Efficient Implementations of Reduced Precision Redundancy (RPR) Multiply and Accumulate (MAC)

Ke Chen, Linbin Chen, Pedro Reviriego, and Fabrizio Lombardi

Abstract— Multiply and Accumulate (MAC) is one of the most common operations in modern computing systems. It is for example used in matrix multiplication and in new computational environments such as those executed on neural networks for deep machine learning. MAC is also used in critical systems that must operate reliably such as object recognition for vehicles. Therefore, MAC implementations must be able to cope with errors that may be caused for example by radiation. A common scheme to deal with soft errors in arithmetic circuits is the use of Reduced Precision Redundancy (RPR). RPR instead of replicating the entire circuit, uses reduced precision copies which significantly reduce the overhead while still being able to correct the largest errors. This paper considers the implementation of RPR Multiply and Accumulate circuits. First, it is shown that the properties of signed integer multiplication (two´s complement format) can be used to make RPR more efficient. Then its principles are extended to the MAC operation by proposing RPR implementations that improve the error correction capabilities with a limited impact on circuit overhead. The proposed schemes have been implemented and tested. The results show that they can significantly reduce the Mean Square Error (MSE) at the output when the circuit is affected by a soft error and the implementation overhead of the proposed schemes is extremely low.

Index Terms— Soft errors, Multiply and Accumulate, Reduced Precision Redundancy

1 Introduction

MULTIPLY and Accumulate (MAC, also known as multiply and add) is one of the most commonly used operations in computing systems. It is for example, the key element of matrix multiplication [1] [2] and it is also used in new algorithms like Deep Learning Neural Networks (DNNs) [3]. In [4] stochastic computation has been proposed for implementing artificial neural networks with reduced hardware and power consumption, while retaining in most cases the required accuracy and still improving the processing speed. MAC is extensively used in digital signal processing [5],[6] and its fused implementation has been extensively analyzed as part of the floating point IEEE 754 Standard [7].

Matrix multiplication (MM) has a high complexity due to the large number of required multiplication and addition operations; however, its computational regularity can be exploited using a systolic array. Different approximate schemes have been presented in [8]. Inexact full adder cells are utilized in a MAC-based processing element (PE) for the Baugh-Wooley multiplier and/or the final adder as circuits implementing the two computational steps required for matrix multiplication using an integer format. In [9], an approximate design has been proposed for the Discrete Cosine Transform (DCT) using signed integers in the transformation matrix. The DCT processing is used for image convolution on a pixel basis of an image by employing multiple approximate circuit and algorithm-level techniques such as truncation and voltage scaling. The integer version of MAC is widely used for example, in image processing.

In many applications, MAC operations are part of a system that needs to operate reliably, like for example cloud computing infrastructure or vehicle security systems [3]. In those cases, the implementations must be able to detect and correct errors to ensure that they do not cause any major failure. There are many types of error that can affect electronic circuits including manufacturing defects, ageing, electromagnetic disturbances and radiation induced soft errors [10]. In particular, soft errors caused by radiation are challenging because they cannot be avoided or detected in advance as it is the case with manufacturing defects or ageing [11]. A soft error typically changes the value of a circuit node and thus can produce an incorrect result for the MAC operation.

There are many techniques to protect a design against soft errors. Probably, the most common one is the use of replication in the form of Dual Modular Redundancy (DMR) to detect errors or Triple Modular Redundancy (TMR) to correct them [11]. A major issue of modular redundancy is its implementation cost. For example, TMR requires circuit area and power of more than three times that of the unprotected design. TMR has also been utilized in a stochastic computational multi-layer perceptron design for neural networks by relying on backward propagation. In [4], a TMR-based probability estimator and divider are employed. By utilizing a TMR voting structure in the processing element and divider, the error due to stochastic fluctuations in the binary search process for training purposes is significantly

• *K. Chen, L. Chen and F. Lombardi are with Northeastern University, Dept. of ECE, Boston Ma 02115, USA (email: chen.ke1@husky.neu.edu, linbin.c@gmail.com, lombardi@ece.neu.edu).*
• *P. Reviriego was with Universidad Antonio de Nebrija at the time of writing this paper. He is currently with Universidad Carlos III de Madrid, Av. Universidad 30, Leganes, Madrid, Spain (email: revirieg@it.uc3m.es).*

reduced. As a result, the latency and energy consumption are also reduced. The analysis in [4] has confirmed that TMR will be required also for machine learning and the new computational systems required for its execution.

For some arithmetic circuits, their properties can be used to reduce the protection cost. In the case of multipliers, for example residue codes [12] and parity prediction [13] have been used to provide an efficient protection. In both cases, the protection targets error detection, by checking the modulo of the output in the first case or its parity in the second case. To implement correction, two copies of the multiplier would be needed using the residue or parity checking to identify the correct one when they are different. This means that a cost that is still larger than twice the unprotected multiplier is needed.

An alternative to complete redundancy such as triplication or duplication is to use Reduced Precision Redundancy (RPR) [14],[15]. RPR uses a full precision version of the circuit and two reduced precision ones with an implementation of a voting logic to correct errors. This significantly lowers the overhead because the reduced precision copies are significantly smaller than the full precision implementation. The price to pay is that small errors cannot be corrected because the reduced precision may induce small differences in output values that are difficult to discriminate. This however is not a problem for many applications in which most failures are due to large errors. In fact, it has been recently shown that the same reasoning can also apply to new circuits like for example Deep Learning Neural Networks (DNNs) [3] for training purposes.

Another issue when RPR is used for simple structures is that the voting logic is significantly more complex than for TMR [16]. This occurs because the difference between the full precision and reduced precision copies needs to be computed and compared to a threshold. Therefore, a subtraction and comparison are needed. This logic may be as complex as some operations, for example addition. The optimization of RPR for addition has been explored in [17] in which it was shown that the voting logic could be simplified for adders.

