On The Systematic Creation of Faithfully Rounded Truncated Multipliers and Arrays

Theo Drane, Thomas Rose and George A. Constantinides

Abstract—Often, when performing fixed-point multiplication, it is sufficient to return a faithfully rounded result, *i.e.* the machine representable number either immediately above or below the arbitrary precision result, if the latter is not exactly representable. Compared to correctly rounded multipliers, *i.e.* those returning the nearest machine representable number, faithfully rounded multipliers use considerably less silicon area, typically by implementing a truncation scheme within the partial product array. A number of such heuristically inspired schemes exist in the literature, however their use in industrial practice is hampered by the absence of verification, and exhaustive simulation is typically infeasible, *e.g.* a 32 bit multiplier requires 2^{64} simulations. We present three truncated multiplier schemes which subsume the majority of existing schemes and derive both closed form necessary and sufficient conditions for faithful rounding. For two of the schemes we provide closed form expressions for the bit vectors giving rise to the worst-case error and the probability of encountering these inputs during Monte-Carlo simulation. From these expressions, we show how HDL code can be created that performs correct-by-construction faithfully rounded multiplication. We also present a method for truncating an arbitrary array while maintaining faithful rounding, creating two novel truncated multiplier schemes in the process.

Index Terms—Data-path design, parallel circuits, high-speed arithmetic, worst-case analysis.

1 Introduction

There are many applications in which the full result of a fixed point multiplication is not required, but an appropriately rounded result can be returned. The challenge is to create the most effective trade off between area and error properties. Even for simple truncation schemes there are a wealth of design options and trade offs that can be made, but gathering error statistics for even modestly sized multipliers is extremely time consuming. Such is this problem, there has even been research into how to perform the exhaustive simulation of truncated multipliers efficiently [1]. In order to facilitate high level datapath synthesis capable of searching the design space of single or interconnected truncated multipliers in an acceptable time, analytic formulae must be found.

The structure of the majority of truncated multiplication schemes of two n by n bit inputs a and b producing an n bit output y is as follows: truncate the multiplier array by removing the value contained in the least significant k columns, denoted $\triangle_k$, prior to the addition of the partial products [2]. A hardware-efficient function of the two multiplicands $f(a,b)$ is then introduced as compensation into column k. Once the resultant array is summed, a further $n-k$ columns are truncated, the result is then the approximation to the multiplication. The structure of the general multiplier truncation scheme is shown in Figure 1, the array in the figure is that of a traditional AND array multiplier. The underlying array may of course differ in structure, ranging from Booth arrays of various radix to squarer arrays and constant multiplication, *etc*, [3]. We will first concentrate on the truncations of the AND array before exploring other array types.

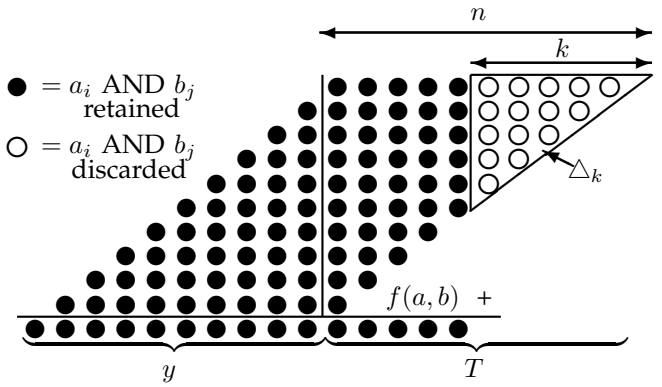

Fig. 1. Structure of AND Array Multiplier Truncation Schemes.

The scheme may be summarised algebraically:

$$y = 2^n \left\lfloor \frac{ab + 2^k f(a,b) - \triangle_k}{2^n} \right\rfloor \qquad a,b,n,k \in \mathbb{Z}^+ \qquad (1)$$

The error, compared to the precise answer, introduced by doing so is

$$\varepsilon = ab - 2^n \left\lfloor \frac{ab + 2^k f(a,b) - \triangle_k}{2^n} \right\rfloor$$
$$\varepsilon = \left(\left(ab + 2^k f(a,b) - \triangle_k \right) \mod 2^n \right) + \triangle_k - 2^k f(a,b)$$
$$\varepsilon = T + \triangle_k - 2^k f(a,b)$$

where $T = \left(ab + 2^k f(a,b) - \triangle_k \right) \mod 2^n$. A design that

- *T. Drane and T. Rose are with Imagination Technologies, Imagination House, Home Park Estate, Kings Langley, Hertfordshire, WD4 8LZ. E-mails:{theo.drane,thomas.rose}@imgtec.com*
- *G. Constantinides is with the Department of Electrical and Electronic Engineering, Imperial College London, Exhibition Road, London, SW7 2BT. E-mail: g.constantinides@imperial.ac.uk*

exhibits faithful rounding is one such that:

$$\forall a, b \qquad |\varepsilon| < 2^n$$

Note that if the correct answer is exactly representable, which occurs when the lower n bits of the multiplier result are all zero, then this perfect answer must be returned by a faithfully rounded scheme, otherwise $|\varepsilon| \geq 2^n$. Early truncation schemes considered $f(a, b)$ being constant [2] and [4], referred to as Constant Correction Truncated schemes (CCT). Following these, the proposal to make $f(a, b)$ a function of a and b appeared, termed Variable Correction Truncation (VCT) where the most significant column that is truncated is used as the compensating value for $f(a, b)$ [5]. A hybrid between CCT and column promoting VCT has been proposed which only uses some of the partial product bits of the promoted column, termed Hybrid Correction Truncation [6]. Arbitrary functions of the most significant truncated column have been considered along with their linearization; one of these linearisations requires promoting all but the four most extreme partial products bits and adding a constant, called LMS truncation due to the fact it targets the least mean square error [7] [8]. Forming approximations to the carries produced by $\triangle_k$ has also been put forward, termed carry prediction [9].

For Booth arrays, typically radix-4, their truncation history followed a similar path to that of the AND arrays, first following a CCT type truncation [10] and column promotion [11]. Exhaustive simulation of the truncated part of the Booth array was used to design compensation circuitry based upon the conditional expectation of the error [12], or in order to construct Karnaugh maps of the ideal correction [13]. Recent work has focused on purely analytic techniques for computing the expected errors [14], [15].

Truncated arrays also have been considered for squarers, radix-4 and 16 and Booth squarer arrays [16], [17], [18]. Truncated arrays that perform multiplication by a fixed constant have been considered in [19] and [20], the former requiring exhaustive simulation in order to establish the truncation scheme and the latter performing analytic calculations to establish the optimal linear compensation factor that minimises the mean square error.

In terms of applications, DSP has been the main focus area but they also appear in creating floating point multipliers, where a one unit in the last place (ulp) accuracy is permitted [21]. The evaluation of transcendental functions has also been considered, utilising truncated multipliers as well as truncated squarers [22].

In general, given the focus has been on DSP applications, second order statistics of the error have been important. New truncation schemes often require exhaustive simulation as part of their construction or their validation. In advanced compensation schemes such as [9], it is commented that it is difficult to know what kind of error is being generated and while exhaustive searches were conducted for $n \leq 8$, for sizes above this, the only option was to resort to random test vectors. In [23], finding the best compensation function requires searching a space exponential in n and is only feasible for $n < 13$. Further the schemes either find compensating functions heuristically or attempt to minimise the average absolute error or mean square error.

Research looking at the absolute maximum error is less common. In [24] bounds for a truncated radix-4 Booth array are created. Truncated multipliers have been designed to minimise second order error [25], and their maximum absolute error has been bound. An explicit attempt to create faithfully rounded multipliers, constructed by truncating, deleting and rounding the multiplication during the array construction, reduction and final integer addition can be found in [26].

The aim of this paper is to systematically create truncated multipliers which are known *a priori* to be faithfully rounded, without the need for simulation or exploration, and as such are amenable to an industry standard synthesis flow. Leading synthesis tools are extremely efficient at performing the summation of an arbitrary number of summands by avoiding expensive carry-propagations and using redundant representations such as carry-save [27]. The array reduction is context-driven depending on the timing and area constraints and standard cell libraries in use. Access to the array reduction or carry-save redundant signals is not possible from within the HDL code. Creating HDL code which explicitly states which compressor cells to use (full-adders, 4-to-2 compressors, *etc.*) in order to gain access to the intermediate redundant representation will lack the timing and context driven reduction achievable by the synthesis tool and will thus produce lower quality results. For these reasons we do not consider the approach of [26] as a viable option, as it modifies the multiplier array reduction directly and requires access to intermediate carry-save signals. We also seek the most hardware efficient multiplier structure for a given architecture, so we require the necessary and sufficient conditions for faithfully rounding. To our knowledge the only tight error bound held within the literature is for the LMS schemes [25] and [28]. We aim to construct a variety of faithfully rounded truncated multipliers for a range of schemes found within the literature and to compare their synthesis properties. Our first interest will be in truncated AND arrays as these are invariably commutative but then go on to consider other array types. The contributions of this paper are:

- the first tight analytic error bounds for CCT and VCT,
- analytic necessary and sufficient conditions for faithfully rounding for CCT, VCT and LMS schemes,
- worst case error and associated error vectors for CCT and VCT as well as the probability of encountering this during simulation,
- procedure for constructing the smallest faithful rounding of an arbitrary array,
- procedure for constructing the smallest faithfully

rounded multipliers,
- experimental synthesis comparison of schemes,
- general construction of a faithfully rounded floating point multiplier.

