Carry Prediction and Selection for Truncated Multiplication

Romain Michard, Arnaud Tisserand, Nicolas Veyrat-Charvillon

Carry Prediction and Selection for Truncated Multiplication

Romain Michard*, Arnaud Tisserand† and Nicolas Veyrat-Charvillon*
* Arénaire project, LIP (CNRS–ENSL–INRIA–UCBL)
Ecole Normale Supérieure de Lyon
46 allée d'Italie. F-69364 Lyon, France
{firstame.lastname}@ens-lyon.fr
† Arith group, LIRMM (CNRS–Univ. Montpellier II)
161 rue Ada. F-34392 Montpellier, France
arnaud.tisserand@lirmm.fr

***Abstract*— This paper presents an error compensation method for truncated multiplication. From two n-bit operands, the operator produces an n-bit product with small error compared to the $2n$-bit exact product. The method is based on a logical computation followed by a simplification process. The filtering parameter used in the simplification process helps to control the trade-off between hardware cost and accuracy. The proposed truncated multiplication scheme has been synthesized on an FPGA platform. It gives a better accuracy over area ratio than previous well-known schemes such as the constant correcting and variable correcting truncation schemes (CCT and VCT).**

I. Introduction

In many digital signal processing applications, fixed-point arithmetic is used. In order to avoid word-size growth, operators with n-bit input(s) must return an n-bit result. For multiplication, the $2n$-bit result of an $(n \times n)$-bit product has to be set back to n-bit by dropping the n least significant bits through a reduction scheme (usually truncation or rounding). This is the purpose of *truncated multipliers*.

Truncated multiplication is used mainly for applications such as finite-impulse response (FIR) filtering and discrete cosine transform (DCT) operations. It can also be used to reduce the hardware cost of function evaluation [1].

This paper starts by the notations and presents the main methods used for truncated multiplication in Section II. In this section, we introduce a simple classification of truncated multiplication schemes. The proposed method is presented in Section III. Our method is based on carry prediction and selection. Section IV presents the error analysis and the implementations results on FPGAs. It also presents a comparison with some existing solutions.

II. Background

A. Notations

Figure 1 presents the partial product array (PPA) of an unsigned (4×4)-bit multiplication (see [2] for full-width multiplication algorithms). The partial product $x_i y_j$ is often represented as a dot for compact notation (see Fig. 2).

We use fixed-point notation with n fractional bits (i.e., $0.x_1x_2x_3 \ldots x_n$) for the operands and the result. As shown in Figure 2, MP represents the $n-1$ most significant columns of the partial product array. MP corresponds to the n bits of the final truncated result. w_{lsb} is the weight of the least significant bit in the truncated result, i.e., $w_{lsb} = 2^{-n}$. The least significant part of the PPA is noted LP, and we further distinguish its k most significant columns as LP_{major}, and the remaining $n-k$ columns as LP_{minor}. In some schemes, the column in LP_{minor} with the highest weight (the left-most column in LP_{minor} in Figure 2) is used. It is referred in the following as $LP^{(h)}_{minor}$. We refer to a column in the PPA by its weight, for example MP extends from columns col_1 to col_n, i.e. from the column where the partial products weight 2^{-1} to the column where the partial products weight $2^{-n} = w_{lsb}$.

				x_3	x_2	x_1	x_0
			$\times$	y_3	y_2	y_1	y_0
				x_3y_0	x_2y_0	x_1y_0	x_0y_0
			x_3y_1	x_2y_1	x_1y_1	x_0y_1	
		x_3y_2	x_2y_2	x_1y_2	x_0y_2		
$+$	x_3y_3	x_2y_3	x_1y_3	x_0y_3			
p_7	p_6	p_5	p_4	p_3	p_2	p_1	p_0

Fig. 1. Partial product array of a (4×4)-bit multiplication.

Function $\text{trunc}_n(x)$ denotes x truncated to the n-th bit, and $\text{round}_n(x)$ stands for x rounded to the n-th bit.