In this work, we consider different RPR implementations of integer Multiply and Accumulate/Add (MAC) in two´s complement (signed) format. We first analyze signed multiplication to show that it is possible to optimize the RPR implementation. This is done by exploiting the dependence of the truncation error on the sign of the operands. In a second part, this technique is extended to the MAC operation using different schemes. To evaluate the benefits of the proposed schemes, both multipliers and MAC operations have been implemented and tested. The results show that the proposed schemes can improve the error correction capabilities of RPR for a given truncation, while requiring a low overhead in terms of implementation. Alternatively, the schemes could be used to increase the truncation and thus reduce the implementation cost while preserving the error detection capability.

The rest of the paper is organized as follows: section 2 provides a brief overview of RPR. The proposed schemes for multiplication and MAC are presented in section 3 and evaluated in section 4. The application of the proposed schemes to DCT processing is presented in section 5. The paper ends with the conclusion and future work in section 6.

2 REDUCED PRECISION REDUNDANCY

This section gives a brief overview of reduced precision redundancy in multipliers and multiply accumulate operations. For more details on this topic, the reader is referred to [15],[16].

2.1 Multiplication

A traditional Implementation of a Reduced Precision Redundant (RPR) multiplier is shown in Figure 1. It can be seen, that in addition to the original full precision multiplier, two redundant copies with reduced precision are used [15]. The simplest way to reduce the precision is to truncate the inputs by removing the least significant bits. The logic to detect and correct errors computes the difference between the full precision output and one of the reduced precision copies. If that difference is larger than a given threshold, it is assumed that an error has occurred.

To locate the error, the outputs of the two reduced precision copies are compared. If they are different, an error has occurred in the redundant copies. However, if they are equal, the error has affected the full precision multiplier.

Let us consider a signed integer multiplier with *N*-bit inputs and reduced precision ones with *N-K* bit inputs. Then, the output of the main multiplier can take values in the range between $-2^{(2N-1)}$ and $2^{(2N-1)}-1$. If the operands are a and b, the output of the reduced precision multipliers can be expressed as:

$$a_r \cdot b_r = (a + a_t) \cdot (b + b_t) = a \cdot b + a_t \cdot b + b_t \cdot a + a_t \cdot b_t$$

where a_t and b_t are the errors introduced when K bits are truncated that can take values in the range -2^K+1 to 0. An interesting observation is that in two´s complement format, truncation can only produce negative errors.

The difference between the outputs of the full and reduced precision multipliers is given by:

$$a \cdot b - a_r \cdot b_r = -(a_t \cdot b + b_t \cdot a + a_t \cdot b_t)$$

Since both a_t and b_t are negative, the largest errors (in magnitude) occur when a and b have the same sign and are large (the maximum values in magnitude are -2^{N-1} and $2^{N-1}-1$). For negative values, the largest error occurs when $a = b = -2^{N-1}+2^K-1$, and its magnitude can be computed to be:

$$2^{N+K} - 2^N - 2^{2K} + 2^{K+1} - 1$$

which can be approximated by 2^{N+K} that it is also an upper bound.

Similarly, for positive values of a and b the largest error in magnitude would be:

$$2^{N+K} - 2^N - 2^{2K} + 1$$

which can also be approximated also by 2^{N+K} that is also an upper bound. Therefore, 2^{N+K} can be used as the threshold to determine if a soft error has affected the full precision multiplier for practical truncation levels.

An alternative approach to determine the threshold to use is to simulate the RPR multiplier in an error free condition and measure for different threshold values the percentage of times that the RP output is selected. This is typically known as the inexact ratio. Then the lowest value that achieves an inexact ratio of zero can be selected. This will be further discussed in the evaluation section.

The implementation of the RPR multiplier just discussed, does not take into account the dependence of the difference between the full and reduced precision multipliers on the input values. However, this feature could be exploited to implement a more efficient RPR scheme as done for adders in [17]. In the next sections, some novel schemes that exploit this feature, are presented.

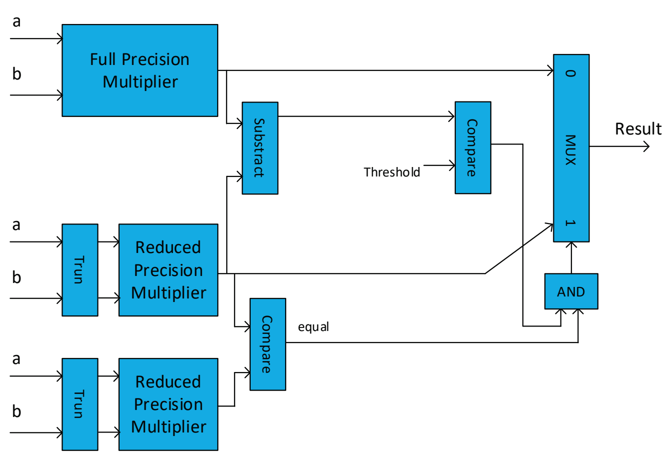

Figure 1. Traditional Reduced Precision Redundancy (RPR) for a Multiplier

2.2 Multiply and Accumulate (MAC)

For the MAC operation, the reduced precision redundancy can be implemented in different ways. One option could be to use a RPR multiplier followed by a RPR adder. This two-stage configuration is shown in Figure 2. An alternative is to perform a full precision MAC and two reduced precision ones and then implement the voting logic for the MAC operation. This one stage approach is illustrated in Figure 3. This second option reduces the voting logic and thus seems more attractive to reduce the implementation cost and also increase the speed of the MAC operation. In the remaining part of the paper, we focus on the one stage RPR implementation.