The paper is

organized as follows. The definitions of the three truncation schemes of interest are given in Section 2. Faithful rounding analysis of each scheme is presented in sections 3, 4 & 5. How to faithfully round an arbitrary array is presented in Section 6 and two resultant novel truncation schemes are given in sections 6.4 and 6.5. How to construct faithfully rounded truncated multiplier schemes is given in Section 7, experimental results in Section 8 and finally how these fixed-point multipliers can be used to construct faithfully rounded floating point multipliers is presented in Section 9.

2 CCT, VCT and LMS Multiplier Truncation

CCT uses a single constant C as the compensating function $f(a,b)$, as first put forward in [2], so in this case:

$$f_{CCT}(a,b) = C$$

Column promoting truncated multiplication (VCT) takes $f(a,b)$ to be the most significant column of $\triangle_k$ (denoted col_{k-1}) as put forward in [5], [21]:

$$f_{VCT}(a,b) = C + col_{k-1}$$

The LMS scheme, as put forward in section 8 of [7], promotes the interior of col_{k-1} into col_k leaving the extreme four partial products bits and adding a constant one into column $n-1$. This can be represented algebraically by noting that elements of col_{k-1} are $a_0b_{k-1}, a_1b_{k-2}, ..., a_{k-2}b_1, a_{k-1}b_0$ as follows:

$$\begin{aligned} f_{LMS}(a,b) =& 2^{n-k-1} + \sum_{i=2}^{k-3} a_i b_{k-1-i} \\ &+ \frac{1}{2}\left(a_0b_{k-1} + a_1b_{k-2} + a_{k-2}b_1 + a_{k-1}b_0\right) \end{aligned}$$

3 Necessary and Sufficient Conditions for CCT Faithful Rounding

In the case of CCT the error is:

$$\varepsilon_{CCT} = T + \triangle_k - 2^k C$$

3.1 Bounding CCT Error

Now T is the result of the summation in columns $n-1$ down to k, so its smallest value is 0 and its largest $2^n - 2^k$ hence we have the bound $0 \le T \le 2^n - 2^k$. Now $\triangle_k$ can be full of zeros when $a_{k-1:0} = b_{k-1:0} = 0$ (where $a_{k-1:0}$ denotes the bits of a in columns $k-1$ down to 0) and full of ones when $a_{k-1:0} = b_{k-1:0} = 2^k - 1$, hence:

$$0 \le \triangle_k \le \sum_{i=0}^{k-1}(2^k - 2^i) = (k-1)2^k + 1$$

So our initial bound on ε_{CCT} becomes:

$$-C2^k \le \varepsilon_{CCT} \le 2^n - (C-k+2)2^k + 1 \qquad (2)$$

The important question is whether or not there exists values for a and b where T and $\triangle_k$ can simultaneously achieve their lower/upper bound. The next section will prove that this is possible and, hence, that this initial bound is, in fact, tight.

3.2 CCT Error Bounds are Attained

The lower bound is achieved when $T = \triangle_k = 0$. Consider the case when $a_{k:0} = 100...000$, then:

$$\begin{aligned} T &= \left(\left(a_{n-1:k+1}2^{k+1} + 2^k\right)b + C2^k\right) \mod 2^n \\ T2^{-k} - C &= (2a_{n-1:k+1} + 1)\, b \mod 2^{n-k} \end{aligned}$$

Now $2a_{n-1:k+1}+1$ is odd, hence coprime to 2^{n-k}, hence, regardless of the value of C, we can always find a and b such that any given T can be achieved when $\triangle_k$ is minimal.

The upper bound is achieved when T and $\triangle_k$ are both maximal. In the case when $a_{k-1:0} = b_{k-1:0} = 2^k - 1$:

$$\begin{aligned} T =& \left(\left(a_{n-1:k}2^k + 2^k - 1\right)\left(b_{n-1:k}2^k + 2^k - 1\right)\right. \\ &\left. + C2^k - \max(\triangle_k)\right) \mod 2^n \\ T2^{-k} &- a_{n-1:k}\left(2^k - 1\right) - C - 2^k + k + 1 \\ &= b_{n-1:k}\left(a_{n-1:k}2^k + 2^k - 1\right) \mod 2^{n-k} \end{aligned}$$

Now $a_{n-1:k}2^k + 2^k - 1$ is odd hence coprime to 2^{n-k} hence, regardless of the value of C, we can always find $a_{n-1:k}$ and $b_{n-1:k}$ such that any given T can be achieved when $\triangle_k$ is maximal.

3.3 CCT Worst Case Error Vectors

Given the bounds on ε_{CCT} the error is positive when the scheme has largest absolute worst case error if $C2^k < 2^n - (C-k+2)2^k + 1$ which simplifies to $2C - k + 1 < 2^{n-k}$. The error is largest positive when $\triangle_k$ is maximal and $T = 2^n - 2^k$, the previous section shows that once $a_{n-1:k}$ is chosen, $b_{n-1:k}$ is fixed; hence there are 2^{n-k} possible worst case error vectors.

The error is largest negative when $\triangle = T = 0$. The previous section shows how this can be achieved when $a_{k:0} = 100...000$ and $-C = a_{n-1:k}b \mod 2^{n-k}$. Now let x be the largest integer such that 2^x divides C, then this implies that $b_{x-1:0} = 0$ and the equation reduces to:

$$a_{n-x-1:k}b_{n-k-1:x} + C/2^x = 0 \mod 2^{n-k-x}$$

This leaves $n-1$ bits of the inputs unconstrained, hence there are 2^{n-1} such error vectors. We assumed $a_k = 1$ but $a \times b$ is divisible by 2^{k+x} so we can distribute these powers of 2 between a and b. There are $k+x+1$ ways of doing so, hence in total there are $(x+k+1)2^{n-1}$ total error vectors. In summary:

Theorem 3.1: CCT Error Vectors
If $2C - k + 1 < 2^{n-k}$ there are 2^{n-k} worst case CCT

error vectors, specified by:

$$a_{k-1:0} = b_{k-1:0} = 2^k - 1$$
$$b_{n-1:k} = -p\left(2^k + C - k + a_{n-1:k}\left(2^k - 1\right)\right) \mod 2^{n-k}$$

where p is an integer satisfying $p \times a = 1 \mod 2^{n-k}$. Hence there is a 2^{-n-k} probability of encountering such inputs in simulation, provided all input sequences are equally likely.

Otherwise there are $(k+x+1)2^{n-1}$ worst case error vectors, where x be the largest integer, such that 2^x divides C, for $m = 0, 1, 2..., k+x$ specified by:

$$a_m = 1 \quad b_{k+x-m-1:0} = a_{m-1:0} = 0$$
$$a_{n-k-x+m-1:m} b_{n-m-1:k+x-m} + C/2^x = 0 \mod 2^{n-k-x}$$

Hence there is a $(k+x+1)2^{-n-1}$ probability of encountering such inputs in simulation in this case, provided all input sequences are equally likely.

3.4 The CCT Theorem

Given the error bounds are tight we can derive the necessary and sufficient conditions for the CCT scheme to be faithfully rounded from Equation 2:

Theorem 3.2: The CCT Theorem
The necessary and sufficient condition for the CCT scheme to be faithfully rounded is:

$$|\varepsilon_{CCT}| < 2^n \iff 2^{n-k} > C > k-2$$

4 Necessary and Sufficient Conditions for VCT Faithful Rounding

In the case of VCT the error is:

$$\varepsilon_{VCT} = T + \triangle_k - 2^k col_{k-1} - 2^k C$$

We will first deal with providing tight bounds for $\mu = \triangle_k - 2^k col_{k-1}$.

4.1 Bounding Maximum VCT μ Error

Exhaustive simulation for small k show particular forms for $\triangle_k$ when μ is maximal. When k is odd, *e.g.* 7 these forms are:

0	1	0	1	0	1	1		0	0	1	0	1	0	1
0	0	0	0	0	0			0	1	0	1	0	1	
0	1	0	1	1				0	0	0	0	0		
0	0	0	0					0	1	0	1			
0	1	1						0	0	0				
0	0							0	1					
0								0						

When k is even, *e.g.* 8 these forms are:

0	1	0	1	0	1	0	1	0	0	1	0	1	0	1	1
0	0	0	0	0	0	0		0	1	0	1	0	1	1	
0	1	0	1	0	1			0	0	0	0	0	0		
0	0	0	0	0				0	1	0	1	1			
0	1	0	1					0	0	0	0				
0	0	0						0	1	1					
0	1							0	0						
0								0							

These simulations gave rise to the following theorem:

Theorem 4.1: The VCT Maximal Theorem

$$\mu = \triangle_k - 2^k col_{k-1} \text{ is maximal} \iff \begin{array}{c} col_{k-1} = 0 \text{ and} \\ col_{k-2} \text{ is alternating} \end{array}$$

Proof:

- $\mu(a_0, b_{k-1}) = a_0(-2^{k-1}b_{k-1} + b_{k-2:0}) + const.$ Maximising μ over a_0 and b_{k-1} gives $a_0 = 1$, $b_{k-1} = 0$ and $b_{k-2:0} > 0$. By a symmetrical argument we get $a_0 = b_0 = 1$ and $a_{k-1} = b_{k-1} = 0$.
- $\mu(a_j, b_{k-1-j}) = -2^{k-1}a_j b_{k-1-j} + 2^j a_j b_{k-2-j:0} + 2^{k-1-j} b_{k-1-j} a_{j-1:0} + const.$ Maximising μ over a_j and b_{k-1-j} given that $a_0 = b_0 = 1$ gives rise to $a_j \neq b_{k-1-j}$ and hence $col_{k-1} = 0$.
- Consider the case when there are two adjacent zeroes in col_{k-2} so we have a location where:

$$\begin{array}{lll} a_{j-1}b_{k-j} & a_{j-1}b_{k-j-1} = & 0 \quad 0 \\ a_j b_{k-j-1} & a_j b_{k-j-2} = & 0 \quad 0 \end{array}$$

  Assuming that $a_j \neq b_{k-1-j}$ for all j and, by symmetry, we may assume $a_{j-1} = 1$. Solving the above equations means $a_{j:j-1}$ =11 and $b_{k-j:k-j-2}$=000. If we had set $a_{j:j-1}$ =01 and $b_{k-j:k-j-2}$=010 then we would have increased μ by:

$$\begin{aligned} & 2^{k-2} + 2^{k-j-1}a_{j-2:0} - 2^j b_{k-j-3:0} \\ >& 2^{k-2} + 2^{k-j-1}a_{j-2:0} - 2^j(2^{k-j-2} - 1) \\ =& 2^j + 2^{k-j-1}a_{j-2:0} > 0 \end{aligned}$$

  Hence when μ is maximal, adjacent zeroes never appear in col_{k-2}.
- If adjacent ones were to appear in col_{k-2} then there would be a one in column col_{k-1}, which contradicts the fact that when μ is maximal $col_{k-1} = 0$.

We can conclude from these four observations that $col_{k-1} = 0$ and col_{k-2} is an alternating binary sequence when μ is maximal. In fact these two conditions heavily restrict a and b. When k is odd these conditions imply $a_{k-1:0} = (2^{k-1} - 1)/3$ and $b_{k-1:0} = (2^k + 1)/3$. When k is even we have $a_{k-1:0} = b_{k-1:0} = (2^k - 1)/3$ or $a_{k-1:0} = b_{k-1:0} = (2^{k-1} + 1)/3$. After much arithmetic, from these cases we can derive the following tight upper bound on μ:

$$\mu \leq \frac{(3k-2)2^{k-1} + (-1)^k}{9}$$

4.2 Bounding Minimum VCT μ Error

Exhaustive simulation for small k show particular forms for $\triangle_k$ when μ is minimal. When k is odd, *e.g.* 7 these forms are:

1	0	1	0	1	0	1		1	1	0	1	0	1	1
0	0	0	0	0	0			1	0	1	0	1	1	
1	0	1	0	1				0	0	0	0	0		
0	0	0	0					1	0	1	1			
1	0	1						0	0	0				
0	0							1	1					
1								1						

When k is even, *e.g.* 8 these forms are:

1	0	1	0	1	0	1	1	1	1	0	1	0	1	0	1
0	0	0	0	0	0	0		1	0	1	0	1	0	1	
1	0	1	0	1	1			0	0	0	0	0	0		
0	0	0	0	0				1	0	1	0	1			
1	0	1	1					0	0	0	0				
0	0	0						1	0	1					
1	1							0	0						
1								1							

These simulations gave rise to the following theorem:

Theorem 4.2: The VCT Minimal Theorem

$$\mu = \triangle_k - 2^k col_{k-1} \text{ is minimal} \iff \begin{array}{c} col_{k-1}\text{'s extremes are 1} \\ col_{k-1}\text{'s interior alternates} \end{array}$$

The proof can be found in the supplemental material associated with this paper. These two conditions heavily restrict a and b. When k is odd these conditions imply $a_{k-1:0} = b_{k-1:0} = (2^{k+1}-1)/3$ or $a_{k-1:0} = b_{k-1:0} = (5 \times 2^{k-1}+1)/3$. When k is even we have unique a and b (up to swapping): $a_{k-1:0} = (5 \times 2^{k-1}-1)/3$ and $b_{k-1:0} = (2^{k+1}+1)/3$. After much arithmetic, from these cases we can derive the following tight lower bound on μ:

$$\mu \geq -\frac{(3k+7)2^{k-1}+(-1)^k}{9}$$

So in summary we have:

$$\varepsilon_{VCT} = T + \mu - 2^k C$$

Inserting the bounds for μ and knowing that $0 \leq T \leq 2^n - 2^k$ we get:

$$\begin{array}{rl} \varepsilon_{VCT} & \geq -\frac{2^{k-1}(3k+18C+7)+(-1)^k}{9} \\ & \leq 2^n + \frac{2^{k-1}(3k-18C-20)+(-1)^k}{9} \end{array} \quad (3)$$

The important question is whether or not there exists values for a and b where T and μ can simultaneously achieve their lower/upper bound. The next section will prove that this is possible and, hence, that this initial bound is, in fact, tight.

4.3 VCT Error Bounds are Attained

The lower bound is achieved when T and $\triangle_k$ are both minimal. When k is even μ is minimised by setting $a_{k-1:0} = (5 \times 2^{k-1}-1)/3$ and $b_{k-1:0} = (2^{k+1}+1)/3$, in this case T is:

$$\begin{aligned} T = \Bigg(\left(a_{n-1:k}2^k + \frac{5 \times 2^{k-1}-1}{3}\right) \left(b_{n-1:k}2^k + \frac{2^{k+1}+1}{3}\right) \\ + C2^k - \min(\mu) \Big) \mod 2^n \end{aligned}$$

$$\begin{aligned} T2^{-k} - C - \frac{5 \times 2^{k+1}+3k+8}{18} - a_{n-1:k}\frac{2^{k+1}+1}{3} \\ = b_{n-1:k}\left(a_{n-1:k}2^k + \frac{5 \times 2^{k-1}-1}{3}\right) \mod 2^{n-k} \end{aligned}$$

Now $a_{n-1:k}2^k + (5 \times 2^{k-1}-1)/3$ is odd hence coprime to 2^{n-k} hence, regardless of the value of C, we can always find a and b such that any given T can be achieved when μ is minimal and k is even. Similarly when k is odd we have $a_{k-1:0} = b_{k-1:0} = (2^{k+1}-1)/3$ or $a_{k-1:0} = b_{k-1:0} = (5 \times 2^{k-1}+1)/3$ and the argument proceeds in an identical manner. We therefore conclude that regardless of the value of C we can always find a and b such that any given T can be achieved when μ is minimal.

The upper bound is achieved when T and $\triangle_k$ are both maximal. When k is odd these conditions imply $a_{k-1:0} = (2^{k-1}-1)/3$ and $b_{k-1:0} = (2^k+1)/3$, in this case T is:

$$\begin{aligned} T = \Bigg(\left(a_{n-1:k}2^k + \frac{2^{k-1}-1}{3}\right) \left(b_{n-1:k}2^k + \frac{2^k+1}{3}\right) \\ + C2^k - \max(\mu) \Big) \mod 2^n \end{aligned}$$

$$\begin{aligned} T2^{-k} - C - \frac{2^k-3k+1}{18} - a_{n-1:k}\frac{2^k+1}{3} \\ = b_{n-1:k}\left(a_{n-1:k}2^k + \frac{2^{k-1}-1}{3}\right) \mod 2^{n-k} \end{aligned}$$

Now $a_{n-1:k}2^k + (2^{k-1}-1)/3$ is odd hence coprime to 2^{n-k} hence, regardless of the value of C, we can always find a and b such that any given T can be achieved when μ is maximal and k is odd. Similarly when k is even we have $a_{k-1:0} = b_{k-1:0} = (2^k-1)/3$ or $a_{k-1:0} = b_{k-1:0} = (2^{k-1}+1)/3$ and the argument proceeds in an identical manner. We therefore conclude that regardless of the value of C we can always find a and b such that any given T can be achieved when μ is maximal.

4.4 VCT Worst Case Error Vectors

Given the bounds on ε_{VCT} the error is positive when the scheme has largest worst case error in absolute value if

$$\begin{aligned} \frac{2^{k-1}(3k+18C+7)+(-1)^k}{9} \\ < 2^n + \frac{2^{k-1}(3k-18C-20)+(-1)^k}{9} \end{aligned}$$

This simplifies to

$$4C + 3 < 2^{n-k+1}$$

TABLE 1
X values for VCT worst case error vectors.

$4C+3<2^{n-k+1}$	k odd	$a_{k-1:0}$	$b_{k-1:0}$	X
✓	✓	00101...10101	01010...01011	$2^k-3k+19$
✓	✓	01010...01011	00101...10101	$2^k-3k+19$
✓	x	01010...10101	01010...10101	$2^{k+1}-3k+16$
✓	x	00101...01011	00101...01011	$2^{k-1}-3k+22$
x	✓	10101...10101	10101...10101	$2^{k+3}+3k-1$
x	✓	11010...01011	11010...01011	$2^{k-1}25+3k+17$
x	x	11010...10101	10101...01011	$2^{k+1}5+3k+8$
x	x	10101...01011	11010...10101	$2^{k+1}5+3k+8$

Note that equality is not possible given the integer nature of n, k and C. The exact error vectors can be found by following through the proofs in the previous sections, maximising and minimising μ and T. Note that in every case there are precisely 2^{n-k+1} error vectors. In summary:

Theorem 4.3: VCT Error Vectors
There are 2^{n-k+1} worst case VCT vectors for any given n, k and C. Table 1. defines two different sets of values for $a_{k-1:0}$ and $b_{k-1:0}$. Further, $a_{n-1:k}$ can take any value, $b_{n-1:k}$ is then required to be:

$$b_{n-1:k} = -p\left(C + X/18 + a_{n-1:k}b_{k-1:0}\right) \mod 2^{n-k}$$

where p is an integer satisfying $p \times a = 1 \mod 2^{n-k}$ and X is defined in Table 1. The probability of encountering these worst case errors in simulation is thus 2^{-n-k-1}, provided all input sequences are equally likely.

