truncated multiplier performs calculations using the most important partial products (Fig. 1) [5] [6], thus introducing a computation error, which can be reduced with specific circuits for accuracy improvement [7]. Our results show that a core DSP unit with a hardware accelerator for floating-point calculations can execute the considered benchmarks with the minimum energy dissipation even when compared to the best fixed-point format. The paper is organized as follows: section 2 introduces the benchmark set, the parameters used for performance (i.e. precision) comparison, and the power evaluation used. Section 3 presents our strategy for using fixed-point formats in DSP systems, the reduced floating-point formats and the low-power truncated mantissa multiplication. Section 4 presents the simulation results and section 5 reports our conclusions.

2 Benchmarks and number formats used

Table 2 shows the set of benchmark applications used for our analysis. Each considered algorithm has a specific global performance parameter (Table 2) in order to define the impact of the number formats on the application-level results. The decoding algorithms (Zero Forcing and Adaptive Equalizers, Soft and Soft Output Viterbi) represent digital transmission systems into white Gaussian noise channel (AWGN). The decoder introduces a percentage of mistakes represented by the bit error rate (BER), employing a signal to noise ratio (SNR) measured in Decibels (dB). The filtering algorithms (Gauss and Jakes) represent the two common noise sources filtered by a simple auto regressive (AR) system [8]. The produced waveforms have been compared to the reference wave, which use an IEEE standard arithmetic. The mean percentage error (MPE) em-

ployees the following ratio:

$$|\frac{\underline{w}_{i,754} - \underline{w}_{i,reduced}}{\underline{w}_{i,754}}| \quad (1)$$

Where $\underline{w}$ is the output waveform. The sine function [14] introduced the percentage error:

$$|\frac{sin(\alpha_{754})_{754} - sin(\alpha_{reduced})_{reduced}}{sin(\alpha_{754})_{754}}| \quad (2)$$

Finally, the FFT's global performance parameter compares the area of the input waveform both in IEEE and reduced precisions. As for power consumption evaluation, we estimated the energy dissipated by adders and multipliers dedicated to the fixed and floating-point formats used. The floating-point support employs a reduced format (see section 3), which guarantees the same global performance of fixed-point arithmetic. Starting from the HDL architectural description (Verilog) both the fixed and floating-point considered units (adder and multiplier), a complete synthesis flow has been done, using standard CMOS industrial libraries at 180nm,130nm [9] and 90nm low-leakage at 200 MHz. We performed a post-synthesis and post-technology-mapping power evaluation by means of Synopsys Power Compiler starting from the circuits' simulated activity. The use of a low-leakage library in synthesis allows the energy comparison considering the dynamic power only. The performance evaluation of the proposed signal-processing algorithms has been made possible by a C++ behavioral model of the fixed and floating-point units. In particular, the C++ class *CFloat*, emulates the variable-precision floating-point format. The key operators were emulated keeping faithful to the physical implementation; the " $*$ " operator performs calculations using the truncated mantissa multiplier. The member functions intercept the basic math (+,*) and relational operators, storing the input operands in a test bench file. The Verilog simulation environment read the test bench files with appropriate PLI (Program Language Interface) routines, stimulating the hardware to perform the involved operations.

Algorithm	Addition	Products
Jakes noise filtered	52.55	47.45
Gaussian noise filtered	65.38	34.62
Zero Forcing Equalizer	58.59	41.41
Soft Viterbi	57.93	42.07
Soft Output Viterbi	60.64	39.36
Adaptive Equalizer	55.23	44.77
Sine Function	50.00	50.00
Fast Fourier Transform	54.57	45.42

