Optimizing DSP Circuits by a New Family of Arithmetic Operators

Javier Hormigo, and Julio Villalba
Dept. Computer Architecture
Universidad de Malaga
Malaga, Spain, E-29071
E-mail: fjhormigo@uma.es

***Abstract*—This paper presents a new family of arithmetic operators to optimize the implementation of circuits for digital signal processing. They are based on using a new fixed-point format which allows performing rounding to nearest as the same cost as truncation. Thanks to the use of rounding, the word-length optimization may improve significantly respect to using conventional units and truncation. That reduction means a simultaneous improvement of area, delay, and, consequently, power consumption. As an example, several FIR filters have been tested, and an area reduction up to 50% along with a speed improvement up to 42% has been obtained.**

***Index Terms*—Digital Signal Processing, fixed-point data-path, word-length optimization, real-number representation, round-to-nearest.**

I. Introduction

The selection of an adequate representation format for each variable on a digital signal processing circuit is one of the most important task to achieve an optimal trade-off between cost parameters (area, energy,. . .) and functionality constrains (delay, quantization error,. . .). For a cheaper implementation, fixed-point formats are usually preferred over floating-point ones, since the latter involve much more complicated operators. One of the main tasks needed to optimize fixed-point implementations of a DSP algorithm is the word-length optimization [1][2].

Said word-length optimization requires finding the word-length combination for the signal within the circuit which presents the minimum cost but, at the same time, it satisfies the required accuracy (i.e., maximum quantization rounding error) and dynamic range (i.e., it does not produce overflow). Using signals with less bit-width implies to use simpler operators which means less area, delay and power consumption. However, this simplification is obtained at the cost of introducing larger quantization error. This error is introduced when an intermediate or final value have to be rounded to meet the corresponding bit-width.

Several rounding modes could be used to perform said rounding, such as round-to-nearest which is the preferred mode for floating point format [3]. However, due to the relatively huge complexity incrementation required to implement these rounding modes under fixed-point format, a simple truncation is the rounding mode used generally in these cases. There is plenty of literature addressing worth-length optimization considering truncation as rounding mode, whereas the other modes has been practically discarded for years [4][5][6][7][8][9][10]. However, a recent work [1] has demonstrated the beneficial of using other rounding modes in certain designs. In the work presented in [1], the optimization of several representative kernels for digital signal processing were analyzed considering, not only truncation but round-to-nearest (biased and not biased version), along with the additional hardware involved for a single operation. They found that, despite the complexity introduced for each single rounding operation, the overall implementation area may be reduced by utilizing the optimal quantization mode combination instead of only truncation.

In this work, we present a new representation number format to implement rounding to nearest for fixed-point arithmetic at the similar cost of truncation. To operate numbers under said new format, new arithmetic units have to be designed. However a slight modification of the conventional circuits is enough to achieve this goal. Therefore, the expected performance gain should largely improve the results of the previous work in [1], since the hardware cost of implementing the same rounding is much lower.

This paper is organized as follows: Section II gives a brief analysis of different conventional rounding modes. In Section III, we present the new proposed fixed-point format, and we show how truncation produces round-to-nearest rounding when targeting a number under this representation. Section IV provides a guideline to implement a fixed-point DSP data-path using the proposed approach. In Section V, the results and comparison corresponding to the hardware implementation of several FIR filters under the new approach are presented as a proof of concept. Finally, Section VI provides the conclusion of this work.

II. Conventional rounding modes

The word-length optimization is a key tool to reduce the cost of fixed-point DSP implementations. Its use implies that each signal is reduced to the minimum feasible number of bits. Therefore, rounding is required almost after each operation and the type of rounding (rounding mode) used influences in two different ways: statistical characteristics of the rounding error generated and hardware complexity of its implementation. Generally, having better statistical characteristics requires greater hardware complexity.