4.5 The VCT Theorem

Given the error bounds are tight we can derive the necessary and sufficient conditions for the VCT scheme to be faithfully rounded from 3:

Theorem 4.4: The VCT Theorem
The necessary and sufficient condition for the VCT scheme to be faithfully rounded is:

$$|\varepsilon_{VCT}| < 2^n \iff 3 \times 2^{n-k+1} - k - 2 > 6C > k - 7$$

5 Necessary and Sufficient Conditions for LMS Faithful Rounding

Error bounds for the LMS scheme were first reported in [28]:

$$-2^{n-1} - \frac{1}{9}\left(2^{k-4}\left(24k - 19 + 3(-1)^k\right) - 3 + 4(-1)^k\right) \leq \varepsilon_{LMS} \leq \frac{2^{n-k-1}}{9}\left(2^k(3k+1) + 8(-1)^k\right)$$

As stated in [28], in absolute value, the most negative error dominates. From this condition we can derive the necessary and sufficient condition for faithful rounding of the LMS scheme:

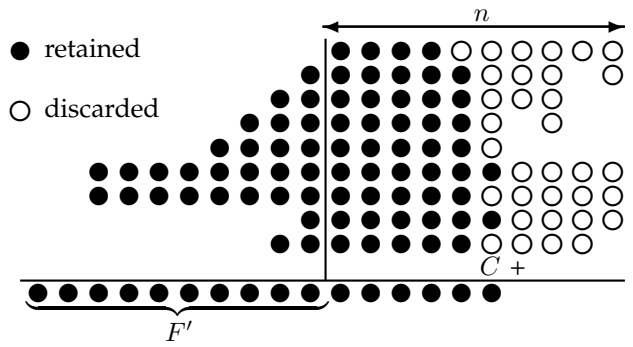

Fig. 2. Structure of an arbitrary array truncation scheme.

Theorem 5.1: The LMS Theorem
The necessary and sufficient condition for the LMS scheme to be faithfully rounded is:

$$|\varepsilon_{LMS}| < 2^n \iff 9 \times 2^{n-k+1} > 6k + 3 + (-1)^k$$

6 Sufficient Conditions for the Faithful Rounding of an Arbitrary Array

There is a range of multiplier arrays found throughout the literature, AND arrays, Booth arrays of various radices, MUX arrays as well merged arrays performing multiply-add or sums-of-products. It would be useful to be able to truncate an arbitrary array such that the result is faithfully rounded. Given we are considering an arbitrary array, we cannot exploit any *a priori* correlations found within the array. Thus we proceed with a strategy akin to a CCT scheme, consider a truncated arbitrary array as in Figure 2.

We assume that each partial product bit can vary independently and takes values $\{0, 1\}$. We discard some partial product bits, we call this set $\triangle$ and compensate by a fixed additive constant C. We wish the scheme to return F' which should be a faithful rounding of the true full summation F when the least significant n bits are ignored. Algebraically, F can be defined as:

$$F' = 2^n \left\lfloor \frac{F - val(\triangle) + C}{2^n} \right\rfloor$$

where $val(\triangle)$ is the value of all the elements in $\triangle$ while respecting their binary weight. The error introduced by performing this approximation is:

$$\begin{aligned}\varepsilon &= F - F' \\ &= ((F - val(\triangle) + C) \mod 2^n) + val(\triangle) - C\end{aligned}$$

We can bound this error by noting the modulo term ranges between 0 and $2^n - 1$, note that these bounds may not be tight due to lack of knowledge of the array:

$$-C \leq \varepsilon \leq 2^n - 1 + val(\triangle) - C$$

For the scheme to be faithfully rounded then:

$$\begin{aligned}&|\varepsilon| < 2^n \\ &val(\triangle) < C + 1 \leq 2^n\end{aligned}$$

Setting C to its maximal possible value, $2^n - 1$, places the least restriction on $\triangle$. Our goal is to minimise the cost

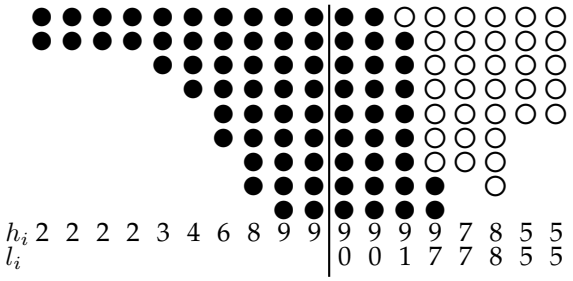

Fig. 3. Illustration of h_i of l_i.

of implementing the truncated array while maintaining faithful rounding, we use the heuristic that summing fewer partial product bits will result in the smallest implementation cost. Therefore we wish to maximise the number of elements in $\triangle$, we notate this as $|\triangle|$. Our optimisation problem then becomes:

$$\begin{aligned} \max \quad & |\triangle| \\ s.t. \quad & val(\triangle) < 2^n \end{aligned}$$

To solve this optimisation problem we introduce variables h_i, the height of the array in column i and l_i, the number of bits we truncate from column i; example values for Figure 2 are illustrated in Figure 3.

Note that the optimisation places no ordering on the bits in each column, merely their number. Our optimisation problem then becomes:

$$\begin{aligned} \max \quad & \sum_{i=0}^{n-1} l_i \\ s.t. \quad & \sum_{i=0}^{n-1} l_i 2^i < 2^n \\ & l_i \leq h_i \end{aligned}$$

Let k be the largest number of least significant columns we could truncate while maintaining faithful rounding, more precisely (using the notation $\max(k : cond)$ which returns the largest value of k which satisfies the condition $cond$):

$$k = \max\left(k : \sum_{i=0}^{k-1} h_i 2^i < 2^n\right)$$

As we shall see, the answer to the optimisation problem is closely related to k. Let l_i^{opt} be the optimal values of l_i which maximise the objective function. The following lemmas contribute to the solution of the optimisation problem.

6.1 Lemma 1: $l_i^{opt} = h_i$ for $i < k$

Proceeding by contradiction:

- If $l_i^{opt} = 0$ for $i \geq k$ and there exists $j < k$ such that $l_j^{opt} < h_j$ then by the definition of k we can increase l_j to h_j thus increasing the objective while not violating the constraint.
- If there exists $i \geq k$ and $l_i^{opt} > 0$ and $j < k$ with $l_j^{opt} < h_j$ then we can decrement l_i^{opt} and increment l_j^{opt}. The objective is unchanged and the constraint is still met as the left hand side of the constraint is reduced by $2^i - 2^j > 0$.

Conclude that if there exists a supposedly optimal set of values for l_i such that $l_i^{opt} < h_i$ for some $i < k$, then by repeated application of the second point, truncations in column k or above can be exchanged for truncations in the least significant k columns. If all the truncations occur in the least significant k columns then these can include all partial product bits of the k columns, by the definition of k. Hence we may assume that optimal l_i values satisfy $l_i^{opt} = h_i$ for $i < k$.

Restate the optimisation problem as a consequence of this lemma:

$$\begin{aligned} \max \quad & \sum_{i=k}^{n-1} l_i \\ s.t. \quad & \sum_{i=0}^{n-k-1} l_{k+i} 2^i < 2^{n-k} - \sum_{i=0}^{k-1} h_i 2^{i-k} \\ & l_i \leq h_i \end{aligned}$$

It is useful to note that by the definition of k:

$$\sum_{i=0}^{k-1} h_i 2^i < 2^n \leq \sum_{i=0}^{k} h_i 2^i$$

$$0 < 2^{n-k} - \sum_{i=0}^{k-1} h_i 2^{i-k} \leq h_k$$

So we can qualify the optimisation problem as:

$$\begin{aligned} \max \quad & \sum_{i=k}^{n-1} l_i \\ s.t. \quad & \sum_{i=0}^{n-k-1} l_{k+i} 2^i < 2^{n-k} - \sum_{i=0}^{k-1} h_i 2^{i-k} \leq h_k \\ & l_i \leq h_i \end{aligned}$$

6.2 Lemma 2: $l_i^{opt} = 0$ for $i > k$

Proceeding by contradiction: say there exists $j > k$ such that $l_j^{opt} > 0$ then that implies that the constraint term contains terms of the following form:

$$... + l_j^{opt} 2^{j-k} + l_k^{opt} < 2^{n-k} - \sum_{i=0}^{k-1} h_i 2^{i-k} \leq h_k$$

If we were to make the transformations $l_j \to l_j - 1$ and $l_k \to l_k + 2^{j-k}$ then the objective function is strictly increased and the first constraint is still maintained. However does the new l_k still satisfy $l_k \leq h_k$? The first constraint is bounded by h_k hence it was already true that:

$$\begin{aligned} l_j^{opt} 2^{j-k} + l_k^{opt} &< h_k \\ (l_j^{opt} - 1)2^{j-k} + (l_k^{opt} + 2^{j-k}) &< h_k \\ l_k^{opt} + 2^{j-k} &< h_k \end{aligned}$$

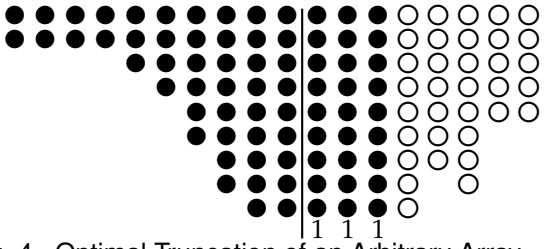

Fig. 4. Optimal Truncation of an Arbitrary Array.

