Optimizing DSP Circuits by a New Family of Arithmetic Operators

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

***Abstract*—A new family of arithmetic operators to optimize the implementation of circuits for digital signal processing is presented. Thanks to use of a new technique which reduces the quantification errors, the proposed operators may decrease significantly the size of the circuits required for most applications. That means a simultaneous reduction of area, delay and power consumption.**

***Index Terms*—Real-number representation,**

I. Introduction

The selection of the 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 among cost parameters (area, energy,...) and functionality constrains (delay, quantization error,...). For cheaper implementation, fixed-point formats are usually preferred over floating-point ones, since the latter involve much more complicated operators. Optimization of fixed-point implementations requires to find the word-length combination 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). Signals using less bit-width implies simpler operator 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 [1]. However, due to the huge complexity incrementation, in relative terms, required to implement these rounding modes in fixed point format, a simple truncation is the rounding mode generally used in this 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[2][3][4][5]. However, a recent work [6] has demonstrated the beneficial of using other rounding modes in certain designs. In the work presented in [6], 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 my be reduced up to 46% by utilizing the optimal quantization mode combination instead of only truncation.

In this work we present a new technique to implement round to nearest for fixed point arithmetic at the similar cost of truncation. Therefore the expected performance gain should largely beat the results of this previous publication[6].

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, 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 both positive or negative, but the magnitude of the error is still up to one ULP. Its implementation need some logic to compute the sticky bit, but this is relatively

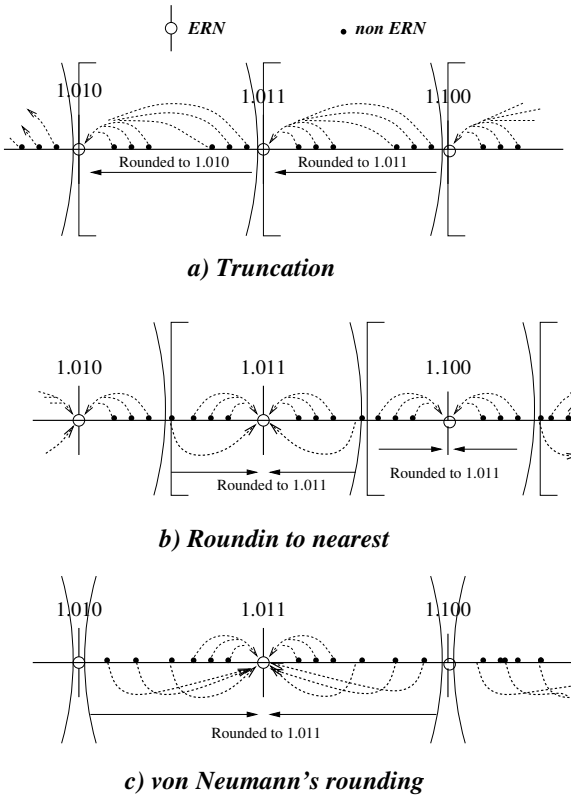


Fig. 1. Some conventional rounding modes

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 to use 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. **??** 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 [6], 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