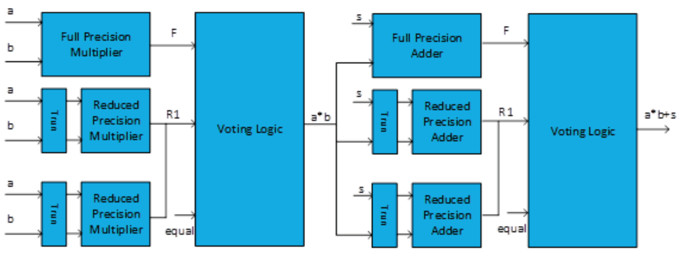

Figure 2. Two stages Reduced Precision Redundancy Multiply Accumulate

In the single stage implementation, the difference between the full precision and the reduced precision implementations can be larger because errors in the multiplication and addition may accumulate. In more detail, in our implementation, the output of the reduced precision multiplier has $2\cdot(N\text{-}K)\text{-}1$ bits and the full precision one has $2\cdot N\text{-}1$ bits. Therefore, $2\cdot K$ bits are truncated in *s* before the addition and introducing an error in the range $-2^{2\cdot K}+1$ to 0. It can be seen that the truncation error in the addition is smaller than in the multiplication where it could reach values close to 2^{N+K} as discussed before. In particular, if the worst case error in the addition is added to the worst case error in multiplication, we get the largest error in magnitude for the MAC that is:

$$\begin{aligned}(2^{2K}-1)+2^{N+K}-2^{N}-2^{2K}+2^{K+1}-1 \\ = 2^{N+K}-2^{N}+2^{K+1}-2\end{aligned}$$

which is still upper bounded by 2^{N+K}.

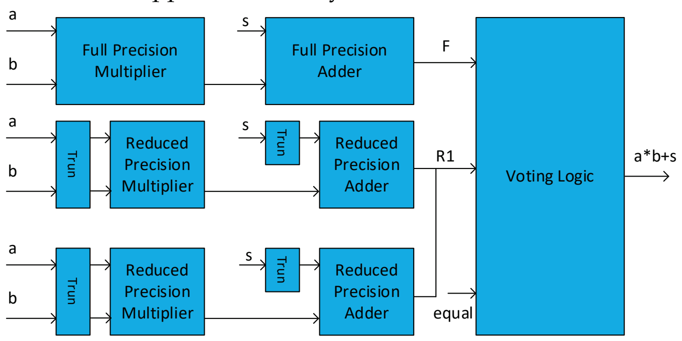

Figure 3. One stage Reduced Precision Redundancy Multiply Accumulate

3 PROPOSED SCHEMES

Next, the proposed reduced precision redundant multiplications and multiply and accumulate implementations are presented. In both cases, the relationship between the signs of the operands and the difference between the full and reduced precision operations are exploited to improve the accuracy of the voting logic.

3.1 Multiplication

The proposed scheme is based on the observation that for signed integers in two's complement format, the truncation error in a multiplication depends on the sign of the operands. This is clearly seen by remembering that truncation errors in two´s complement can only be negative and looking at the expression of the truncation error:

$$a\cdot b - a_r\cdot b_r = -(a_t\cdot b + b_t\cdot a + a_t\cdot b_t)$$

When the two operands (*a* and *b*) have the same sign, the errors introduced by truncation add. However, when the operands have different signs, errors will compensate. Therefore, the difference between the full precision and the reduced precision copies will be larger in the first case. This can be exploited by using different thresholds depending on the sign of the operands.

In more detail, the magnitude of the largest error when operands have different signs occurs when one operand takes the maximum (in magnitude) value and its truncation error is zero, while the truncation error for the other operand is maximum (to minimize the compensation). For a negative value, this would be:

$$(-2^{K}+1)\cdot(-2^{N-1}) \cong 2^{N+K-1}$$

and a similar error is obtained for a positive value. This is half the worst case error seen for the general case in the previous section. Therefore, when the signs are different, one option would be to use a threshold that is half the value of the general threshold.

A second improvement can be made when both operands have the same sign, as then, the difference between the full precision results and the reduced precision results can only have one sign. For example, if both are positive, the difference can only be positive. This means that we can check the sign to detect errors. For example, a negative difference when the operands are both positive is not possible in an error free situation and means that an error has affected the circuit. This enables the detection of errors that would go undetected in the traditional implementation.

The proposed RPR multiplier that incorporates both improvements is shown in Figure 4. It can be seen that the threshold is divided by two when the signs of the operands are different (the output of the XOR is one). In that case, the maximum error due to truncation is also smaller. Additionally, when the signs of the operands are the same and the difference between the full and reduced precision multipliers has also the opposite sign, an error is detected. This is because truncation should produce a difference of the same sign. Therefore, a difference of the opposite sign can only be due to an error. In those cases, if the two reduced precision copies generate the same result, then the error has occurred in the full precision multiplier and the output from the reduced precision ones should be used.