Hence the transformation still results in a feasible l_k. Conclude optimal values of l_i for $i > k$ are all zero. This lemma shows that if there is a set of supposedly optimal values for l_i which have truncations in a column above k then these can be exchanged for more truncations in column k.

Restating the optimisation problem as a result of this lemma:

$$\begin{aligned} &\max \quad l_k \\ &s.t. \quad l_k < 2^{n-k} - \sum_{i=0}^{k-1} h_i 2^{i-k} \leq h_k \end{aligned}$$

Whose trivial solution is:

$$l_k^{opt} = \left\lceil 2^{n-k} - 1 - \sum_{i=0}^{k-1} h_i 2^{i-k} \right\rceil$$

6.3 Faithfully Rounded Array Theorem

We can now state the result of the optimisation problem:

Theorem 6.1: Faithfully Rounded Array Theorem
The optimal truncations l_i for an array with heights h_i returning a faithfully rounded result to the nth column are:

$$l_i^{opt} = \begin{cases} h_i & i < k \\ \left\lceil 2^{n-k} - 1 - \sum_{j=0}^{k-1} h_j 2^{j-k} \right\rceil & i = k \\ 0 & i > k \end{cases}$$
$$\text{where } k = \max\left(k : \sum_{j=0}^{k-1} h_j 2^j < 2^n \right)$$

Given the uneven truncation of the optimal form we term truncations performed using this method as *ragged*. As an example if we take Figure 3 we have $n = 8$ and array heights for the least significant 8 columns $\{9, 9, 9, 9, 7, 8, 5, 5\}$. Computing k gives 5 and $l_5 = 0$. The optimal truncations can be seen in Figure 4. Recall that the additive constant is always $2^n - 1$, therefore a 1 needs to be added to every column (it is not added to the least significant k columns as its addition will have no impact).

6.4 Ragged Truncated Multipliers - AND Array (RAT)

Applying this technique to a traditional AND array multipliers, in the case of the multiplication of two unsigned n bit numbers with a faithfully rounded n bit output; the array height of the ith column in the least significant n columns is $i + 1$. Applying the Faithfully Rounded Array Theorem:

$$l_i^{opt} = \begin{cases} i+1 & i < k \\ \left\lceil 2^{n-k} - 1 - \sum_{j=0}^{k-1} (j+1) 2^{i-k} \right\rceil & i = k \\ 0 & i > k \end{cases}$$
$$\text{where } k = \max\left(k : \sum_{j=0}^{k-1} h_j 2^j < 2^n \right)$$

Simplifying we get:

$$l_i^{opt} = \begin{cases} i+1 & i < k \\ 2^{n-k} - k & i = k \\ 0 & i > k \end{cases}$$
$$\text{where } k = \max\left(k : (k-1) 2^k < 2^n \right)$$

As an example consider $n = 12$, then $k = 8$ and $l_8 = 8$, as illustrated in Figure 5.

Note that the truncations into column k can be chosen such that the resultant truncated multiplier is still commutative.

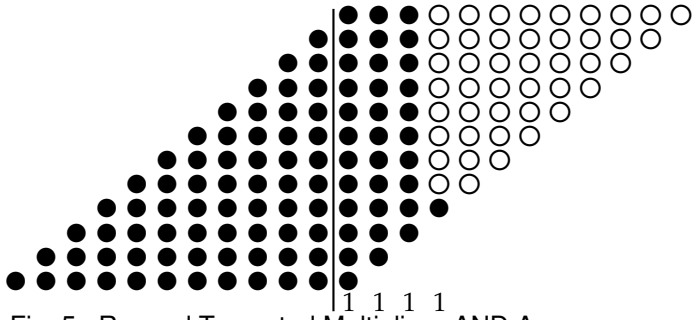

Fig. 5. Ragged Truncated Multiplier - AND Array.

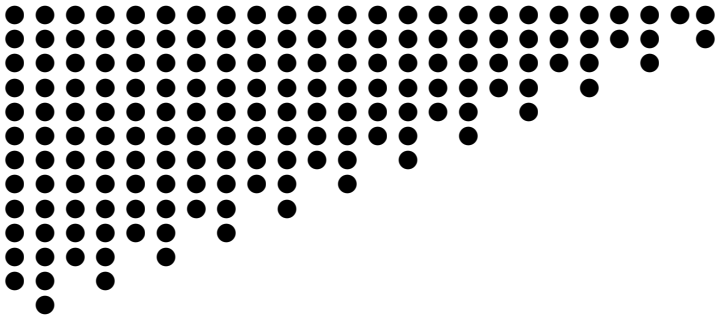
Fig. 6. Radix-4 Booth Array - Least Significant Columns.

6.5 Ragged Truncated Multipliers - Booth Array (RBT)

In the case of a radix-4 Booth array multipliers. For the multiplication of two unsigned n bit numbers with a faithfully rounded n bit output, the least significant n columns of the multiplier array take the form as in Figure 6.

Given the specific structure of the array we can compute the maximal value of any least significant k

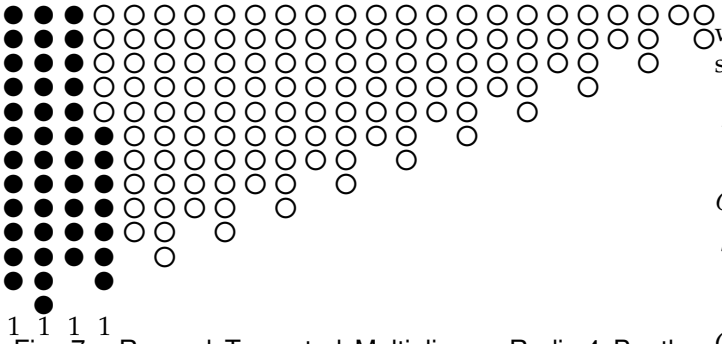

Fig. 7. Ragged Truncated Multipliers - Radix-4 Booth Array.

columns of the array as:

$$\sum_{i=0}^{k-1} h_i 2^i = \left\lfloor \frac{k+1}{2} \right\rfloor 2^k$$

Now applying the Faithfully Rounded Array Theorem:

$$l_i^{opt} = \begin{cases} h_i & i < k \\ \left\lceil 2^{n-k} - 1 - \left\lfloor \frac{k+1}{2} \right\rfloor \right\rceil & i = k \\ 0 & i > k \end{cases}$$
$$\text{where } k = \max\left(k : \left\lfloor \frac{k+1}{2} \right\rfloor 2^k < 2^n \right)$$

Simplifying we get:

$$l_i^{opt} = \begin{cases} h_i & i < k \\ 2^{n-k} - 1 - \left\lfloor \frac{k+1}{2} \right\rfloor & i = k \\ 0 & i > k \end{cases}$$
$$\text{where } k = \max\left(k : (k+1)2^k < 2^{n+1} \right)$$

As an example we can truncate the example in Figure 6 where $n = 24$, in which case $k = 20$ and $l_{20} = 5$. The resultant truncation is illustrated in Figure 7.

Note that truncated Booth multipliers are non commutative.

7 Constructing Faithfully Rounded Multipliers

We now have the necessary and sufficient conditions for faithful rounding of three truncation schemes as well as the construction of ragged AND and Booth arrays. For the three original truncation schemes, we aim to create the lowest cost faithfully rounded designs. We use the design heuristic that larger k values remove more partial product bits and are thus more efficient to implement. Varying C has extremely limited impact on hardware resources used, however as an heuristic we assume that a small Hamming weight and small numerical value is desirable, so let $minHamm(a, b)$ return number of smallest value within the integers with smallest Hamming weight which exist in the interval $[a, b]$. The following values for k and C thus guarantee faithful rounding while minimising hardware costs for the three truncation schemes CCT, VCT & LMS:

$$\begin{aligned} k_{CCT} &= \max\left(k : \exists \quad C \quad s.t. \quad 2^{n-k} > C > k-2\right) \\ &= \max\left(k : 2^n > (k-1)2^k\right) \\ C_{CCT} &= minHamm(k_{CCT} - 1, 2^{n-k_{CCT}} - 1) \\ k_{VCT} &= \max\left(k : \exists C s.t. 3 \times 2^{n-k+1} - k - 2 > 6C > k - 7\right) \\ &= \max\left(k : 3 \times 2^n \geq k2^k\right) \\ C_{VCT} &= minHamm\left(\left\lceil \frac{k}{6} \right\rceil - 1, \left\lfloor \frac{3 \times 2^{n-k+1} - k - 3}{6} \right\rfloor\right) \\ k_{LMS} &= \max\left(k : 9 \times 2^{n-k+1} > 6k + 3 + (-1)^k\right) \end{aligned}$$