Table 1. Percentage of operations

3 Algorithms implementation and formats evaluation approach

3.1 Fixed point algorithm implementation

If the fixed-point format is chosen, since ANSI C does not provide fixed-point types, it is common practice to translate standard ansi C algorithms by replacing the real operations with integer operation. The process of conversion introduces an output rounding noise. Usually, the conversion utility assigns a particular fixed-point format to each internal variable, in order to reduce the rounding noise: a framework devoted to ANSI C floating-point to fixed-point conversion analyses the dynamic range of each variable and calculates the integer part (IWL=Integer Word Length) and the decimal part (DWL=Decimal Word Length) to be used in fixed-point representation [10][11]. The real variables are split up into (few) different groups, each with a specific word-length. The use of scaling factors represents a further precision optimization: several signal processing algorithms permit to scale their internal node with modest performance degradation. Several frameworks devoted to the floating to fixed-point conversion provide dynamic range profiling (FixPt [12]). Other frameworks introduce a further auto-scaling algorithm in order to increase the precision, working with a larger DWL. Other tools analyze the transfer function of internal node in a simply linear discrete-time system to analytically calculate the dynamic range of an internal node. (This bound depends on the node's transfer function and the input signal's spectrum). Our approach is very similar to the FixPt C++ class library. We overloaded the basic math operators (Table 3), analyzing the dynamic range of each internal variables. The process of profiling starts defining an identifier for each real variable:

$$Id = trace("realvar")$$

The occurrences of "realvar" after calculation will be stored in a custom list:

$$traceid(Id, realvar)$$

Such profiling procedure produces the required IWL for each considered real variable. As for the decimal part, we operate with a unique optimized DWL, to reduce the required alignments by shifting. A dedicated C++ class *fint* supports the variable-precision fixed-point format. The overloaded math operators, which require addition and multiplications, store the operands in an output file used as test bench vector.

3.2 Reduced floating point formats used

Reduced precision floating point and power-aware architectures represent the basic guidelines for the energy reduction in a floating-point unit (FPU). The reduced format has advantages in application specific cores, where classes of applications require the same precision. The choice of

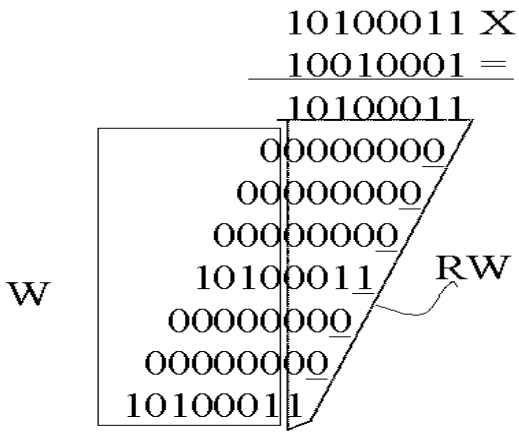

Figure 1. The principle of truncated multiplication

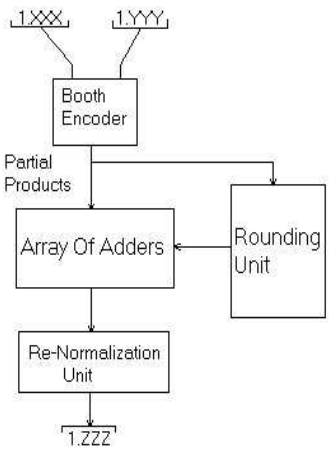

Figure 2. The truncated floating-point mantissa multiplier.

a reduced format has to consider several factors related to the significand and exponent fields. In particular, the significand has to guarantee a sufficient precision in the result. Additionally, the exponent field regulates the range of representable number. A reduced format has a good significand/exponent width ratio, which permits the execution of signal processing algorithms with the minimal loss of precision and exception events (overflow, underflow, de-normal). An exponent field too small causes a quick flushing to zero operating differences between two little and nearby floating-point numbers, in absence of de-normal support [13]. Since the floating-point multiplier wastes the 60%-70% of energy dissipated by a FPU, the low-power research provided different solutions for a reduced power FP multiplication. In particular, the proposed solutions fall into two categories:

. Parallel truncated multiplier (TM) with precision error, which saves 70% in power.

. Serial multiplier: Variable precision with big latencies.