The rounding mode associated to the simplest hardware implementation is truncation. Given two fixed-point formats, A and B, with n and m bits, respectively, being $n > m$, the rounding of a number represented using A to format B by truncation is performed just by taking the m Most Significant Bits (MSBs) of the original number. This operation is trivial, and it has no hardware cost, but its rounding error may be up to one Unit-in-the-Last-Place (ULP), i.e. the weight of the Least Significant Bit (LSB). Furthermore, it is very biased since it is always positive (for numbers with the same sign). Despite of those problems, it is the rounding mode generally used for fixed-point DSP hardware implementations.

Another rounding mode with a simple hardware implementation is von Neumann's rounding or jamming. In this case, the rounding is performed by selecting the $m-1$ MSBs of the original number and setting the LSB of the final number to one, if the discarded bits are not all zero. This operation produces an unbiased rounding since the error may be either positive or negative, but the magnitude of the error is still up to one ULP. Its implementation needs some logic to compute the sticky bit, but this is relatively simply. However, it is not commonly used, since indeed it duplicates the range of values represented for one number, as we will see later.

The round-to-nearest is the rounding mode which produces the lowest magnitude of the rounding error. Again, the m MSBs are selected but, in this case, if the MSB of the discarded bits is one, one ULP is added to get the result. Thus, the implementation of this rounding requires using an incrementer to perform said addition. This rounding mode produces a rounding error up to 0.5 ULPs and may be positive or negative. In spite of the fact that round-to-nearest produces the lowest rounding error, it is not generally used in fixed-point DSP application due to its relative complexity compared to fixed-point operations itself.

Fig. 1 summarizes how the values on a real line are rounded according to these three rounding modes. The Exactly Represented Numbers (ERNs) of the target format is represented, along with the range of inexact values represented by each ERN for each rounding mode. It is clearly seen that von Neumann's rounding duplicates the range of values represented by an ERN and truncation is very biased. These provokes that the quantization rounding error produced by round-to-nearest mode was significantly lower. In [1], word-length optimization have been studied considering different combination of rounding modes, and it has been demonstrate that the utilization of round-to-nearest may reduce the overall hardware cost, despite of the individual cost increment of using said rounding mode. This cost reduction is achieved because the lower rounding error produces a greater reduction of bit-width which overcomes the cost of implementing the rounding.

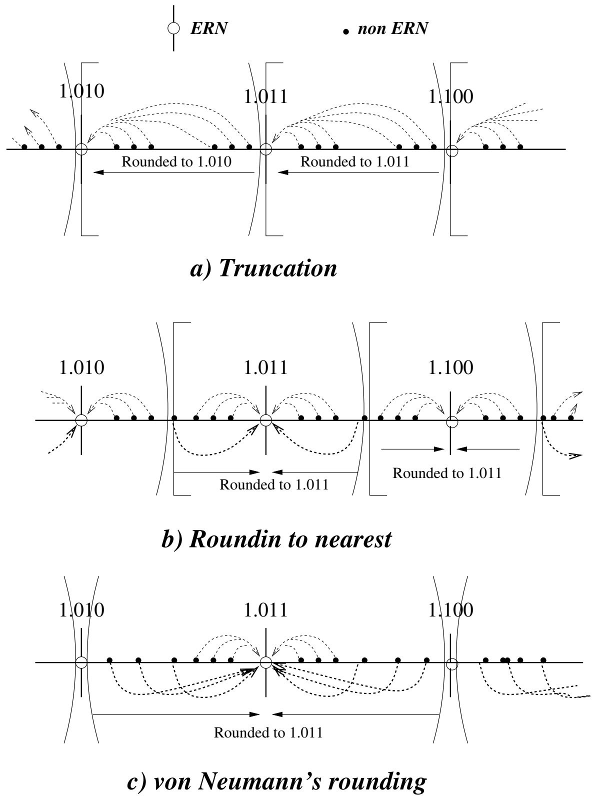

Fig. 1. Some conventional rounding modes

III. New proposed fixed-point format

In this section, we present a new binary fixed-point format which allows performing round-to-nearest in the same way (and cost) as truncation. Thus, utilizing this proposed format should produce a reduction on the word-length similar to the one achieved when round-to-nearest is used under conventional format, but the overall reduction should be greater since the rounding is implemented much more easily.