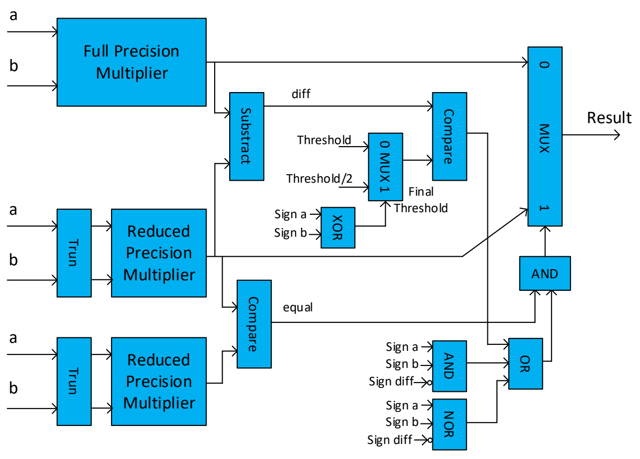

Figure 4. Proposed Reduced Precision Redundancy for a Multiplier

3.2 Multiplication and Accumulate

In the case of MAC, the design can be further improved by compensating the truncation error of the multiplication with the adder. The flowchart of the proposed optimizations for the MAC operation is shown in Figure 5. It can be seen that the sign of the operands is exploited to reduce the error both in the multiplication (left) and addition (right). This new scheme is illustrated in Figure 6; the carry-in signal of the adder is set based on the sign of the multiplication operands. So, when the operands are positive, truncation in the multiplication will introduce a positive error; thus setting the carry-in of the adder to one will introduce an error of opposite sign such that both errors will tend to compensate. This means that the worst case error for positive operands will be slightly smaller.

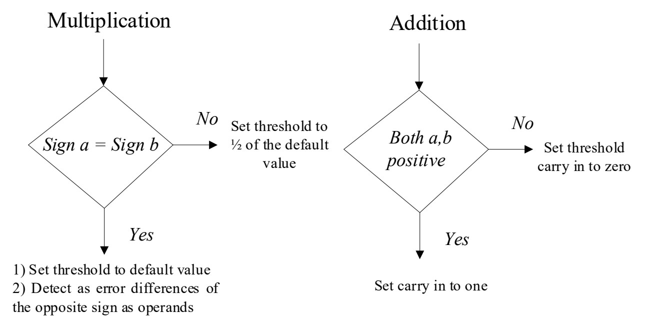
Multiplication
Sign a = Sign b
No
Set threshold to ½ of the default value
Yes
1) Set threshold to default value
2) Detect as error differences of the opposite sign as operands
Addition
Both a,b positive
No
Set threshold carry in to zero
Yes
Set carry in to one

Figure 5. Flow chart of the proposed optimizations for MAC. Multiplication (left), Addition (right)

4 Evaluation

To evaluate the proposed schemes, first the thresholds used for detecting errors have been verified by simulation. Then both the soft error tolerance and the implementation cost have been analyzed. To that end, the proposed schemes have been implemented in HDL and mapped to the FreePDK 45nm library.

4.1 Threshold Configuration

The thresholds used for error detection are those obtained in the analysis presented in section 2 and section 3 for the traditional and proposed implementations respectively. More precisely, 2^{N+K} in the first case and 2^{N+K-1} in the second case when operands have different signs and 2^{N+K} with sign checking when they have the same sign.

To check that those values are indeed correct, simulations have been done with different thresholds for voting and a variable referred to as the inexact ratio is measured. The inexact ratio is defined as the fraction of cases for which the output is taken from the reduced precision version in an error free condition. As an example, the inexact ratio versus the threshold value for $N=16$ is illustrated in Figure 7. It can be seen that there is one threshold value (referred to as critical) from which the inexact ratio becomes zero both for a traditional RPR multiplier and for the proposed scheme. In this case, this value is 2^{20}. The same simulations have been done for other word widths (N) and truncation levels (K) and the results are shown in Table 1. It can be seen that in all cases, the values obtained are given by 2^{N+K} which was the value obtained in the previous sections by analyzing the worst case error due to truncation (in the proposed scheme, that threshold is divided by two when operands have different signs). In the rest of the simulations, the threshold values given in Table 1 are used. The values obtained for the MAC operation were the same.