Summarising the ragged truncation schemes we have k, the number of least significant columns to remove, C, the constant added into the array and l, the number of bits to remove from column k:

$$\begin{aligned} k_{RAT} &= \max\left(k : 2^n > (k-1)2^k\right) \\ k_{RBT} &= \max\left(k : 2^{n+1} > (k+1)2^k\right) \\ C_{RAT} &= 2^n - 2^{k_{RAT}} \\ C_{RBT} &= 2^n - 2^{k_{RBT}} \\ l_{RAT} &= 2^{n-k_{RAT}} - k_{RAT} \\ l_{RBT} &= 2^{n-k_{RBT}} - 1 - \left\lfloor \frac{k_{RBT}+1}{2} \right\rfloor \end{aligned}$$

Figure 8 contains the number of fewer partial product bits than CCT that the VCT, LMS and RAT schemes contain for $n = 8..32$. Note that for certain regions, particularly around $n = 16$ and $n = 32$, the RAT scheme has the fewest partial product bits and for $n = 24$ the LMS and VCT scheme have the minimal count. It can be shown analytically that $k_{VCT} \geq k_{LMS}$, so the VCT scheme will generally have no more partial product bits than LMS. RAT has been designed to minimise partial product bit count without reference to bit correlations, its error bounds may not be tight. In contrast, LMS and VCT have tight error bounds but a different architecture. Hence the partial product bit counts and synthesis will not strictly favour one architecture over another. Note that the values for k are strictly less than $n-1$, therefore the truncations are independent of the input bits a_{n-1} and b_{n-1} in the cases of the truncated AND arrays. So if we had considered a and b to be two's complement then the analysis that gave rise to the necessary and sufficient conditions for faithful rounded would be unchanged. Therefore a standard AND array for two's complement inputs can be truncated in an identical fashion to the unsigned multipliers presented here.

All these functions are simple enough to be embeddable directly into HDL code. As such it is possible to create fully parameterisable RTL whose only input parameter is n, where HDL functions compute the relevant k, C and l signals (where appropriate) and a partial product array can be formed and summed. Example VHDL fragments can be found in the supplemental material for calculating k_{CCT}, k_{LMS}, a $minHamm$ function given an

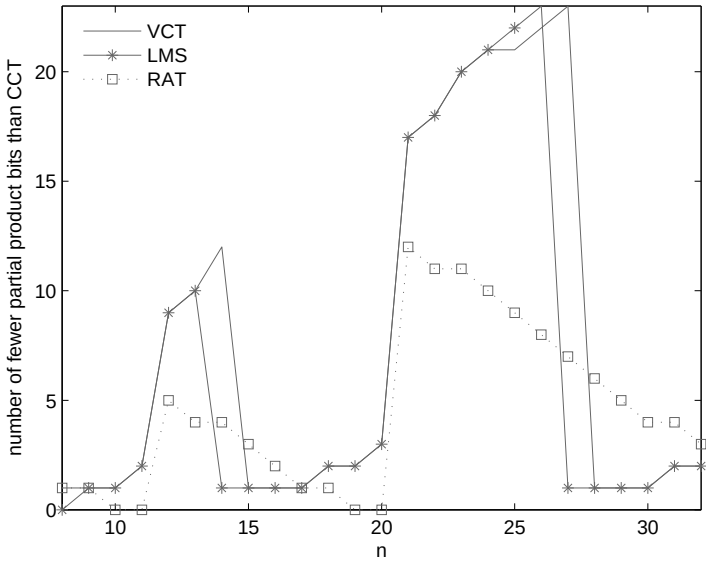

Fig. 8. Comparison of the Number of Partial Product Bits for the Faithfully Rounded Multipliers.

interval $[a, b]$ and an example of how a partial product array may be created and summed.

8 Experimental Benchmarks

We created five parameterisable pieces of HDL code that return a faithful rounding of an unsigned multiplication result as well as a reference multiplier which return the correctly rounded round towards nearest, ties to even (RTE) result, in order to see the benefit of truncation. Note that the construction of the schemes CCT, VCT and LMS as presented in the previous sections have the fewest partial product bits of that architecture which are faithfully rounded. This is due to the fact that their error bounds are tight. The RAT and RBT schemes do not necessarily have tight error bounds, but are of interest given the generality of their construction. There are schemes found within the literature whose error bounds are not tight, but can still be used to produce HDL which guarantees faithful rounding. We include these in the synthesis comparisons, there is a variant on VCT found in [29] and a CCT version of Booth radix-4 [10]. Note we do not compare against [26], [12], [13], [14] & [15] as their approach cannot be embedded into HDL as they require offline compensation circuit construction or modifications to the synthesis process. We performed synthesis comparisons for multipliers of size n =16, 24 & 32. Synopsys Design Compiler 2009.06-SP5 in ultra mode using the TSMC 65nm library Tcbn65lpwc was used for the synthesis experiments. We requested the synthesis tool to synthesize the design to achieve different delays; by applying Boolean optimization techniques and utilizing different standard cells, Design Compiler seeks the design with smallest area that meets the required delay. Thus we can see the full delay and area trade off of the various multipliers. These experiments were performed for each value of n, generating a range of delay and area points. Truncated multipliers based upon an AND array are commutative and are compared against an AND array implementation of RTE in Figures 9, 10 and 11 (note the split y-axis). Truncated multipliers based upon a radix-4 Booth array are non-commutative and are compared against a radix-4 Booth array implementation of RTE in Figures 12, 13 and 14.

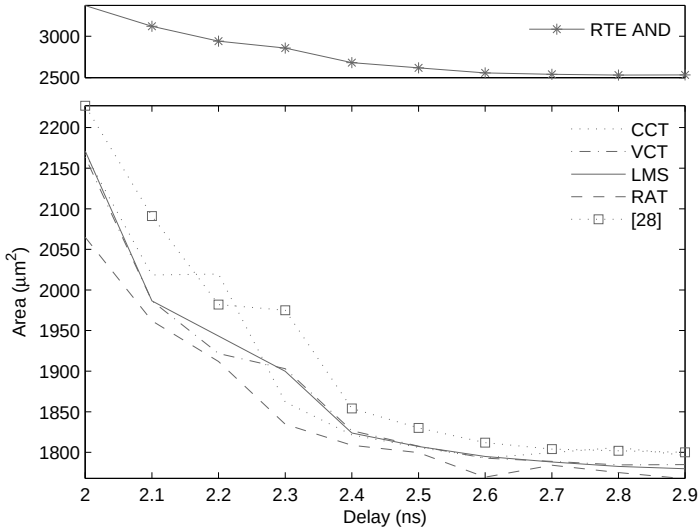

Fig. 9. Area/Delay Comparisons of Faithfully Rounded AND Array Multipliers n=16.

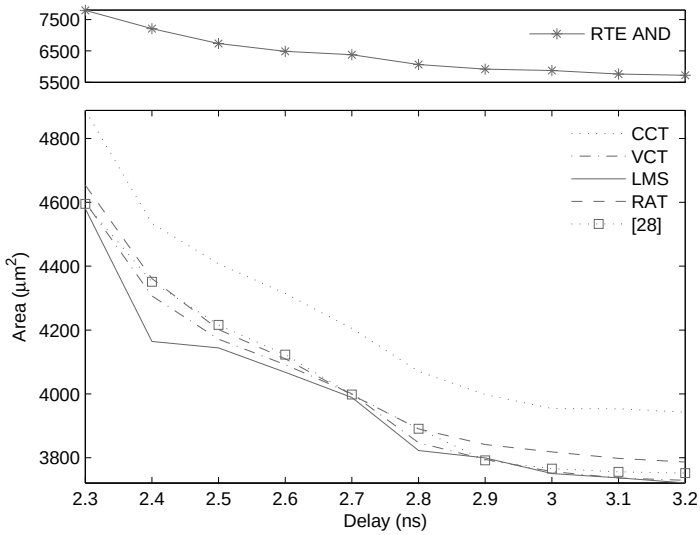

Fig. 10. Area/Delay Comparisons of Faithfully Rounded AND Array Multipliers n=24.

These figures demonstrate that truncated AND array multipliers can provide an area benefit of 30-43% over the correctly rounded, RTE multiplier, which increases as n grows. As predicted from the inspecting the partial product counts, the RAT scheme consistently exhibits the smallest area for $n = 16, 32$, whereas LMS and VCT dominates for $n = 24$. Truncated Booth arrays, in the

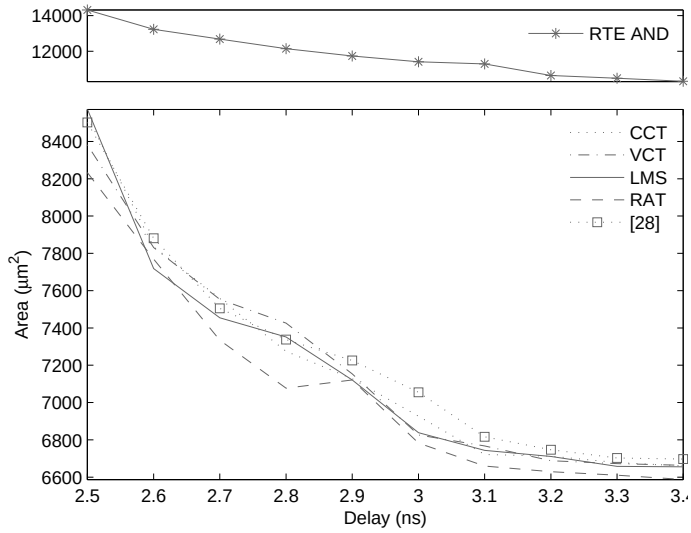

Fig. 11. Area/Delay Comparisons of Faithfully Rounded AND Array Multipliers n=32.