A truncated multiplication scheme computes the partial products in MP, and add an error compensation value (ECV) computed as a function of LP. The result of a truncated multiplication is noted

$$P = \text{trunc}_n\left(MP + f(LP)\right).$$

The most obvious way of performing a truncated multiplication is first to compute the exact $2n$ bits of the result, then round it to n bits. This full-width result is

$$P_{FW} = \text{round}_n\left(MP + LP\right).$$

While giving the smallest possible error, which is only due to the rounding, this method also requires the highest amount

1-4244-0383-9/06/$20.00 ©2006 IEEE

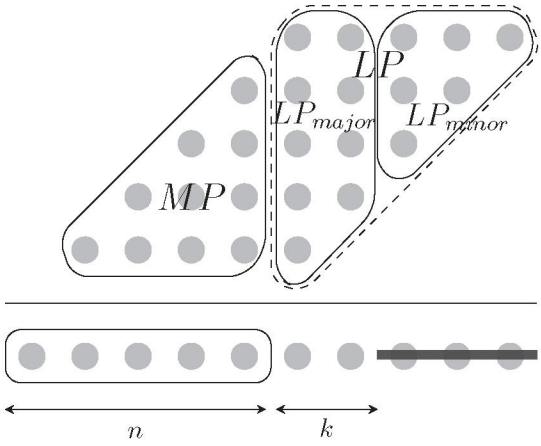


Fig. 2. The different parts of a partial product array.

of hardware by computing all the partial products.

Since the sum bits in the $2n$-bit full-width product are not all used, one is tempted to remove some low-weight columns in LP in order to diminish the hardware cost of the multiplier. However, by doing this, the carries in the low-weight part of the PPA are lost, thereby introducing an evaluation error.

Two kinds of error occur in truncated multiplication: the evaluation error E_{eval}, which is due to the columns that are removed in LP, and the truncation error E_{trunc}, which occurs when the computed value of the PPA is reduced to an n-bit value.

A direct-truncated multiplier computes only the $n-1$ most significant columns of the PPA. While minimizing the required amount of hardware, this approach does not take into account any of the carries propagating from LP, and leads to a maximal evaluation error. The result of a direct-truncated multiplier is

$$P_{DT} = \text{trunc}_n\left(MP\right).$$

B. Classification of the Truncated Multiplication Schemes

The truncated multiplication problem is the trade-off between accuracy and hardware cost. At one side, there is the full-width multiplier with the best accuracy but the highest cost. At the other side, there is the direct-truncated multiplier with the lowest cost but the worst accuracy. Many truncated multiplication schemes have been proposed with intermediate trade-offs.

We propose a simple classification based on the complexity of the ECV computation scheme. We distinguish two main kinds of solutions: *static* ECV and *dynamic* ECV. Static ECV means that the correction value does not depend on the actual values of the operands (the value is fixed at design time). Dynamic ECV uses a correction value computed using the actual operands (at run-time). Obviously, dynamic ECV is more accurate but it requires larger circuit area.

In order to refine the classification, we add subgroups depending on the PPA part impacted by the truncated multiplication method. We propose the five groups defined below.

- Static ECV with C: no partial product from LP is computed, the constant C is based on the LP part.
- Static ECV with $LP_{major}+C$: all partial products from LP_{major} are computed, the constant C is based on the LP_{minor} part.
- Dynamic ECV with $f(LP_{major})$: all partial products from LP_{major} are computed and used to evaluate the correction due to LP.
- Dynamic ECV with $LP_{major}+f(LP_{minor})$: all partial products from LP_{major} are computed, some partial products from LP_{minor} (usually $LP^{(h)}_{minor}$) are used to evaluate the ECV.
- Dynamic ECV with LP: all partial products in LP are computed and used to compute the ECV.

Table I presents the classification of the previous methods and our method accordingly to those groups.

C. Static ECV: C

In [6], the expected value of $C = Sum(LP)$ is estimated by assuming that each bit of the inputs has a probability $1/2$ of being one. The probabilities of each carry and sum bit are evaluated using the logic properties of the half-adder and full-adder cells.

The direct-truncated multiplication scheme described previously also fits in this category with $C = 0$, LP is not computed nor approximated.