Based on the same idea as von Neumann's rounding, instead of forcing the LSB to one when it is required, we define a new number representation format which includes an implicit LSB which is constant and equal to one. This new implicit bit provokes that the ERNs represented by a bit vector under a conventional format were shifted by half-ULP when the same bit-vector represents a number under the new format. Fig. 2 shows an example of a three fractional bits fixed-point format. The ERNs under the proposed format are always on the middle point of two ERNs under its corresponding conventional format. The distance between consecutive ERNs is the same under both formats, i.e. one ULP. Thus, the precision of both formats is the same. Moreover, the number of bits required to represent both formats are also the same, since the new bit is implicit and it is not needed to storage or transmit it. Therefore, both conventional and the proposed formats have equivalent characteristics but their ERNs are different.

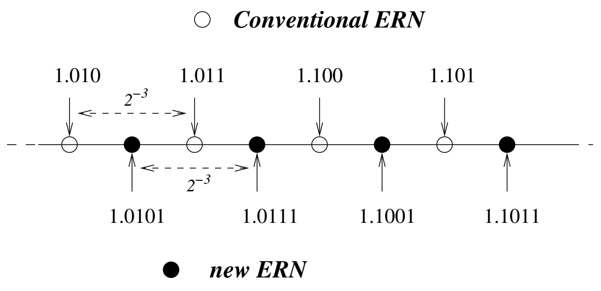

Fig. 2. ERNs for both conventional and proposed format

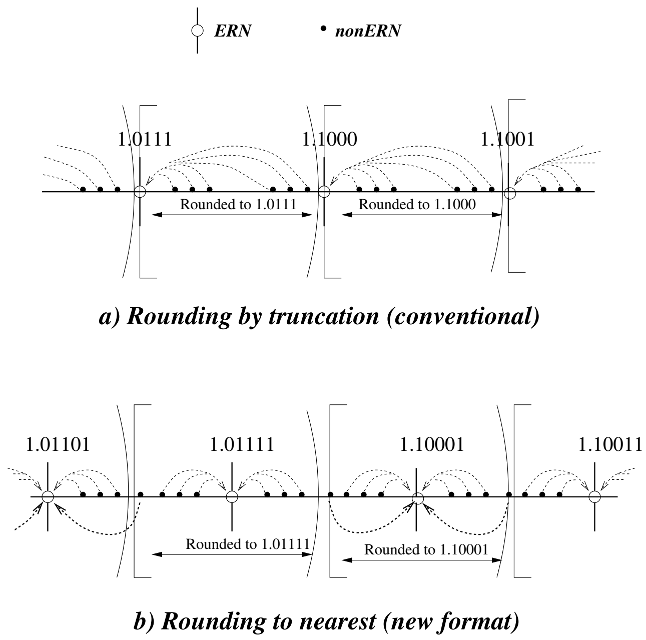

Fig. 3. Truncation for both conventional and proposed format

This new location of the ERNs produces that truncation (i.e., discarding the LSBs of a number to reduce the number of significant bits) to obtain a number under the new proposed format is actually a rounding to the nearest ERN. This fact is easily observed graphically as it is shown in Fig. 3. When only the 5 MSBs remains, the nonERN values selected for each bit-vector are the same under both the conventional and the proposed formats, but the ERN which represents those values are different. Given a range of nonERN values, the ERN representing said values under the conventional format always means an effective rounding down, whereas under the new format it always means an effective round-to-nearest. Therefore, using the proposed approach, rounding operation may produce the statistical characteristic of round-to-nearest but at the cost of the simple implementation of truncation.

IV. Fixed-point data-path implementation using the new format

In the previous section, we have seen how the use of a new format facilitates the hardware implementation of round-to-nearest rounding. However, the implementation of fixed-point DSP data-path requires using other arithmetic units which may be different under the new format. In this section, we study fixed-point arithmetic units for operands under the proposed format and conversion, since they are the key building-blocks to design a fixed-point data-path for DSP applications.