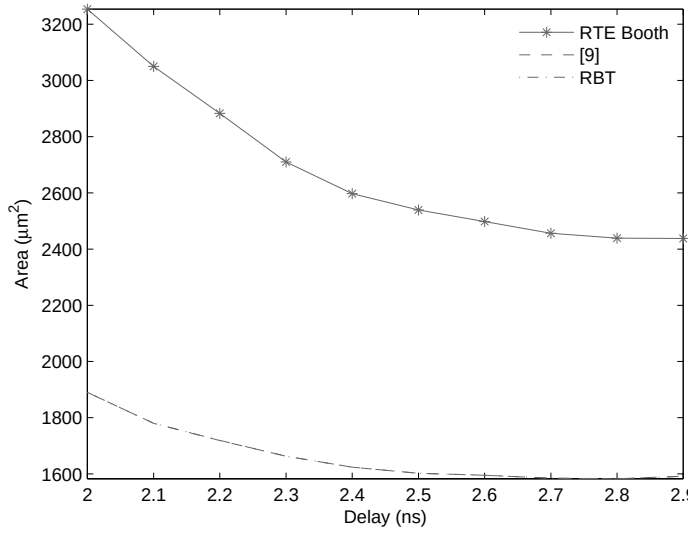

Fig. 12. Area/Delay Comparisons of Faithfully Rounded Booth Array Multipliers n=16.

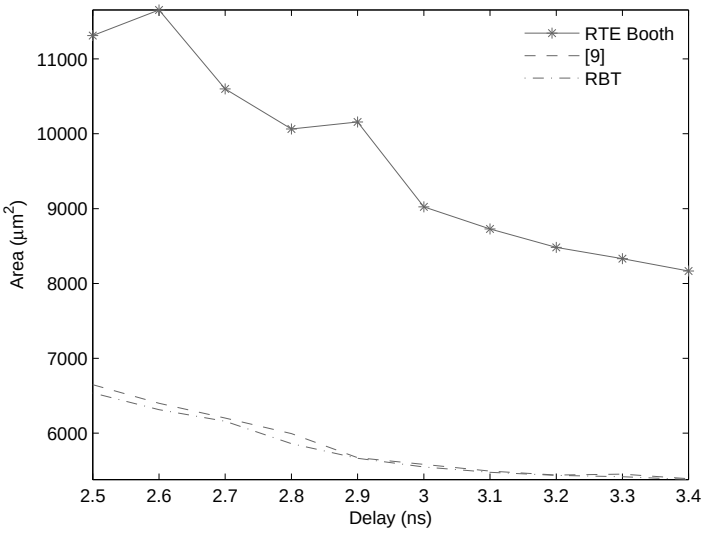

Fig. 13. Area/Delay Comparisons of Faithfully Rounded Booth Array Multipliers n=24.

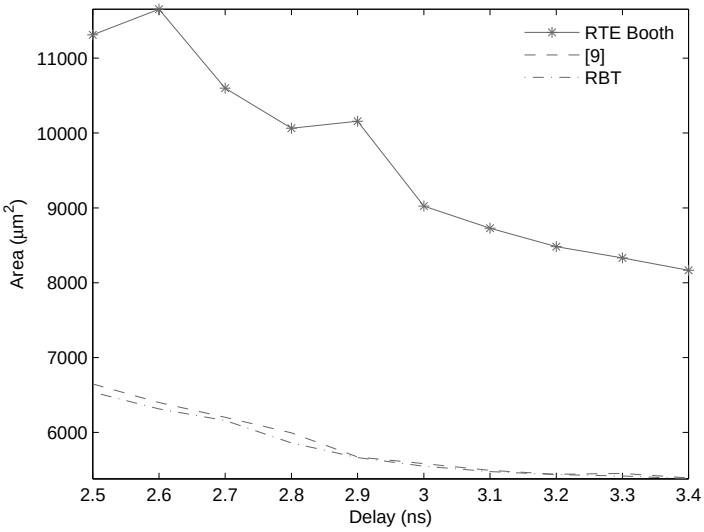

Fig. 14. Area/Delay Comparisons of Faithfully Rounded Booth Array Multipliers n=32.

form of the RBT design, offers a consistent improvement of 34-46% area compared to a radix-4 Booth RTE design. The RBT scheme is slightly superior to [10], due to the fact that [10] is CCT applied to a Booth array and RBT removes at least as many partial products as a CCT approach. It is interesting to note that within the set of non-Booth truncated multipliers none of the schemes has a strictly superior area for all bit widths, designer will have to choose based upon their particular hardware, accuracy and commutativity requirements. The synthesis tool uses a timing, area and driven parallel reduction of the arrays using a range of compression cells (full-adders, half-adders and 4-to-2 compressors). The final carry propagate adder will be formed by a hybrid of adder architectures optimised to the delay profile of the intermediate carry-save representation resulting from the array reduction. This will typically comprise of ripple adders for the least significant bits, carry look-ahead and parallel prefix adders for the most significant bits.

9 Application to the Construction of Faithfully Rounded Floating-Point Multipliers

Here we generalise the work found in [21], by showing how any faithfully rounded fixed point multiplier architecture can be used in floating-point multiplication.

Floating-point numbers are represented by the triple: sign, exponent and mantissa, $\{s, exp, mant\}$. Excluding denormals and exception cases, these numbers are interpreted as $(-1)^s 2^{exp-bias} 1.mant$. Consider multiplication of A & B represented by $\{sa, expa, manta\}$ & $\{sb, expb, mantb\}$ respectively, returning Y in the form $\{sy, expy, manty\}$. The equations governing the outputs are:

$$\begin{aligned} sy &= sa \oplus sb \\ expy &= expa + expb - bias \\ 1.manty &= 1.manta \times 1.mantb \end{aligned}$$

These equations need slight modification given that $1.manta \times 1.mantb$ produces numbers in the interval [1,4) and so a one bit renormalisation may be required as well as rounding. The fixed point steps to producing $manty$, for n bit mantissas, are thus:

$$\begin{aligned} a_{n:0} &= 2^n + manta // \text{ adding in the implicit one} \\ b_{n:0} &= 2^n + mantb \\ c_{m-1:0} &= multFR(a, b) \\ manty &= \textbf{if} \quad (c_{m-1} == 1) \quad \textbf{then} \quad c_{m-2:m-n-1} \\ &\qquad\qquad\qquad\quad \textbf{else} \quad c_{m-3:m-n-2} \end{aligned}$$

where $multFR$ returns a faithful rounding of the top m bits of the multiplication of a and b. Now $multFR$ can be any of the truncation schemes we have already constructed. In order to construct the most hardware efficient floating point multiplier, a design with the smallest precision for the intermediate variable c is desirable. What is the smallest value of m such that the floating point multiplier is faithfully rounded?

9.1 A Faithfully Rounded Floating Point Multiplier is Guaranteed if $m = n + 2$

The proof splits into the following points:

- **Case:** $c_{n+1} = 0$ In this case $manty = c_{n-1:0}$ which is faithfully rounded due to definition of $multFR$ hence in this case the floating point multiplier is faithfully rounded.
- **Case:** $c_{n+1} = 1$ **and** $c_0 = 0$ If we say that c is a fixed point number $2.n$ in length and the infinitely precise answer is r. Then during renormalisation c_0 is removed hence r and c are related as follows:

  $$\begin{aligned} |r - c| &< 2^{-n} < 2^{-n+1} \\ |r - c| &< 2^{-n+1} \end{aligned}$$

  In this case one ulp is 2^{-n+1}, hence this meets the accuracy condition.
- **Case:** $c_{n+1} = 1$ **and** $c_0 = 1$ Then from the definition of $multFR$ we have

  $$\begin{aligned} |r - c| &< 2^{-n} \\ -2^{-n} < r - c &< 2^{-n} \\ 0 < r - \left(c - 2^{-n}\right) &< 2^{-n+1} \\ \left|r - \left(c - 2^{-n}\right)\right| &< 2^{-n+1} \end{aligned}$$

  In this case one ulp is 2^{-n+1}. Also, due to renormalisation, the answer we return is $c - 2^{-n}$. Due to this inequality we can see that, our result meets will be within one ulp and hence faithfully rounded.

In conclusion, if the following fixed point algorithm is used as part of a floating point multiplier the entire design will be faithfully rounded:

$$\begin{aligned} a_{n:0} &= 2^n + manta \\ b_{n:0} &= 2^n + mantb \\ c_{n+1:0} &= multFR(a, b) \\ manty &= \textbf{if} \quad (c_{n+1} == 1) \quad \textbf{then} \quad c_{n:1} \quad \textbf{else} \quad c_{n-1:0} \end{aligned}$$

We used this to construct parameterisable correct-by-construction faithfully rounded floating point multiplier HDL code. In the case of a single precision multiplier where the mantissa width $n = 23$ we performed a synthesis experiment with Synopsys Design Compiler 2009.06-SP5 in ultra mode using the TSMC 65nm library Tcbn65lpwc. We compared a round to nearest, even single precision floating point multiplier to a faithfully rounded floating point multiplier constructed using our fixed point truncated multipliers. The result is shown in Table 2. Our experiment targeted zero delay and we found a 30% area improvement for a slightly improved delay.

TABLE 2
Floating Point Multiplier RTE versus Faithful Rounding.