D. Static ECV: $LP_{major}+C$

Truncated multiplication schemes with a static ECV approximate the error done by leaving out the low-weight columns LP_{minor} with a constant, which is computed either by exhaustive search on the input values or as a statistical evaluation of the expected value of LP_{minor}. In order to improve the accuracy, the k columns of LP_{major} are used as an extension of MP. The resulting $(n+k)$-bit value is then rounded or truncated to n bits.

In [3], every input bit is assumed to have a probability $\frac{1}{2}$ of being one. Each partial product therefore has an expected value $\frac{1}{4}$. By adding their expected values over LP_{minor}, Lim gets the expected value of the evaluation error:

$$E_{eval} = -\frac{w_{lsb}}{4}\sum_{i=0}^{n-k-1}(i+1)\,2^{i-n}. \qquad (1)$$

The multiplication result is:

$$P = \text{round}_n\left(MP + LP_{major} + \text{round}_{n+k}\left(-E_{eval}\right)\right),$$

and the parameter k is chosen so as to give to the evaluation error a variance lower than the variance $w^2_{lsb}/12$ of the rounding error, which is treated as a random noise.

This method is refined in the constant correction truncated (CCT) multiplication [7], where the error made by truncating the $(n+k)$-bit result to an n-bit value is computed assuming for each result bit of the multiplication a probability $\frac{1}{2}$ of being

Static ECV		Dynamic ECV		
C	$LP_{major} + C$	$f(LP_{major})$	$LP_{major} + f(LP_{minor})$	LP
direct-truncated [6]	[3] [7]	[4] [8] [11]	[5] [9], [10] our method	full-width

TABLE I
CLASSIFICATION OF THE ECV METHODS USED IN LITERATURE.

one. This gives:

$$E_{trunc} = -\frac{w_{lsb}}{2}\sum_{i=-k}^{-1} 2^i = -\frac{w_{lsb}}{2}(1-2^{-k}).$$

The multiplication result is:

$$P = \text{trunc}_n\left(MP + LP_{major} + \text{round}_{n+k}\left(-E_{eval} - E_{trunc}\right)\right).$$

E. Dynamic ECV: $f(LP_{major})$

In order to further diminish the error, some schemes have been proposed where, instead of approximating the value of the partial products in LP_{minor} by a constant, it is expressed as a function of the partial products in LP_{major}.

[4] gives an ECV for a modified Booth encoded PPA, where each line of partial products in LP_{minor} is estimated as a multiple of the corresponding partial product in LP_{major} ($k = 1$). This results in a data-dependent ECV.

[8] presents another dynamic ECV for a modified Booth multiplier. For every possible combination of bits in the recoded operand, a corresponding expected value of LP_{minor} is computed by statistic analysis, and added to the exact value of LP_{major} ($k = 1$). This gives for every possible value of the recoded operand an approximation of the carries propagated from LP. A carry generation circuit is then computed using a Karnaugh map. For sizes larger than 12, the exhaustive simulation is replaced by statistical analysis.

F. Dynamic ECV: $LP_{major} + f(LP_{minor})$

In [5], the ECV for a Baugh-Wooley array multiplier is computed in three parts. First LP_{major} is computed and summed. Then the partial products in $LP^{(h)}_{minor}$ are computed, some of them are inverted, and all this is summed. The pattern of inversions applied to the partial products in $LP^{(h)}_{minor}$ is parametrized by an integer Q. The sum is noted $\theta_{Q,k}$. Finally the expected value of $(LP_{minor} - \theta_{Q,k})$ is estimated, and added do $LP_{major} + \theta_{Q,k}$. The best value of Q is obtained by exhaustive search. For $n \geq 16$, a statistical analysis can be performed.

The variable correction truncated (VCT) multiplication [9], [10] estimates the carries propagated from LP_{minor} by adding to the least column of LP_{major} the partial products of $LP^{(h)}_{minor}$. This is equivalent to multiplying these partial-products by two. An immediate consequence is that the ECV is minimal when the multiplication operands are minimal, and vice versa. The truncation error E_{trunc} is the same as the one defined in the CCT multiplication.