Algorithm	Global Parameter
Jakes noise filtered	Mean Percentage Error
Gaussian noise filtered	Mean Percentage Error
Zero Forcing Equalizer	Bit Error Rate at4 dB
Soft Viterbi	Bit Error Rate at 3dB
Soft Output Viterbi	Bit Error Rate at 2dB
Adaptive Equalizer	Bit Error Rate at 4dB
Sine Function	Mean Percentage Error
Fast Fourier Transform	$\|H(0)\|/\|H(0)_{double}\|$

Table 2. The Global Parameters

Being devoted to high speed applications, we considered the TM, which performs calculations using a limited set of partial products. This architecture introduces a computation error, which can be reduced until 1 ulp (unit in the last place). In particular, the corrected result from truncated multiplication reduced the rounding and truncation errors by including a constant correction (Fig. 2). This constant correction derives from a prior error analysis (fixed bias correction) or the excluded partial products permit the average error reduction considering the changes in the traffic's statistical properties. The use of normalized operands (MSB=1) reduces the percentage error. The proposed signal-processing benchmarks were executed using FPU which employed a truncated multiplier and a single-path floating-point signed adder. The used FPU did not provide the de-normal support, so we coded the real numbers with normalized significand and a 2's complement exponent.

4 Simulation Results

The main goal of this research is to evaluate methods for improving the energy efficiency of floating point support versus fixed-point in signal processing applications. Thus we evaluated the algorithms' performance and power considering the best fixed-point format and the reduced floating-point support. As for performance, the formats imply similar results measured as application-level parameters (Table 4). The fixed-point support has a better resolution, with respect the floating-point number, operating with the same word width. Several results confirmed this assumption: Jakes, Gauss, Sine and FFT. The remaining algorithms have the bit error rate as global performance parameter. This parameter is strongly related to the error introduced performing multiply and accumulate operations (metrics, filtering, equalizations). The presence of an exponent field in the floating-point support permits a greater

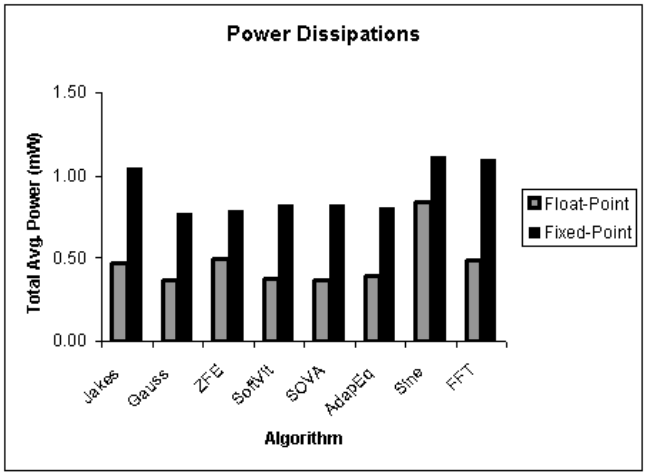

Figure 3. The average power dissipation, executing the considered signal-processing algorithms (130nm technology)

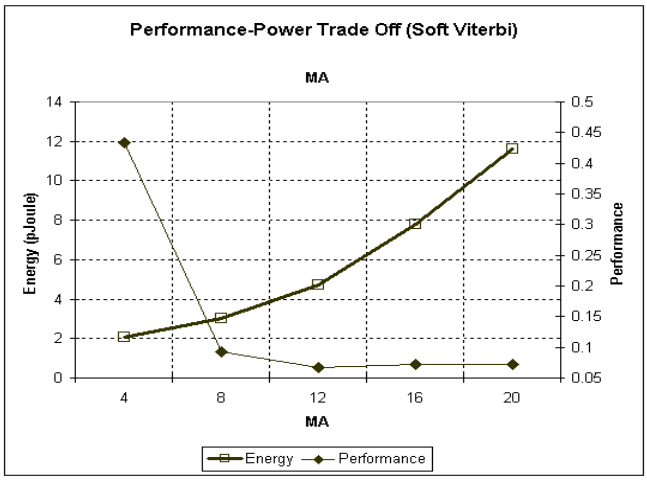

Figure 4. The performance power trade-off in the soft viterbi scenario

”Limit of the Sum” LOS [8], i.e.:

$$LOS_{float} > LOS_{fixed} \quad (3)$$

This issue implicates a better performance in the decoding algorithms using the floating-point support. We evaluated the average power in floating-point or fixed-point as follows:

$$P_{Average} = \alpha+ \cdot P_{Adder} + \alpha* \cdot P_{Multiplier} \quad (4)$$

where $\alpha+$ and $\alpha*$,represent the percentage of addition and multiplication in the considered algorithm (Table 1). The results indicated the signal processing algorithms (both in fixed and floating point), which use trigonometric functions (Jakes, FFT and Sine), were more power hungry compared to filtering and decoding problems. Moreover, the power optimization for the floating-point multiplier (TM) drastically reduces the total power budget, in every considered algorithm as illustrated in Table 5 and 8 in 180nm low-leakage technology, Table 6 and 9 (Fig. 3) in 130nm low-leakage technology and finally Table 7 and 10 employing a 90nm high-VT technology. Zero forcing equalizer and Sine test benches represent the unique exceptions in 180nm. If we have not used the truncated multiplier, the fixed-point support should be more convenient. To better evaluate the effectiveness of the TM approach, we experimented with various mantissa precisions (MA), each having a different average power figure. The input trace consisted of the soft Viterbi algorithm applied to the single path signed adder and truncated multiplier in the floating-point context. Fig. 4 shows the bit error rate, the average power consumption (130nm) for different mantissa precisions. From this figure, we observe the following:

1. Performance remains basically unchanged varying the precision from 12-bit to 23-bit
2. Instead, the power consumption increases more than 160% in the same bit interval.
3. Improvements in performance come at price in terms of power dissipation. In particular, the experiments show how the 7-bit represent the trade-off between performance and power consumption.

Fig. 5 shows the same data, referred to every considered algorithm in this paperwork (130nm). The trade-off between performance and power consumption comes out to be 8-bit, as global result coming from the full analysis of the proposed algorithms.

+	-	*	/	<<	>>
==	!=	<	>	≤	≥

Table 3. The overloaded operators in fixed point

Algorithm	Fixed-Point	Float-Point	IEEE 64-bit
Jakes noise filtered	0.024	0.064	0
Gaussian noise filtered	0.030	0.034	0
Zero Forcing Equalizer	0.246	0.220	0.246
Soft Viterbi	0.080	0.070	0.071
Soft Output Viterbi	0.340	0.320	0.300
Adaptive Equalizer	0.154	0.140	0.121
Sine Function	0.115	0.168	0
Fast Fourier Transform	0.96	0.97	1

Table 4. The global performance parameters, using fixed and the proposed floating point supports.

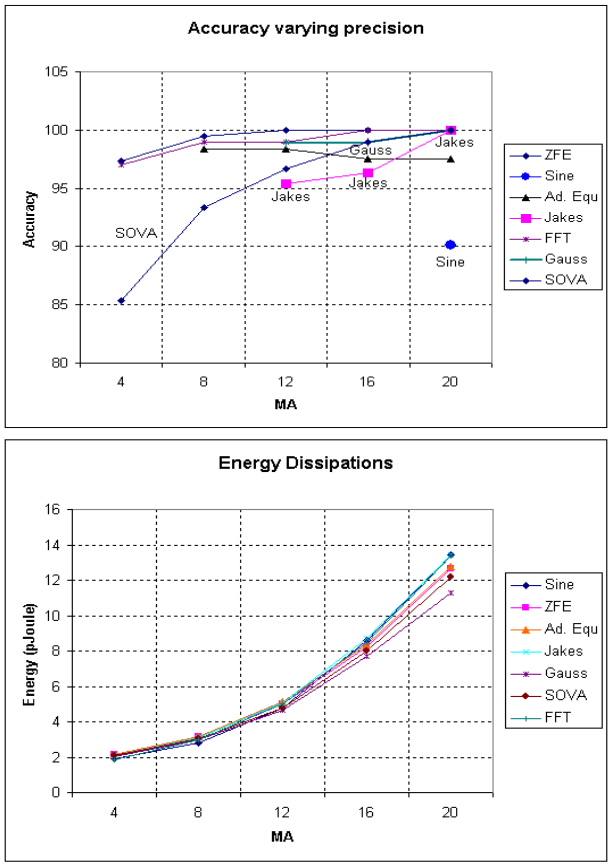

Figure 5. The performance power trade-off(full analysis)