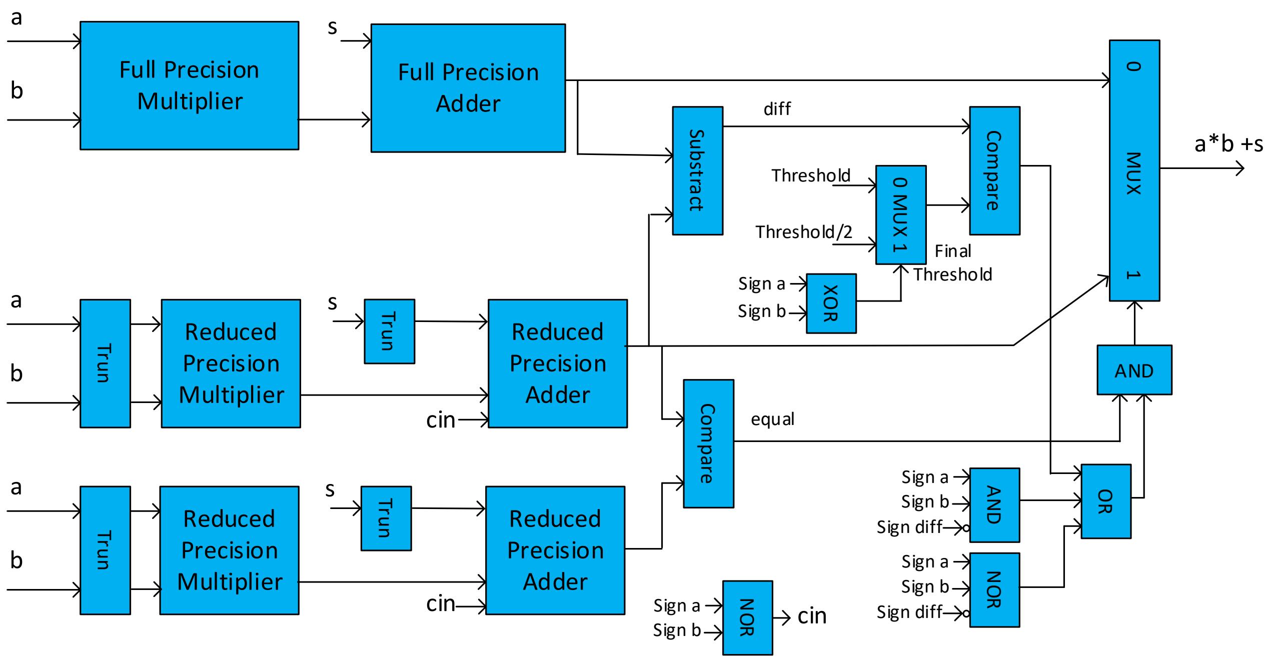

Figure 6. Proposed Reduced Precision Redundancy for Multiply and Accumulate

TABLE 1 CRITICAL THRESHOLD FOR DIFFERENT WORD WIDTHS AND TRUNCATION LEVELS

	K=2	*K*=4	*K*=8	*K*=12	*K*=16
N=16	2^{18}	2^{20}	2^{24}	-	-
N=24	2^{26}	2^{28}	2^{32}	2^{36}	-
N=32	2^{34}	2^{36}	2^{40}	2^{44}	2^{48}

4.2 Soft Error Tolerance

To evaluate the impact of soft errors on the traditional and the proposed RPR schemes, random errors that are uniformly distributed have been injected in both implementations. Then, the square error has been measured. The Mean Square Error (MSE) across all injections is used to compare the implementations. The MSE ratios between the proposed multiplier and the traditional RPR multiplier are presented in Figure 8 when the number of bits is 16, 24 and 32. It can be seen that the ratio is in the 0.4 to 0.6 range. Therefore, the proposed scheme can reduce the MSE by 40 to 60 per cent. The reductions are larger for smaller word widths and they also improve as truncation increases. The results for the MAC operation are shown in Figure 9. It can be observed that they are very similar to those of the multiplier. One reason for this phenomenon is that the probability of the multiplier experiencing a soft error is significantly larger than the probability of an adder experiencing a soft error due to the complexity difference between these two arithmetic circuits. Another reason is that the truncation in the multiplication tends to dominate the error.

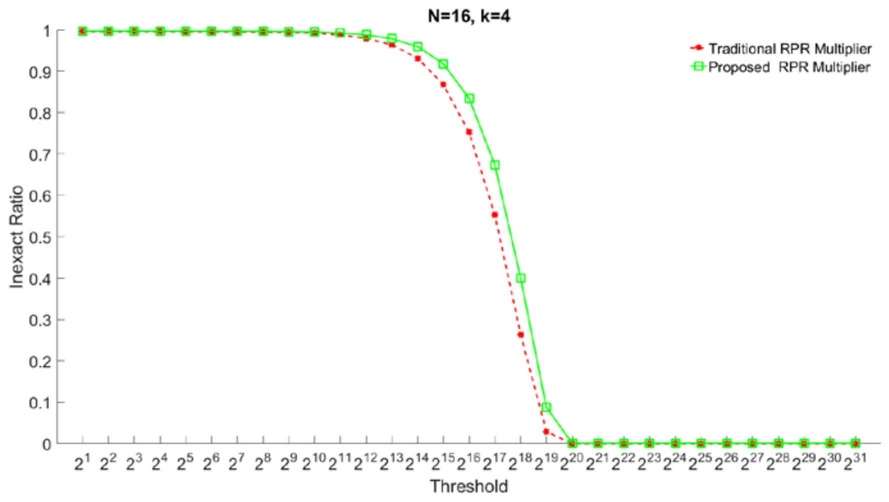

Figure 7. Inexact ratio versus threshold for a 16-bit multiplier protected with two redundant ones of 12 bits

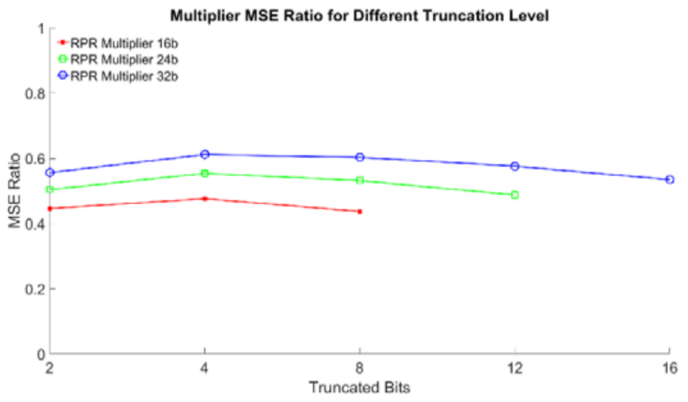

Figure 8. Ratio of the Mean Square Error (MSE) with soft error of a RPR multiplier protected with the proposed scheme versus a traditional RPR multiplier

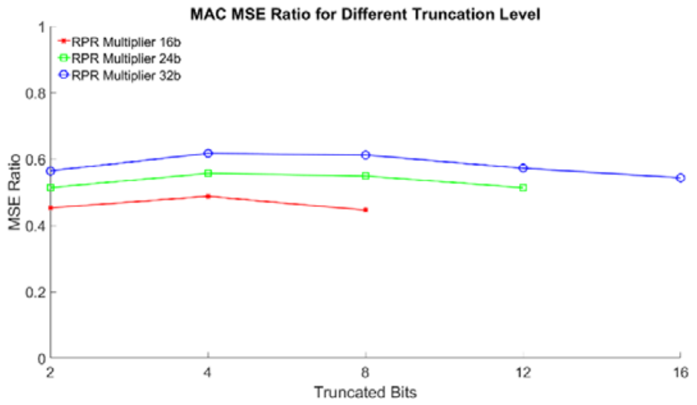

Figure 9. Ratio of the Mean Square Error (MSE) with soft error of a RPR MAC protected with the proposed scheme versus a traditional RPR MAC

4.3 Implementation cost

The area, delay, power and PDP of the RPR multipliers analyzed in the previous subsection are plotted in Figure 10 through Figure 13. It can be observed that the delay does not decrease significantly by increasing K. The proposed RPR multiplier design is slightly slower than the traditional RPR multiplier design. However, the power and area decrease rapidly with increasing K. The difference between the proposed and traditional RPR multiplier designs is negligible as expected, because the proposed RPR multiplier design just has a few more gates than the traditional RPR multiplier design. The PDP difference of the proposed and traditional RPR multiplier designs is also marginal. In summary, the overall delay and hardware cost of the proposed and traditional design are nearly the same.