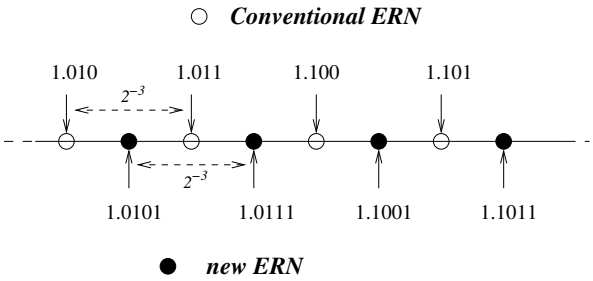


Fig. 2. ERNs for both conventional and proposed format

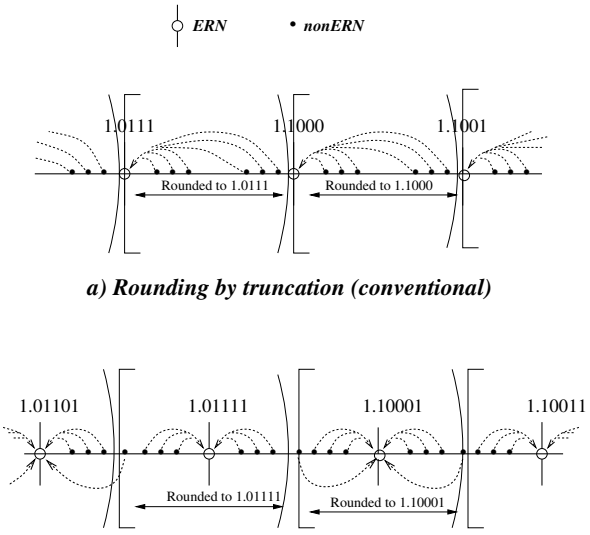
individual cost increment of using said rounding mode. This cost reduction is achieved because the lower rounding error produces a greater reduction of bit-width which overcome the cost of implementing the rounding.

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. **??** 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 are the same under both formats, i.e. one ULP. Thus, the precision of both format 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 need to storage or transmit it. Therefore, both conventional and the proposed formats have equivalent characteristics but their ERNs are different.

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 inexact 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 inexact values, the ERN representing said values under the conventional format always means an effective rounding down, whereas under the new format it always



a) Rounding by truncation (conventional)

b) Rounding to nearest (new format)

Fig. 3. Truncation for both conventional and proposed format

means an effective round-to-nearest. Therefore, using the proposed format, the rounding may be performed with the statistical characteristic of round-to-nearest and 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 to use 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 requires just truncating to the amount of bits desired, since this operation produces a round-to-nearest rounding when targeting a number under the 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. In the point a rounding is needed, this rounding is performed by truncation and generating 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 extending explicitly 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 operate using a conventional logic. The conversion of the result back to the input format also could be performed 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 formats, 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, to determine whether the new arithmetic units are most costly than its equivalent conventional, each concrete arithmetic operation has to be studied particularly. Next, we analyze addition and multiplication since they are the mos important arithmetic operation for DSP. 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.

V. Results and comparison

To test the performance of the proposed formats and circuits, several FIR filter examples have been design, word–length optimized and implemented on FPGA. This process has been performed using both the new proposed arithmetic with round-to-nearest and a standard 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, we have computed the coefficients of several low–pass and high–pass FIR filters for different number of taps using Matlab. Then, the floating-point version of the filters, considering the direct-form structure, has been optimized for fixed-point computation using ”Floating-Point to Fixed-Point Transformation Toolbox” [7]. 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 methods to achieve this goal, although this methods 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

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

TABLE I
STATISTICAL PARAMETERS OF WORD-LENGTH OPTIMIZATION

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

TABLE II
IMPLEMENTATION RESULTS

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 signal 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, some 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, half of the signals have less than 10 bits whereas they have 6 bits or less for the proposed version. Similarly, 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. it is seen that the amount of reduction depends on the concrete filter observed, and apparently it does not show any pattern.

Using these optimum combinations of different signal sizes calculated for each filter and rounding method, the corresponding VHDL circuits have been design. Then, they have been synthesized for a XILINX Virtex-6 family FPGA, using ISE v14.3 software. The results of area and delay obtained for all different filters are shown in table II. We should clarify several important points about these FPGA implementations. To isolate the delay of the filter from communication, all input and output signal have been registered. In contrast, the data-path of the filters is combinational (except the delays lines, for input signal that have to be stored). Thus, the delay presented refers to the one between the input signal and the output signal.

On the other hand, the embedded multipliers presented on the FPGAs have not been utilized to implement multiplications, since their use may difficult a precise comparison, specially due to the small sizes of multipliers required. Thus, regular slice logic has been used to implement the multipliers, but this implementation may be in to different form: standard multiplier o multiplier to a constant. The synthesis software only use 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 introduced a cost which overcome the advantage of use multipliers to a constant. Thus, the overall results 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 arithmetic, the multiplier to a constant version is always better for our proposal. However, it depends on the case for the truncation version and both results are shown.

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 to add and constant extra-bit to the operands. 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.

REFERENCES

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

[2] 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.

[3] 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.

[4] 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.

[5] 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.

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

[7] 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/