The multiplication result is:

$$P = \text{trunc}_n\left(MP + LP_{major} + 2LP^{(h)}_{minor} + \text{round}_{n+k}\left(-E_{trunc}\right)\right)$$

An hybrid correction truncation (HCT) multiplication [11] realizes a compromise between the CCT and VCT multiplication by only using a percentage p of the partial products in $LP^{(h)}_{minor}$ for the ECV, and adding $1-p$ of the evaluation error E_{eval}, defined in the CCT multiplication. The truncation error E_{trunc} is also the same as in the CCT and VCT multiplications.

The multiplication result is:

$$P = \text{trunc}_n\left(MP + LP_{major} + p\cdot 2LP^{(h)}_{minor} + \text{round}_n\left((p-1)\cdot E_{eval} - E_{trunc}\right)\right)$$

G. Dynamic ECV: $LP_{major} + LP_{minor}$

Only the full-width multiplier fits into this category: this is the case where all of LP is computed.

III. PROPOSED METHOD

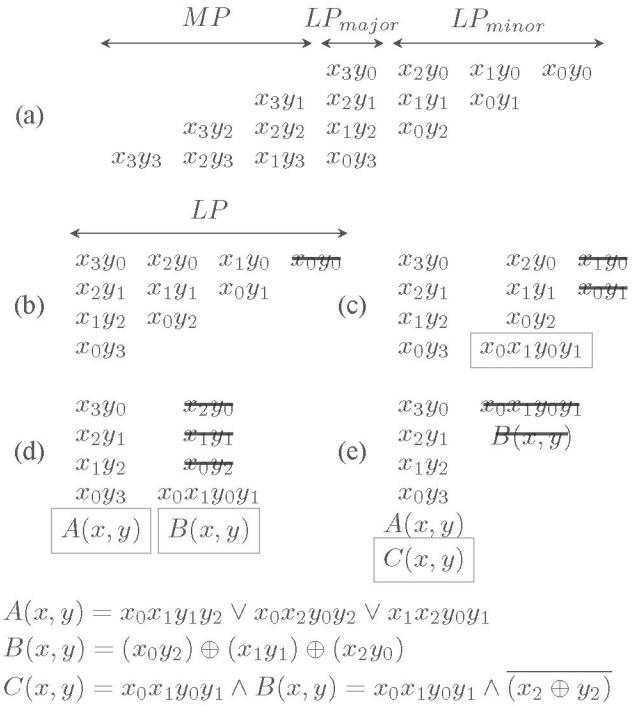
In this work, a new data-dependent truncated multiplication scheme is introduced. It is named *prediction-selection* correcting truncated (PSCT) multiplication. It is proposed for direct non-recoded unsigned array multiplication.

In the CCT, VCT and HCT multiplication schemes, the carries propagated from LP_{minor} are estimated, either by statistical analysis or with the help of the partial products in column $LP^{(h)}_{minor}$. It is then difficult to know what kind of error is done, and what additional terms might be introduced in order to improve accuracy.

Our approach tries to address this issue by computing in a first time the exact values of every carry generated in LP_{minor}, and then discarding the less probable ones. This scheme simplifies the computation of the ECV and lower the associated hardware cost, while keeping track of the error made by removing those products.

A. Carry Prediction

Consider a complete PPA as the one used for the full-width multiplication in Figure 2. Since the $n-k$ least significant bits of the result are discarded, the corresponding sum bits of LP_{minor} do not have to be computed. But if LP_{minor} is



$$A(x,y) = x_0x_1y_1y_2 \vee x_0x_2y_0y_2 \vee x_1x_2y_0y_1$$
$$B(x,y) = (x_0y_2) \oplus (x_1y_1) \oplus (x_2y_0)$$
$$C(x,y) = x_0x_1y_0y_1 \wedge B(x,y) = x_0x_1y_0y_1 \wedge \overline{(x_2 \oplus y_2)}$$

Fig. 3. The four steps of carry prediction for $n = 4$ and $k = 1$.

not implemented at all, all the carries generated there are lost, leading to the evaluation error described in Eq. (1).