First, the input data may be introduced into the digital data-path. Ideally the input data should be converted directly from the real world to a digital number under the proposed format. In many applications, this requires to tune in the analog-to-digital converters to give their output number under the proposed format, which should not be a real problem.

If the input values are already digital numbers under a conventional format, a conversion is required. This conversion may be performed just by truncating those numbers to the amount of bits desired. This operation produces a round-to-nearest rounding, since we are targeting a number under the new proposed format. But, this initial conversion may introduce additional rounding errors, due to double rounding problems. On the contrary, for this last case, another option is to operate using conventional formats as long as a rounding operation is not required. At the point a rounding is needed, this rounding is performed by truncation and it generates a number under the new proposed format. In this way, the amount of rounding error introduced is minimized.

Once there are numbers represented under the proposed format, the design of arithmetic units to operate with these numbers is required. Taking into account the definition of the new format, it is easily seen that the conversion of a number under the new format to a conventional one could be easily obtained by explicitly extending its bit-vector with its implicit LSD. Therefore, arithmetic units to operate numbers under the proposed format could be design just by extend each input operand by a constant LSB set to one. After this trivial conversion, the extended operands may be operated using a conventional arithmetic logic. The conversion of the result back to the input format could be performed also trivially by truncating it to the desired number of bits. An arithmetic unit designed in this way produces the results rounded to the nearest. This is a general procedure which is valid for any operation. Nevertheless, taking into account that this new LSB is a constant value, a more optimal design could be obtained by studying each operation in detail.

On the other hand, the equivalent arithmetic unit for conventional format has input operands with one bit less, but it requires a rounding unit at the end to perform the round-to-nearest of the result. Therefore, each concrete arithmetic operation has to be studied particularly to determine whether the new arithmetic units are most costly than its equivalent conventional one. However, although one particular operation may result less efficient under the proposed format, the important matter is whether or not, the overall efficiency of the data-path is improved under the new format.

Regarding the computed result, again, ideally the DSP circuit should deliver the obtained results under the new proposed format. However, sometimes a conversion to conventional format may be required. Similarly to the arithmetic unit, a simple conversion could be performed by explicitly extending its bit-vector with its implicit LSD. Then, if more bits of precision are required for the final result, a zero extension is carried out. On the other way, if fewer bits are required, then a conventional rounding to get the desired number of bits is performed.

V. Results and comparison

To test the performance of the proposed formats and circuits, several FIR filter examples have been designed, word–length optimized and implemented on FPGA. This process has been performed using both the new proposed arithmetic with round-to-nearest and the conventional one with truncation. The main results of these FPGA implementations have given in this section.

Let us give more details about the process we have followed. First, using Matlab, we have computed the coefficients of several low–pass and high–pass FIR filters for different number of taps. Then, the floating-point version of each filter, considering the direct-form structure, has been optimized for fixed-point computation using "Floating-Point to Fixed-Point Transformation Toolbox" [11]. For a given error threshold, this Matlab toolbox optimizes the worth–length combination of the given DSP data-path to minimize the estimated area cost of the circuit when implemented it on FPGA. It uses a well-known gradient based method to achieve this goal, although this method does not guarantee to obtain the global minimum.

For each filter, the optimum combination word-length has been computed for both versions, the one using our proposed arithmetic units and round-to-nearest as rounding mode and standard arithmetic circuits with truncation. The range of sizes utilized to look for the optimum word-length combination goes from 1 bit to 16 bits for each signal within the filter data-path. These signals include all coefficients of the filter and all input, output and internal signals.

In Table I, the results of this word-length optimization are summarized. We have grouped together all signal word-lengths within the same filter and then, their statistical parameters have been calculated. Table I shows the quartile values for the word-length (in bits) of all signals within each filter. For all cases, it is clearly observed a significant reduction of the number of bits when round-to-nearest is used instead of truncation. For example, according to Table I, in the High-pass filter with 5 taps, a 25% of signals have 4 bits or less for both versions. But, for the conventional approach, half of the signals have less than 10 bits whereas they have 6 bits or less for the proposed version. Similarly, for the conventional approach, a 25 % of signals has 13 or more bits but they have only more than 7 bits for the proposed version. Looking at the Table I, it is seen that the amount of reduction depends on the concrete filter observed, and apparently it does not show any pattern.