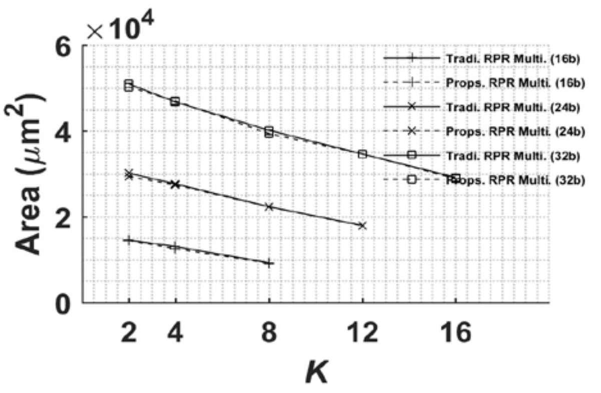

Figure 10. Area for RPR multipliers protected with different levels of truncation

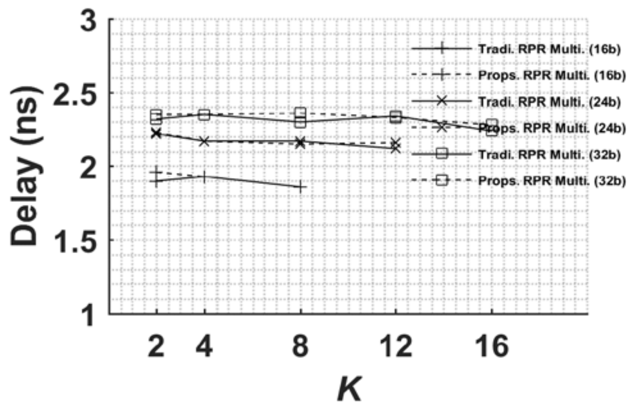

Figure 11. Delay for RPR multipliers protected with different levels of truncation

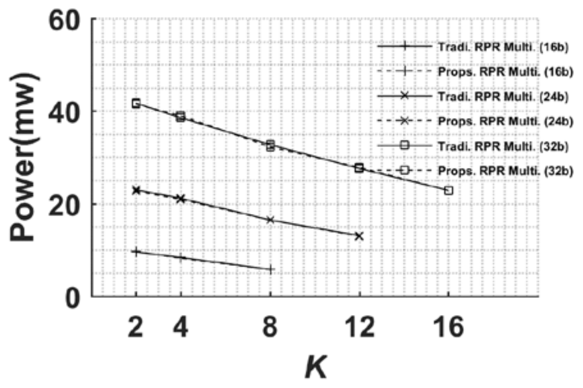

Figure 12. Power for RPR multipliers protected with different levels of truncation

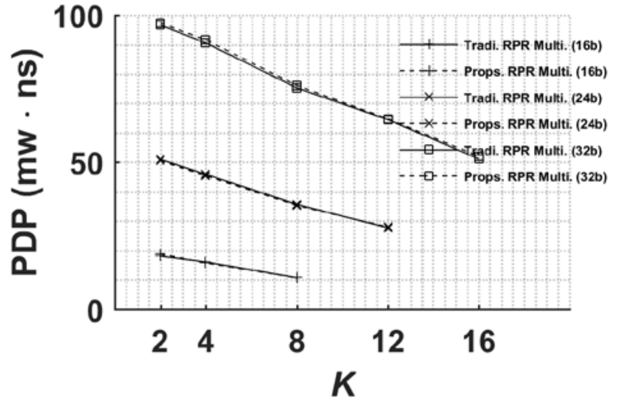

Figure 13. PDP for RPR multipliers protected with different levels of truncation

The costs of the proposed RPR MAC are shown in Figures 14 to 17. They follow similar trends as for the multipliers. This is expected as the multiplier dominates the cost and delay of the MAC operation. Therefore, it can be concluded that the proposed scheme has almost the same cost and delay as the traditional RPR implementation.