In order to keep the evaluation error low while removing unnecessary hardware, only the logic formulas of the carries generated in LP_{minor} have to actually be implemented. These expressions are obtained by replacing the full-adder and half-adder cells in LP_{minor} by their respective logic definitions:

$$\begin{aligned}
&\text{carry}_{FA}(a,b,c) = ab \vee ac \vee bc \\
&\text{sum}_{FA}(a,b,c) = a \oplus b \oplus c \\
&\text{carry}_{HA}(a,b) = ab \\
&\text{sum}_{HA}(a,b) = a \oplus b
\end{aligned}$$

where ab means "a and b", $a \vee b$ is "a or b" and $a \oplus b$ stands for "a exclusive-or b".

It is possible to express the logic formulas of the carries through LP_{minor}. Once implemented, the carries generated in LP_{minor} are known exactly.

Figure 3 shows the carry prediction steps for a 4-bit multiplication with one column in LP_{major} (a).

- (b) The first step computes the carries generated in the least significant column of LP_{minor}, col_8. Since there is only one partial product there, no carry can be generated.
- (c) The carry generated in col_7 is added to col_6. This carry is expressed logically as $x_0x_1y_0y_1$.
- (d) The first carry in col_6, $A(x,y) = \text{carry}_{FA}(x_0y_2, x_1y_1, x_2y_0)$ is similarly added to col_5, and the corresponding sum bit $B(x,y) = \text{sum}_{FA}(x_0y_2, x_1y_1, x_2y_0)$ replaces the three partial products in col_6.
- (e) Finally $x_0x_1y_0y_1$ and $B(x,y)$ are removed and their carry $C(x,y)$ is added to col_5, in LP_{major}.

We note LP'_{major} and LP'_{minor} the parts of the PPA corresponding to LP_{major} and LP_{minor} obtained at the end of the prediction process.

All the partial products left in LP'_{minor} are sum bits, which do not need to be computed, since every carry originally generated in LP_{minor} is now computed in LP'_{major}.

After the carry prediction process, LP'_{minor} contains only one partial product in each column. Assuming that each result bit of the multiplication has a probability $\frac{1}{2}$ of being one, the expected value of the evaluation error E_{eval} is lower than $2^{-k-1}w_{lsb}$, so that $\text{round}_{n+k}(-E_{eval}) = 0$.

We can replace $\text{round}_{n+k}(LP)$ by LP'_{major}:

$$\begin{aligned}
P_{PS} &= \text{round}_n\left(MP + LP'_{major}\right) \\
&= \text{round}_n\left(MP + \text{round}_{n+k}\left(LP_{major} + LP_{minor}\right)\right) \\
&= P_{FW}
\end{aligned}$$

At this point, the truncated operator we obtain gives the same result as the full-width multiplication.

B. Carry Selection

So far, the evaluation error is lower than $2^{-k-1}w_{lsb}$, but the partial products in LP'_{major} become very complicated as n grows, and a large hardware cost may result. In order to reduce the area requirements of the truncated multiplier, some carries have to be simplified. For that purpose, the logic formulas are written under their simplified disjunctive normal form, and a "filter" is applied on the last column in LP'_{major}. This consists in removing any conjunction which number of variables exceeds a given threshold t. Assuming that the input bits have a probability $\frac{1}{2}$ of being one, a conjunction of $t+1$ variables will have a probability 2^{-t-1} of being one. Applying the filter, one makes an error of about $m \times 2^{-k-t-1}w_{lsb}$, where m is the number of conjunctions that were removed in the process. The evaluation error E_{eval} is updated in accordance to this value. The multiplication result is:

$$P_{PS} = \text{round}_n\left(MP + LP'_{major} + \text{round}_{n+k}(-E_{eval})\right) \quad (2)$$

Let us consider the previous example (Figure 3) of an n-bit truncated multiplication with $n = 4$ and $k = 1$. The only column in LP'_{major} contains $x_3y_0, \cdots, x_0y_3$, $A(x,y)$ and $C(x,y)$, where $C(x,y) = x_0x_1x_2y_0y_1y_2 \vee x_0x_1\overline{x_2}y_0y_1\overline{y_2}$ in its disjunctive normal form.