Rounding	Delay (ns)	Area (μm^2)
RTE	2.39	9975
Faithfully Rounded	2.32	7007

10 Conclusion

Necessary and sufficient closed form conditions for three multiplier truncation schemes have been derived as well as a method for faithfully rounding any array. In the past, the industrial adoption of such schemes has been hampered by the risk and/or time taken for exhaustive verification. As a result of the conditions derived in this paper, we have demonstrated a practical procedure for the synthesis of such multipliers and arrays with a rigorous guarantee of faithful rounding. Our approach covers three techniques found in the literature, analytically provides faithful rounding conditions for all three and also the worst-case error vectors for two schemes. We have also shown how an arbitrary array may be optimally truncated while returning a faithfully rounded result - by applying this to any array, whose summation results in a multiplication, we can automatically generate a myriad of correct-by-construction faithfully rounded multipliers. We have also shown a method for the creation of a correct-by-construction faithfully rounded floating point multiplier.

Acknowledgments

The authors would like to acknowledge Imagination Technologies Ltd for supporting this research.

REFERENCES

[1] E. Walters and M. Schulte, "Fast, bit-accurate simulation of truncated-matrix multipliers and squarers," in *Signals, Systems and Computers (ASILOMAR), 2010 Conference Record of the Forty Fourth Asilomar Conference on*, Nov. 2010, pp. 1139–1143.

[2] M. J. Schulte and E. E. Swartzlander, Jr., "Truncated multiplication with correction constant," in *Workshop on VLSI Signal Processing*, vol. 6, no. 20-22, Oct. 1993, pp. 388–396.

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

[4] 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, Feb. 1996, pp. 90–95.

[5] E. J. King and E. E. Swartzlander, Jr., "Data-dependent truncation scheme for parallel multipliers," in *Thirty-First Asilomar Conference on Signals, Systems & Computers*, vol. 2, no. 2-5, Nov. 1997, pp. 1178–1182.

[6] J. E. Stine and O. M. Duverne, "Variations on truncated multiplication," in *Euromicro Symposium on Digital System Design*, Sep. 2003, pp. 112–119.

[7] N. Petra, D. De Caro, V. Garofalo, E. Napoli, and A. Strollo, "Truncated binary multipliers with variable correction and minimum mean square error," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 57, no. 6, Jun. 2010, pp. 1312–1325.

[8] ——, "Design of fixed-width multipliers with linear compensation function," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 58, no. 5, May 2011, pp. 947–960.

[9] R. Michard, A. Tisserandt, and N. Veyrat-Charvillon, "Carry predicition and selection for truncated multiplication," in *IEEE Workshop on Signal Processing Systems Design and Implementation*, 2006, pp. 339–344.

[10] A. A. Katkar and J. E. Stine, "Modified booth truncated multipliers," in *Proceedings of the 14th ACM Great Lakes symposium on VLSI*, ser. GLSVLSI '04. New York, NY, USA: ACM, 2004, pp. 444–447.

[11] 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*, vol. 52, no. 6, Jun. 2005, pp. 299–303, .

[12] H.-A. Huang, Y.-C. Liao, and H.-C. Chang, "A self-compensation fixed-width booth multiplier and its 128-point fft applications," in *Circuits and Systems, 2006. ISCAS 2006. Proceedings. 2006 IEEE International Symposium on*, May 2006, pp. 3538–3541.

[13] Y.-H. Chen, T.-Y. Chang, and R.-Y. Jou, "A statistical error-compensated booth multipliers and its dct applications," in *TENCON 2010 - 2010 IEEE Region 10 Conference*, Nov. 2010, pp. 1146–1149.

[14] C.-Y. Li, Y.-H. Chen, T.-Y. Chang, and J.-N. Chen, "A probabilistic estimation bias circuit for fixed-width booth multiplier and its dct applications," *Circuits and Systems II: Express Briefs, IEEE Transactions on*, vol. 58, no. 4, Apr. 2011, pp. 215–219.

[15] Y.-H. Chen and T.-Y. Chang, "A high-accuracy adaptive conditional-probability estimator for fixed-width booth multipliers," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 59, no. 3, Mar. 2012, pp. 594–603.

[16] V. Garofalo, M. Coppola, D. De Caro, E. Napoli, N. Petra, and A. Strollo, "A novel truncated squarer with linear compensation function," in *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*, Jun. 2010, pp. 4157–4160.

[17] S. Datla, M. Thornton, and D. Matula, "A low power high performance radix-4 approximate squaring circuit," in *Application-specific Systems, Architectures and Processors, 2009. ASAP 2009. 20th IEEE International Conference on*, Jul. 2009, pp. 91–97.

[18] K.-J. Cho, W.-K. Kim, B.-K. Kim, and J.-G. Chung, "Design of low error fixed-width squarer," in *Signal Processing Systems, 2003. SIPS 2003. IEEE Workshop on*, Aug. 2003, pp. 213–218.

[19] S.-M. Kim, J.-G. Chung, and K. K. Parhi, "Low error fixed-width csd multiplier with efficient sign extension," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 50, no. 12, 2003, pp. 984–993.

[20] N. Petra, D. De Caro, A. Strollo, V. Garofalo, E. Napoli, M. Coppola, and P. Todisco, "Fixed-width csd multipliers with minimum mean square error," in *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*, Jun. 2010, pp. 4149–4152.

[21] K. E. Wires, M. J. Schulte, and J. E. Stine, "Variable-correction truncated floating point multipliers," in *Thirty-Fourth Asilomar Conference on Signals, Systems and Computers*, vol. 2, 2000, pp. 1344–1348.

[22] E. George Walters, III and M. J. Schulte, "Efficient function approximation using truncated multipliers and squarers," in *17th IEEE Symposium on Computer Arithmetic*, Jun. 2005, pp. 232–239.

[23] A. Strollo, N. Petra, and D. DeCaro, "Dual-tree error compensation for high performance fixed-width multipliers," *Circuits and Systems II: Express Briefs, IEEE Transactions on*, vol. 52, no. 8, Aug. 2005, pp. 501–507.

[24] K.-J. Cho, S.-M. Lee, S.-H. Park, and J.-G. Chung, "Error bound reduction for fixed-width modified booth multiplier," in *Signals, Systems and Computers, 2004. Conference Record of the Thirty-Eighth Asilomar Conference on*, vol. 1, Nov. 2004, pp. 508–512.

[25] V. Garofalo, N. Petra, and E. Napoli, "Analytical calculation of the maximum error for a family of truncated multipliers providing minimum mean square error," *Computers, IEEE Transactions on*, vol. 60, no. 9, Sep. 2011, pp. 1366–1371.

[26] H.-J. Ko and S.-F. Hsiao, "Design and application of faithfully rounded and truncated multipliers with combined deletion, reduction, truncation, and rounding," *Circuits and Systems II: Express Briefs, IEEE Transactions on*, vol. 58, no. 5, May 2011, pp. 304–308.

[27] R. Zimmermann, "Coding guidelines for datapath synthesis," https://www.synopsys.com/dw/doc.php/wp/coding_guidelines.pdf, Jul. 2005.

[28] V. Garofalo, "Truncated binary multipliers with minimum mean square error: analytical characterization, circuit implementation and applications," PhD, Universita degli Studi di Napoli Federico II, 2009.

[29] H. Park and E. E. Swartzlander, Jr., "Truncated multiplication with symmetric correction," in *Signals, Systems and Computers, 2006. ACSSC '06. Fortieth Asilomar Conference on*, Nov. 2006, pp. 931–934.

Theo A. Drane received the BA. degree (with honors) in the Mathematical Tripos from Emmanuel College, Cambridge, UK, in 2001. During which he was the three times winner of the Braithwaite-Batty Prize for mathematics and awarded an Honorary Bachelor Scholarship. He has worked for the Datapath consultancy Arithmatica and now leads Imagination Technologies' Datapath Group which focuses on implementation, optimisation, verification and validation of mathematically intensive hardware. He is currently undertaking an industrial PhD in conjunction with Imperial College London, UK.

Thomas M. Rose received the BA. degree (with honors) and the MMath degree in the Mathematical Tripos from University of Cambridge, Cambridge, UK, in 2011. Since September 2011, he has been a graduate hardware design engineer at Imagination Technologies Ltd. From 2009-2011 he was a Senior scholar at Trinity College, University of Cambridge.

George A. Constantinides (S'96-M'01-SM'08) received the M.Eng. degree (with honors) in information systems engineering and the Ph.D. degree from Imperial College London, London, UK, in 1998 and 2001, respectively. Since 2002, he has been with the faculty at Imperial College London, where he is currently Reader (Associate Professor) in Digital Systems and Head of the Circuits and Systems research group. He is a recipient of the Eryl Cadwaladar Davies Prize for the best doctoral thesis in Electrical Engineering at Imperial College (2001), an Imperial College Research Excellence Award (2006), and a fellowship from the EPSRC. He was Programme Co-Chair of the IEEE International Conference on Field-Programmable Technology (FPT) in 2006 and Field Programmable Logic (FPL) in 2003 and will be programme (general) chair of the ACM International Symposium on Field-Programmable Gate Arrays in 2014 (2015). He currently serves on the programme committees of several international conferences, including FPGA, FPL, and FPT. He has published over 150 research papers in peer refereed journals and international conferences. Dr Constantinides is a Senior Member of the IEEE and a Fellow of the British Computer Society.