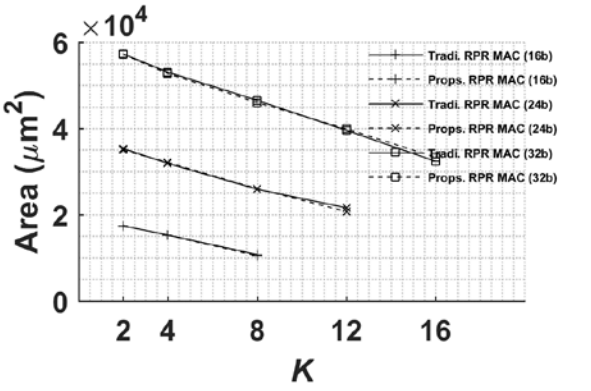

Figure 14. Area for a MAC protected with different levels of truncation

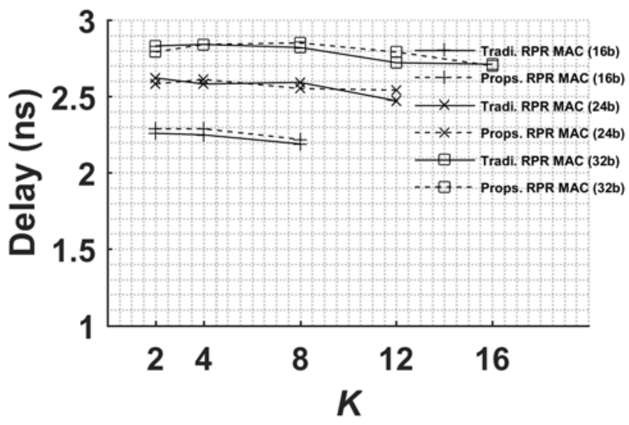

Figure 15. Delay for a MAC protected with different levels of truncation

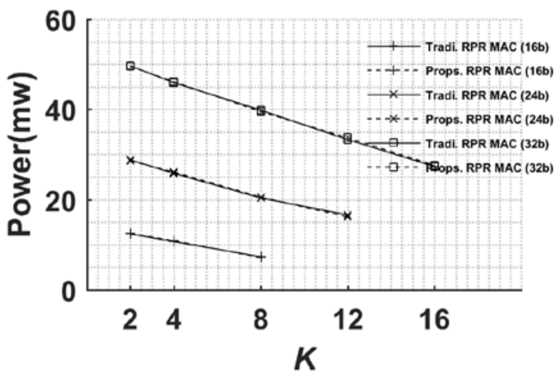

Figure 16. Power for a MAC protected with different levels of truncation

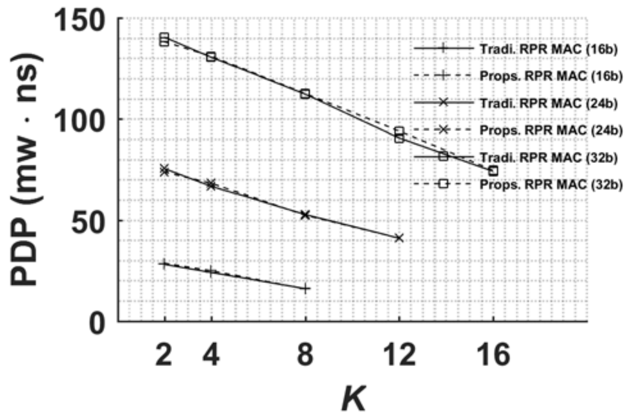

Figure 17. PDP for a MAC protected with different levels of truncation

5 Application: DCT

The Discrete Cosine Transform (DCT) removes the correlation of image elements in the transform domain [18] ; it is considered a quasi-optimal transform and has been widely applied in the fields of image and video coding/compression. In [19], instead of floating point numbers, DCT uses integers in the transform matrix; the transformation core is a signed integer transform involving no floating-point calculation and achieving a high accuracy. The core transformation can be completed only with simple signed integer matrix multiplication which are multiply and addition operations, therefore its computational complexity is significantly reduced. In matrix notation, the discrete two-dimensional radix-8 DCT is given by Y = P·X·P^T, where X is the 8x8 input image frame, P is the transform matrix and Y is the transformed (output) matrix. The DCT is implemented by two stages of matrix multiplication; these are given by:

$$P = \begin{bmatrix} P_{11} & P_{12} & \dots & P_{1M} \\ P_{21} & P_{22} & \dots & P_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ P_{M1} & P_{M2} & \dots & P_{MM} \end{bmatrix} = A \times B$$

$$= \begin{bmatrix} A_{11} & A_{12} & \dots & A_{1M} \\ A_{21} & A_{22} & \dots & A_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ A_{M1} & A_{M2} & \dots & A_{MM} \end{bmatrix} \times \begin{bmatrix} B_{11} & B_{12} & \dots & B_{1M} \\ B_{21} & B_{22} & \dots & B_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ B_{M1} & B_{M2} & \dots & B_{MM} \end{bmatrix}$$

where

$$P_{ij} = \sum_{k=1}^{M} A_{ik} \cdot B_{kj}$$

Thus, MAC is used as the basic processing step for matrix multiplication to calculate $A_{ik} \times B_{kj}$+S, where S is the accumulated sum. As described in [20], one of most common transform matrix (as also utilized in this paper) is

$$P1 = \begin{bmatrix} 64 & 64 & 64 & 64 & 64 & 64 & 64 & 64 \\ 89 & 75 & 50 & 18 & -18 & -50 & -75 & -89 \\ 83 & 36 & -36 & -83 & -83 & -36 & 36 & 83 \\ 75 & -18 & -89 & -50 & 50 & 89 & 18 & -75 \\ 64 & -64 & -64 & 64 & 64 & -64 & -64 & 64 \\ 50 & -89 & 18 & 75 & -75 & -18 & 89 & -50 \\ 36 & -83 & 83 & -36 & -36 & 83 & -83 & 36 \\ 18 & -50 & 75 & -89 & 89 & -75 & 50 & -18 \end{bmatrix}$$

For evaluating the signed integer DCT, 20 8-bit and 16-bit images are used; all images are from the publicly available database of [21]. Random (uniformly distributed) errors are injected in both the traditional and the proposed RPR schemes; then, the average Peak Signal Noise Ratio (PSNR) is measured for all 20 images at different values of K. The results are reported in Tables 2 and 3. As expected the PSNR increases using the proposed RPR scheme; on average for the two word-widths considered, the improvement is at least 5 dB and increases with K.