If no threshold is imposed, the truncated multiplication is equivalent to the ideal rounded multiplication.

For a value of the threshold $t = 4$, both conjunctions in $C(x,y)$ are removed, and the evaluation error is increased by the probability of $C(x,y)$ being one.

If the restriction on t is set down to 2, the three terms constituting $A(x,y)$ will also disappear. The evaluation error is further increased by the probability of $A(x,y)$ being one, and the truncated multiplier is now equivalent to a CCT multiplier.

Multiplication scheme	$\lvert\beta\rvert$	ε_{avg}	σ	ε_{max}
Direct-truncated	7.66e-1	7.66e-1	6.19e-1	3.06
Ideal rounded	6.25e-2	2.34e-1	1.68e-1	5.00e-1
PSCT $k=1$, $t=6$	6.25e-2	2.34e-1	1.68e-1	5.00e-1
PSCT $k=1$, $t=4$	4.69e-2	2.36e-1	1.71e-1	5.62e-1
PSCT $k=1$, $t=2$	3.12e-2	2.69e-1	2.15e-1	1.06
CCT $k=1$	3.12e-2	2.69e-1	2.15e-1	1.06
VCT $k=1$	3.44e-1	3.59e-1	2.81e-1	9.37e-1

TABLE II

ERROR ANALYSIS OF SEVERAL TRUNCATED MULTIPLICATION SCHEMES FOR $n=4$

IV. RESULTS

A. Mathematical Error

For each studied multiplier scheme, the absolute bias $|\beta|$, average absolute error ε_{avg}, standard deviation σ and absolute maximum error ε_{max} are given in output lsb $w_{lsb}=2^{-n}$. They are computed exhaustively for $n \leq 8$, and using an extensive random sampling for larger values of n.

The mathematical data for the previous example is given in Table II. We can see that, by acting on the value of the threshold t, we realize a compromise between the full-width multiplier and the CCT multiplier.

B. Synthesis

We studied the implementation of truncated multiplication schemes on FPGAs. The CAD tool used was Xilinx ISE8.1i and the target was an FPGA of the Spartan 3 family (XC3S200) with a medium speedgrade (-5). Synthesis and place-and-route were area-oriented with a standard effort . The multipliers were implemented using LUTs (not hardware block multipliers).

The Xilinx devices are optimized for 4 up to 8-bit input functions. This allows us to perform an efficient implementation of PSCT multipliers with a threshold up to $t=8$. We implemented the PSCT multipliers for t running from 3 to 8. A PSCT multiplier where $t=2$ is equivalent to a CCT multiplier with the same value of the parameter k.

C. Comparisons

The comparisons are lead with some well known truncation schemes for direct multiplication, that is the CCT [7] and VCT [9] multiplications. The full-width multiplier and direct-truncated multiplier are used as a reference.

The comparisons were done for $n=8$, 12 and 16. Our method is not yet fit for higher values of n, because of the fast growing computational cost of the prediction process.

Figure 4 shows how the different schemes behave for $n=8$, 12 and 16 from to top bottom. The X-axis gives the average absolute error, which is our principal accuracy criterion. The Y-axis gives the hardware cost relatively to the full-width multiplier. The aim is to perform a good accuracy while minimizing the hardware cost. This corresponds to the lower left part of each graph.

For $n=8$, the CCT is outperformed by the PSCT for $t=3$, 4 and 5. One can compute with the same average accuracy as the CCT with smaller PSCT multipliers. Similarly, for $n=12$, the PSCT for $t=3$ require less hardware to provide the same average accuracy as the CCT. For $n=16$, the two schemes are equivalent.