Algorithm	IWL	Energy Adder	Energy Multi-plier
Jakes noise filtered	6	5.12 pJ	17.86 pJ
Gaussian noise filtered	4	5.72 pJ	18.52 pJ
Zero Forcing Equalizer	5	5.72 pJ	14.94 pJ
Soft Viterbi	8	5.72 pJ	15.27 pJ
Soft Output Viterbi	8	5.32 pJ	17.54 pJ
Adaptive Equalizer	5	5.60 pJ	16.00 pJ
Sine Function	1	5.38 pJ	19.38 pJ
Fast Fourier Transform	6	5.82 pJ	18.98 pJ

Table 5. The energy requirement for each proposed algorithms using the fixed-point support (180nm tech lib).

Algorithm	IWL	Energy Adder	Energy Multi-plier
Jakes noise filtered	6	1.12 pJ	9.78 pJ
Gaussian noise filtered	4	1.22 pJ	8.92 pJ
Zero Forcing Equalizer	5	1.23 pJ	7.83 pJ
Soft Viterbi	8	1.17 pJ	8.18 pJ
Soft Output Viterbi	8	1.16 pJ	8.72 pJ
Adaptive Equalizer	5	1.21 pJ	7.54 pJ
Sine Function	1	1.15 pJ	9.98 pJ
Fast Fourier Transform	6	1.25 pJ	10.61 pJ

Table 6. The energy requirement for each proposed algorithms using the fixed-point support (130nm tech lib).

5 Conclusion and Discussions

The work introduced and evaluated the principle of using a reduced floating-point format and truncated multiplier for a low-power FPU in application specific DSP cores. We considered in this work, together the reduced floating-point format and the power-aware architecture for the real arithmetic units.

Although the signal processing algorithms based on trigonometric functions are very power hungry, our results suggest the use of a reduced floating-point format when operating with a non-strongly-constrained global performance target. The floating-point math operations might be performed with dedicated power-aware hardware accelerators, operating with a reduced precision. The effects of gradual underflow, underflow and overflow detection and exceptions handling will be emulated in software. We demonstrated the effectiveness of this approach, comparing the power dissipations with respect to a standard integer unit for fixed-point operations. (Unfortunately, the truncated multiplier cannot be used in the standard integer multiplier due the exact performing of integer operations). The use of hardware accelerators represents a convenient low-power solution when the area is not a strong constraint and static power is made negligible by a low-leakage technology. These results confirm how nowadays the standardization of floating-point formats should consider precisions less than the single format included in the IEEE 754 standard.

Acknowledgment

We are grateful to Prof. Mauro Olivieri of La Sapienza University of Rome for his suggestions in improving this work.

References

[1] C. Inacio: *The DSP decision: fixed point or floating?* IEEE Spectrum September 1996.

[2] R. Brodersen and A. Chandrakasan and S. Sheng: *Low-Power Signal Processing Systems.* In IEEE Workshop on VLSI Signal Processing 1992

[3] *IEEE Standard for Binary Floating-Point Arithmetic* ANSI/IEEE Std. 754-1985 New York, The institute of Electrical and Electronics Engineers, Inc., August 12,1985.

Algorithm	IWL	Energy Adder	Energy Multiplier
Jakes noise filtered	6	0.46 pJ	5.58 pJ
Gaussian noise filtered	4	0.52 pJ	5.56 pJ
Zero Forcing Equalizer	5	0.53 pJ	4.98 pJ
Soft Viterbi	8	0.52 pJ	4.86 pJ
Soft Output Viterbi	8	0.49 pJ	5.36 pJ
Adaptive Equalizer	5	0.51 pJ	5.54 pJ
Sine Function	1	0.49 pJ	5.54 pJ
Fast Fourier Transform	6	0.53 pJ	5.64 pJ

Table 7. The energy requirement for each proposed algorithms using the fixed-point support (90nm tech lib).