TABLE 2 AVERAGE PSNRS FOR DCT WITH RPR VOTER FOR N=16

K	Proposed RPR	Traditional RPR
2	61.63dB	56.85dB
4	60.50dB	54.96dB
8	58.34dB	51.48dB

TABLE 3 AVERAGE PSNRS FOR DCT WITH RPR VOTER FOR N=8

K	Proposed RPR	Traditional RPR
2	58.11dB	52.16dB
4	53.92dB	47.91dB

6 CONCLUSION

This paper has presented novel schemes for Reduced Precision Redundancy (RPR) as applicable to the commonly used operation of Multiply and Accumulate (MAC) in signed integer format. The proposed schemes utilize compensating error features in the multipliers and adders when signed integer processing is executed. Novel designs for MAC are proposed; initially the thresholds used for detecting errors within the RPR schemes have been determined analytically and checked by simulation. Then both the soft error tolerance and the implementation cost have been analyzed. The results show that a reduction of 40 to 60 percent on the Mean Square Error (MSE) can be achieved with only a marginal impact on the implementation cost in terms of delay and hardware overhead. As application, the computation of the DCT has been presented; the results confirm an improvement of PSNR of at least 5 dB on average and increasing with the reduced precision.

REFERENCES

[1]. Z. Luo and M. Martonosi "Accelerating pipelined integer and floating-point accumulations in configurable hardware with delayed addition techniques." IEEE Transactions on Computers, Bol. 49, No. 3, pp. 208-218, 2000.

[2]. J. T. Kao; M. Miyazaki; A. R. Chandrakasan "A 175-MV multiply-accumulate unit using an adaptive supply voltage and body bias architecture," IEEE Journal of Solid-State Circuits, Vol. 37, No. 11, pp. 1545-1554, 2002.

[3]. G. Li, S. Kumar, S. Hari, M. Sullivan, T. Tsai, K., Pattabiraman, J. Emer, and S.W. Keckle, "Understanding Error Propagation in Deep Learning Neural Network (DNN) Accelerators and Applications", In Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC), 8:1–8:12, 2017.

[4]. Y. Liu, S. Liu, Y. Wang, F. Lombardi and J. Han, "A Stochastic Computational Multi-Layer Perceptron Supporting a Backward Propagation Algorithm," IEEE Transactions on Computers (in press).

[5]. M. de la Guia Solaz and R. Conway "Razor Based Programmable Truncated Multiply and Accumulate, Energy-Reduction for Efficient Digital Signal Processing," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol 23, No. 1, pp. 189 – 193, 2015.

[6]. R. H. Brackert, M. D. Ercegovac, and A. N. Willson "Design of an on-line multiply-add module for recursive digital filters", In proceedings of 9th IEEE Symposium on Computer Arithmetic, pp. 34-41, 1989.

[7]. C. Jeangoudoux and C. Lauter "A Correctly Rounded Mixed-Radix Fused-Multiply-Add" In Proceedings of the 25th IEEE Symposium on Computer Arithmetic, pp 17-24, Amherst, 2018.

[8]. K. Chen, J. Han and F. Lombardi, "Design and Analysis of an Approximate 2D Convolver," in Proceedings of the IEEE International Symposium on DFT in VLSI and Nanotechnology Systems, pp. 31-34, Storrs, October 2016.

[9]. K. Chen, J. Han and F. Lombardi "Matrix Multiplication by an Inexact Systolic Array" in Proceedings of the ACM/IEEE Symposium on Nano Architectures, pp 151-156, Boston, July 2015.

[10]. N. Kanekawa, E. H. Ibe, T. Suga, and Y Uematsu, "Dependability in Electronic Systems: Mitigation of Hardware Failures, Soft Errors, and Electro-Magnetic Disturbances", New York, NY, USA: Springer-Verlag, 2010, doi: 10.1007/978-1-4419-6715-2.

[11]. M. Nicolaidis, "Design for soft error mitigation", IEEE Transactions on Device and Materials Reliability, vol. 5, no. 3, pp. 405–418, Sep. 2005.

[12]. I. Alzaher-Noufal and M. Nicolaidis, "A CAD Framework for Generating Self-Checking Multipliers Based on Residue Codes", in Proceedings of the Conference on Design, Automation and Test in Europe (DATE), 1999.

[13]. M. Nicolaidis and R. O. Duarte, "Fault-Secure Parity Prediction Booth Multipliers", IEEE Design & Test of Computers, vol. 16, no. 3, pp. 90-101, 1999.

[14]. S. Byonghyo, S. R. Sridhara, and N. R. Shanbhag, "Reliable low-power digital signal processing via reduced precision redundancy," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 12, no. 5, pp. 497-510, 2004.

[15]. B. Shim and N. R. Shanbhag, "Energy-efficient soft error-tolerant digital signal processing," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 14, no. 4, pp. 336-348, 2006.

[16]. M. A. Sullivan, "Reduced precision redundancy applied to arithmetic operations in field programmable gate arrays for satellite control and sensor systems," M.S. Thesis, Naval Postgraduate School, Monterey, California, USA, 2008.

[17]. A. Ullah, P. Reviriego, S. Pontarelli, and J. A. Maestro, "Majority Voting-Based Reduced Precision Redundancy Adders," IEEE Transactions on Device and Materials Reliability, vol. 18, no. 1, pp. 122-124, 2018.

[18]. N. Ahmed ,T. Natarajan and K.R. Rao, "Discrete Cosine Transform," IEEE Transactions on Computers, vol. 23, no.1, pp.90-93, Jan. 1974.

[19]. J. Wu and Y. Li, "A new type of integer DCT transform radix and its rapid algorithm," in Proceedings of the International Conference on Electric Information and Control Engineering (ICEICE), pp.1063-1066, 15-17 April 2011.

[20]. P.K. Meher, S. Y. Park, B.K. Mohanty, K. S. Lim and C. Yeo, "Efficient Integer DCT Architectures for HEVC," IEEE Transactions on Circuits and Systems for Video Technology, vol.24, no.1, pp.168-178, Jan. 2014.

[21]. TESTIMAGES - Testing images for scientific purposes https://sourceforge.net/projects/testimages/.