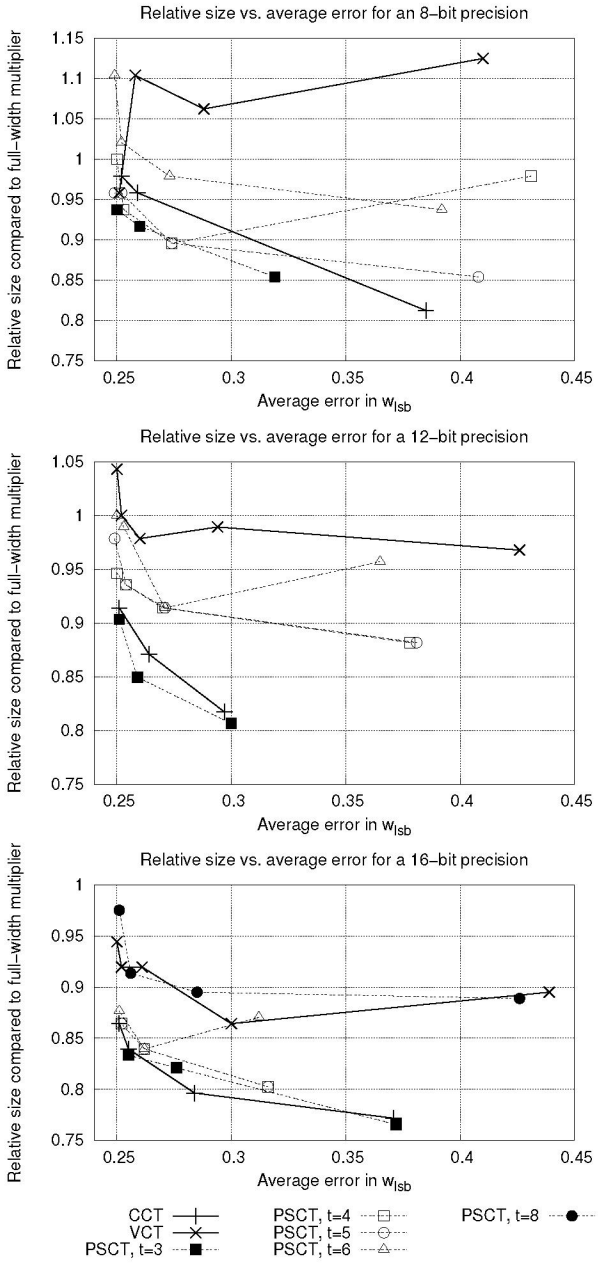


Fig. 4. Relative size vs. average error for truncated multiplication schemes.

Tables III, IV and V show accuracy results for the truncated multiplication schemes. If one wants to get an average accuracy as small as possible, that is get close to 0.25, the PSCT multiplication has a lower hardware cost than the other truncated multiplication methods.

Multiplication scheme	ε_{avg}	σ	ε_{max}	area	delay
Ideal rounded	2.49e-1	1.46e-1	5.00e-1	48	10.3
Direct-truncated	1.75	9.76e-1	7.00	32	8.9
CCT $k = 4$	2.52e-1	1.51e-1	5.66e-1	47	10.8
VCT $k = 4$	2.51e-1	1.50e-1	5.66e-1	46	11.4
PSCT $k = 4$, $t = 3$	2.50e-1	1.48e-1	6.29e-1	45	10.2
PSCT $k = 4$, $t = 5$	2.49e-1	1.47e-1	5.51e-1	46	11.7

TABLE III

ACCURACY RESULTS FOR 8-BIT TRUNCATED MULTIPLICATION SCHEMES

Multiplication scheme	ε_{avg}	σ	ε_{max}	area	delay
Ideal rounded	2.49e-1	1.45e-1	5.00e-1	93	12.7
Direct-truncated	2.73	1.24	9.00	53	11.5
CCT $k = 5$	2.51e-1	1.48e-1	5.71e-1	85	12.3
VCT $k = 4$	2.52e-1	1.49e-1	6.03e-1	93	14.6
PSCT $k = 5$, $t = 3$	2.51e-1	1.48e-1	6.56e-1	84	12.3
PSCT $k = 5$, $t = 4$	2.50e-1	1.46e-1	5.94e-1	88	14.3
PSCT $k = 5$, $t = 5$	2.49e-1	1.46e-1	5.78e-1	91	14.3