TABLE I
Statistical parameters of word-length optimization

Filtro	Q1		Q2	(MED)	Q3	
	New	std	New	std	New	std
LFIR3	2	4.5	5.5	7	7	11
HFIR3	4	3.5	5	6	7	8.75
LFIR5	3	4	6	7	6.75	12.75
HFIR5	4	4	6	9.5	7.75	13
LFIR8	2.25	4	5	7	7	10
HFIR8	3.25	4	5	7	8	10
LFIR10	2	4	4.5	6	6	8.75
HFIR10	2.25	4	5	9	7	12

Using these optimum combinations of different signal sizes calculated for each filter and rounding method, the corresponding VHDL circuits have been designed. Then, they have been synthesized for a XILINX Virtex-6 family FPGA, using ISE v14.3 software. The area and delay results obtained for all different filters are shown in Table II. We should clarify several important points about these FPGA implementations. To isolate the delay caused by the filter itself by the one caused by communications, all input and output signal have been registered. In contrast, the data-path of the filters is fully combinational (except the delays lines, for the input signal that have to be stored). Thus, the delay presented in Table II refers to whole computation time needed to compute a new output value since a new input sample arrives.

On the other hand, the embedded multipliers presented on the FPGAs have not been utilized to implement multiplications. In other case, it may difficult a precise comparison, especially due to the small sizes of multipliers required. Thus, regular slice logic has been used to implement the multipliers. This implementation may be in two different forms: standard multiplier o multiplier to a constant. The synthesis software only uses dedicated implementation of multiplication to a constant for unsigned operators. Then, a conversion from signed to unsigned number is required to take advantage of the optimization due to constant coefficients. This conversion may introduce a cost which overcomes the advantage of using multipliers to a constant. Thus, the overall result depends on the constant value itself and the sizes of the operands. For this reason, we have implemented the two versions, i.e. using standard multipliers logic and using canonical signed digit multiplier with conversion. Since the conversion between unsigned and signed numbers is very easy for the proposed format, the multiplier to a constant version is always better for our proposal. However, it depends on the case for the conventional version and, thus both results are shown. Between these two versions, the one with the minimum area and delay has been taken for comparing it to the proposed

TABLE II
IMPLEMENTATION RESULTS FOR VIRTEX-6 FPGA

Filter	Area (LUTs)				
	New	std(KCM)	std	min(std)	%
LFIR3	82	184	129	129	57
HFIR3	75	207	116	116	55
LFIR5	123	240	381	240	95
HFIR5	149	383	247	247	66
LFIR8	168	269	431	269	60
HFIR8	178	219	369	219	23
LFIR10	153	488	312	312	104
HFIR10	233	569	299	299	28

Filter	Delay (ns)				
	New	std(KCM)	std	min (std)	speedup
LFIR3	4.694	6.695	6.648	6.648	1.42
HFIR3	5.953	8.752	6.715	6.715	1.13
LFIR5	7.728	8.67	9.859	8.67	1.12
HFIR5	7.716	10.64	8.597	8.597	1.11
LFIR8	8.592	11.692	11.7	11.692	1.36
HFIR8	8.698	9.693	12.268	9.693	1.11
LFIR10	10.153	12.235	11.546	11.546	1.14
HFIR10	10.888	12.693	11.598	11.598	1.07

approach.

Regarding to Table II, it is observed that the proposed implementation clearly outperforms the one implemented in a conventional way, especially when a occupied area is considered. The area increment, when using the conventional implementation instead of the proposed one, ranges from 23 % to 104% (more than double), and the mean for these examples is 61%. While, they reduce the cost in area, the same circuits also improve speed. The speedup for the proposed implementation respect to the conventional one ranges from 1.07 to 1.42, and the mean is 1.18. Therefore, for these examples, the average FIR filter using our approach are 18% faster and 38% smaller than the conventional approach.