Algorithm	MA	Energy Adder	Energy Multiplier
Jakes noise filtered	8	13.00 pJ	8.67 pJ
Gaussian noise filtered	6	8.64 pJ	5.62 pJ
Zero Forcing Equalizer	8	13.44 pJ	8.27 pJ
Soft Viterbi	6	9.17 pJ	5.00 pJ
Soft Output Viterbi	6	9.18 pJ	5.15 pJ
Adaptive Equalizer	6	9.55 pJ	5.42 pJ
Sine Function	12	16.85 pJ	12.78 pJ
Fast Fourier Transform	8	12.63 pJ	8.80 pJ

Table 8. The energy requirement for each proposed algorithms using the floating-point support (180nm tech lib).

Algorithm	MA	Energy Adder	Energy Multiplier
Jakes noise filtered	8	2.70 pJ	1.95 pJ
Gaussian noise filtered	6	2.10 pJ	1.30 pJ
Zero Forcing Equalizer	8	2.80 pJ	2.00 pJ
Soft Viterbi	6	2.35 pJ	1.25 pJ
Soft Output Viterbi	6	2.20 pJ	1.30 pJ
Adaptive Equalizer	6	2.40 pJ	1.40 pJ
Sine Function	12	3.95 pJ	4.50 pJ
Fast Fourier Transform	8	2.60 pJ	2.26 pJ

Table 9. The energy requirement for each proposed algorithms using the floating-point support (130nm tech lib).

Algorithm	MA	Energy Adder	Energy Multiplier
Jakes noise filtered	8	1.25 pJ	0.99 pJ
Gaussian noise filtered	6	0.98 pJ	0.68 pJ
Zero Forcing Equalizer	8	1.29 pJ	0.97 pJ
Soft Viterbi	6	1.07 pJ	0.64 pJ
Soft Output Viterbi	6	1.07 pJ	0.66 pJ
Adaptive Equalizer	6	1.10 pJ	0.68 pJ
Sine Function	12	1.66 pJ	1.86 pJ
Fast Fourier Transform	8	1.24 pJ	0.99 pJ

Table 10. The energy requirement for each proposed algorithms using the floating-point support (90nm tech lib).

[4] J. Tong, D. Nagle, R. Rutenbar :*Reducing Power by Optimizing the Necessary Precision/Range of Floating-Point Arithmetic*. IEEE. Trans. on VLSI Vol. 8, No. 3 (2000)

[5] M.J. Schulte, J.E. Stine and J.G. Jansen: *Reduced Power Dissipation through Truncated Multiplication.* In IEEE Alessandro Volta Memorial Workshop on Low-Power Design 1999.

[6] Y. Lim: *Single Precision Multiplier with Reduced Circuit Complexity for Signal Processing Application.* IEEE Transaction on Computers Vol.41 1992.

[7] M.J. Schulte and E. E. Swartzlander: *Truncated Multiplication with correction constant* IEEE Workshop on VLSI Signal Processing VI, 1993.

[8] G. Visalli and F. Pappalardo: *Low-Power Floating-Point Encoding for Signal Processing Applications.* IEEE Workshop on Signal Processing Systems August 2003.

[9] ST Microelectronics, *Corelib HCMOS9 Data Book and CB45000 Standard Cell Library Datasheet*, Aug. 2002.

[10] T. Aamodt and P. Chow: *Numerical Error Minimizing Floating-Point to Fixed-Point ANSI C Compilation.* In 1st Workshop on Media Processors and Digital Signal Processing, November 1999.

[11] T. Aamodt and P. Chow *Embedded ISA Support for Enhanced Floating-Point to Fixed-Point ANSI C Compilation* In 3rd International Conference on Compilers, Architectures, and Synthesis for Embedded Systems, November 2000

[12] W. Cammack and M. Paley: *FixPt A C++ Method for Development of Fixed Point Digital Signal Processing Algorithms.* In Proc. 27th Annual Hawaii Int. Conf. System Sciences, 1994.

[13] D. Goldberg: *What Every Computer Scientist Should Know About Floating-Point Arithmetic*. ACM Computer Surveys 1991.

[14] W.H. Press, B.P. Flannery, S.A. Teukolsky and W.T. Vetterling: *Numerical Recipes in C* Cambridge University Press 1995.