TABLE IV

ACCURACY RESULTS FOR 12-BIT TRUNCATED MULTIPLICATION SCHEMES

Multiplication scheme	ε_{avg}	σ	ε_{max}	area	delay
Ideal rounded	2.49e-1	1.45e-1	5.00e-1	162	15.2
Direct-truncated	3.76	1.48	12.5	95	13.2
CCT $k = 6$	2.51e-1	1.47e-1	5.62e-1	140	15.0
VCT $k = 4$	2.52e-1	1.50e-1	6.36e-1	149	15.3
VCT $k = 5$	2.50e-1	1.46e-1	5.59e-1	153	15.3
PSCT $k = 5$, $t = 4$	2.52e-1	1.50e-1	6.41e-1	140	16.0
PSCT $k = 5$, $t = 6$	2.51e-1	1.49e-1	6.26e-1	142	18.2

TABLE V

ACCURACY RESULTS FOR 16-BIT TRUNCATED MULTIPLICATION SCHEMES

CONCLUSION

We presented a new truncated multiplication scheme. The method first computes the logic expression of the carries propagated from LP_{minor}, then performs simplifications while keeping control over the introduced error. This scheme achieves an improvement both for accuracy and hardware requirements over previous schemes. The proposed method has been implemented on FPGAs, it shows an area reduction for comparable accuracy on 8 and 12-bit multipliers.

In a near future we plan to improve the speed of our method in order to deal with larger multipliers. We also plan to study the effects of different groupings of the partial products during the carry prediction phase, that should lead to accuracy and hardware cost improvements.

REFERENCES

[1] E. G. Walters and M. J. Schulte, "Efficient function approximation using truncated multipliers and squarers," in *Proc. of the 17th IEEE Symposium on Computer Arithmetic*. IEEE Computer Society, June 2005, pp. 232–239.

[2] M. D. Ercegovac and T. Lang, *Digital Arithmetic*. Morgan Kaufmann, 2003.

[3] Y. C. Lim, "Single-precision multiplier with reduced circuit complexity for signal processing applications," *IEEE Transactions on Computers*, vol. 41, no. 10, pp. 1333–1336, Oct. 1992.

[4] T.-B. Juang and S.-F. Hsiao, "Low-error carry-free fixed-width multipliers with low-cost compensation circuits," *IEEE Transactions on Circuits and Systems—II: Analog and Digital Signal Processing*, vol. 52, no. 6, pp. 299–303, June 2005.

[5] L.-D. Van and C.-C. Yang, "Generalized low-error area-efficient fixed-width multipliers," *IEEE Transactions on Circuits and Systems—I: Regular Papers*, vol. 52, no. 8, pp. 1608–1619, Aug. 2005.

[6] S. S. Kidambi, F. El-Guibaly, and A. Antoniou, "Area-efficient multipliers for digital signal processing applications," *IEEE Transactions on Circuits and Systems—II: Analog and Digital Signal Processing*, vol. 43, no. 2, pp. 90–95, Feb. 1996.

[7] M. J. Schulte and E. E. Swartzlander, "Truncated multiplication with correction constant," in *IEEE Workshop on VLSI Signal Proccessing VI*, Oct. 1993, pp. 388–396.

[8] K.-J. Cho, K.-C. Lee, J.-G. Chung, and K. K. Parhi, "Design of low-error fixed-width modified booth multiplier," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 12, no. 5, pp. 522–531, May 2004.

[9] E. J. King and E. E. Swartzlander, "Data-dependent truncation scheme for parallel multipliers," in *Proc. of 31th Asilomar Conference on Signals, Systems & Computers*, vol. 2. IEEE, Nov. 1997, pp. 1178–1182.

[10] E. E. Swartzlander, "Truncated multiplication with approximate rounding," in *Proc. of 33th Asilomar Conference on Signals, Systems & Computers*, vol. 2. IEEE, Oct. 1999, pp. 1480–1483.

[11] J. E. Stine and O. M. Duverne, "Variations on truncated multiplication," in *Proc. Euromicro Symposium on Digital System Design (DSD)*. IEEE, Sept. 2003, pp. 112–119.