According to the work in [1], the maximum cost reduction achievable for 16-tap FIR filters using round-to-nearest rounding with conventional formats instead of truncation were up to 8.9% with a mean of 5.5%. Thus, as we expected, the improvement achieved using our approach is largely greater than the one reported in [1].

Summarizing, although only a few examples have been studied and then we should not extrapolate these results as sure, taking into account that all these results obtained goes in the same direction, we could say that the use of the proposed arithmetic circuits and format may produce, in general (at least for FIR filter implementation), a very significant reduction on the area occupied along with a moderate speed improvement. However, each particular case should be studied thoroughly.

VI. CONCLUSION

A new family of arithmetic operators to optimize the implementation of circuits for digital signal processing has been presented. They are based on the use of a new fixed-point format for real numbers, which allow performing round to the nearest in a very simple way. Using rounding instead of truncation, as it is generally do it, allows reducing the overall bit-width of the DSP circuit when performing word-length optimization for a given error threshold. In contrast, arithmetic operations using the new format may require extending the operands with a constant extra-bit. However, the benefits of using the new format clearly outperform the cost of this extra-bit, as it has been demonstrate using FIR filter examples. In average, an area reduction of 38% and speed increase of 18% is achieved simultaneously. The general ideas presented in this paper may be also applied to floating-point numbers. A few patent applications have been filed for different issues regarding to the circuits to operate under the new format.

ACKNOWLEDGMENT

This work was supported in part by the Ministry of Education and Science of Spain under contract TIN2010-16144.

REFERENCES

[1] D. Menard, D. Novo, R. Rocher, F. Catthoor, and O. Sentiey, "Quantization mode opportunities in fixed-point system design," *European Signal Processing Conference*, pp. 542–546, 2010.

[2] W. Sung and K.-I. Kum, "Simulation-based word-length optimization method for fixed-point digital signal processing systems," *Signal Processing, IEEE Transactions on*, vol. 43, no. 12, pp. 3087–3090, Dec 1995.

[3] M. D. Ercegovac and T. Lang, *Digital Arithmetic.* Morgan Kaufmann Publishers, 2004.

[4] Y. C. Lim, Y. J. Yu, K. L. Teo, and T. Saramaki, "FRM-based FIR filters with optimum finite word-length performance," *Signal Processing, IEEE Transactions on*, vol. 55, no. 6, pp. 2914–2924, June 2007.

[5] G. Caffarena, G. Constantinides, P. Cheung, C. Carreras, and O. Nieto-Taladriz, "Optimal combined word-length allocation and architectural synthesis of digital signal processing circuits," *Circuits and Systems II: Express Briefs, IEEE Transactions on*, vol. 53, no. 5, pp. 339–343, May 2006.

[6] O. Sarbishei, K. Radecka, and Z. Zilic, "Analytical optimization of bit-widths in fixed-point LTI systems," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 31, no. 3, pp. 343–355, March 2012.

[7] S. Kim and W. Sung, "Fixed-point error analysis and word length optimization of 8x8 IDCT architectures," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 8, no. 8, pp. 935–940, Dec 1998.

[8] K.-I. Kum and W. Sung, "Combined word-length optimization and high-level synthesis of digital signal processing systems," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 20, no. 8, pp. 921–930, Aug 2001.

[9] O. Sarbishei and K. Radecka, "On the fixed-point accuracy analysis and optimization of polynomial specifications," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 32, no. 6, pp. 831–844, June 2013.

[10] S. Vakili, J. Langlois, and G. Bois, "Enhanced precision analysis for accuracy-aware bit-width optimization using affine arithmetic," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 32, no. 12, pp. 1853–1865, Dec 2013.

[11] K. Han and B. L. Evans. (2006) Floating-point to fixed-point transformation toolbox. [Online]. Available: http://users.ece.utexas.edu/ bevans/projects/wordlength/converter/