Correctly Rounded Constant Integer Division via Multiply-Add

Theo Drane and Wai-chuen Cheung
Imagination Technologies Ltd
Home Park Estate, Kings Langley
Hertfordshire, WD4 8LZ
Email: {theo.drane,wai-chuen.cheung}@imgtec.com

George Constantinides
Department of Electrical and Electronic Engineering,
Imperial College London,
Exhibition Road, London, SW7 2BT
Email: g.constantinides@imperial.ac.uk

***Abstract*—Implementing integer division in hardware is expensive when compared to multiplication. In the case where the divisor is a constant, expensive integer division algorithms can be replaced by cheaper integer multiplications and additions. This paper presents the conditions for multiply-add schemes to perform correctly rounded unsigned invariant integer division under one of three rounding modes. We propose a heuristic to explore the space of implementations meeting the conditions we derive. Experiments show that an average speed up of 20% and area reduction of 50% can be achieved compared to existing correctly rounded approaches. Extension to two's complement numbers is also presented.**

I. Introduction

Creating bespoke hardware for integer division for a given invariant integer arises in a variety of situations, *e.g.* bespoke filtering, base conversions, certain caching algorithms, arithmetic with number formats where divisors of the form $2^n - 1$ are common. Previous work has mainly focused on software implementations, given the lack of native integer division instructions within existing hardware [1]. Where hardware implementations are considered they are invariably sequential in nature. Certain divisors have been expanded into infinite products which translate into multiple shift and add instructions [2], [3]. An alternative proposed in [4], [5], [6], has been to replace the division by a single multiply-add instruction, computing $x/d \approx \lfloor (ax+b)/2^k \rfloor$ for suitable values of a, b and k (note that division by 2^k is zero cost).

The results in [7] show how only n bit multiply-add operations are required and are thus optimal from a software perspective. However this work suffices with an n by n bit multiplication as opposed to finding the smallest bit widths as would be required by a hardware implementation. Invariant integer division using a serial multiplier is developed in [8]. In this paper we seek optimised hardware for computing invariant integer division. Parallel multipliers and adders are appropriate for low latency hardware construction. Our stated problem is to find integers a, b and k such that:

$$Round\left(\frac{x}{d}\right) = \left\lfloor \frac{ax+b}{2^k} \right\rfloor \quad x \in [0, 2^n - 1]$$

In this paper we will address three rounding modes, round towards zero (RTZ), round to nearest, even (RTE) and faithful rounding (FR) (the latter only requires that the difference between the true answer and the returned answer to be less than one in absolute value). We will also assume that d is an odd integer larger than one and note that without loss of generality we can assume that a is odd.

The contributions of this paper are:

- necessary and sufficient conditions for the multiply-add schemes to meet round towards zero, round to nearest, even and faithful rounding conditions,
- a hardware optimisation heuristic to explore the space,
- extension to division of two's complement numbers, and
- experimental synthesis comparison of the three rounding schemes against previous work.

II. Conditions for Round Towards Zero Schemes

In this case we require and can re-express the RTZ requirement as follows:

$$\begin{aligned} &\left\lfloor \frac{x}{d} \right\rfloor = \left\lfloor \frac{ax+b}{2^k} \right\rfloor \qquad \forall x \in [0, 2^n - 1] \\ &0 \le \frac{ax+b}{2^k} - \left(\frac{x - (x \mod d)}{d} \right) < 1 \\ &x\left(1 - \frac{ad}{2^k}\right) - \frac{bd}{2^k} \le x \mod d < x\left(1 - \frac{ad}{2^k}\right) - \frac{bd}{2^k} + d \end{aligned} \tag{1}$$

Thus we require $x \mod d$ (a sawtooth function in x) to be bounded above and below by two lines of equal slope. Figure 1 shows an example for $d = 11$ and $n = 6$. In this example:

$$\begin{aligned} &\frac{3x - 176}{256} \le x \mod 11 < \frac{3x + 2640}{256} && x \in [0, 63] \\ &\text{Thus } \left\lfloor \frac{x}{11} \right\rfloor = \left\lfloor \frac{23x + 16}{2^8} \right\rfloor && x \in [0, 63] \end{aligned}$$

and division by 11 has been replaced by multiplication by 23, addition of 16 and a (free) truncation. In general it is necessary and sufficient to check that the upper bound of (1) is met for the peaks of $x \mod d$ which occur at $x = md - 1$ where $0 < m \le \lfloor 2^n/d \rfloor$:

$$\begin{aligned} &d - 1 < (md - 1)\left(1 - \frac{ad}{2^k}\right) - \frac{bd}{2^k} + d \\ &m(ad - 2^k) < a - b \end{aligned}$$

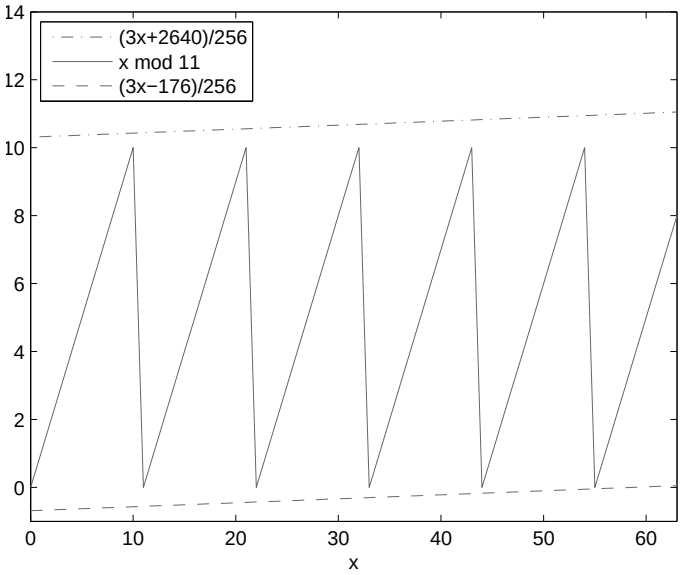

Fig. 1. Example Linearly Bounded Sawtooth Function.

and the lower bound of (1) is met by the troughs which occur at $x = md$ for $0 \leq m \leq \lfloor 2^n/d \rfloor$.

$$md\left(1 - \frac{ad}{2^k}\right) - \frac{bd}{2^k} \leq 0$$
$$m(2^k - ad) \leq b$$

Now given that a and d are odd then $ad - 2^k \neq 0$. Depending on the sign of $ad - 2^k$ different values of m will stress these inequalities. We may conclude that the necessary and sufficient condition for the design to be round towards zero compliant is:

$$\begin{array}{ll} -\dfrac{b+1}{\lfloor 2^n/d \rfloor} < ad - 2^k < a - b & \text{if } ad - 2^k < 0 \\ ad - 2^k < \dfrac{a-b}{\lfloor 2^n/d \rfloor} & \text{if } ad - 2^k > 0 \end{array}$$

III. Conditions for Round To Nearest, Even and Faithfully Rounded Schemes

In the case of round to nearest, even, special care for rounding must be taken when x/d takes half integer values, namely when $x/d = A + 1/2$ for some integer A. However, rewriting this as $2(x - Ad) = d$ we see this only holds for even d and d is odd by assumption. Thus we may write:

$$\left\lfloor \frac{x}{d} + \frac{1}{2} \right\rfloor = \left\lfloor \frac{ax+b}{2^k} \right\rfloor$$

By following a similar argument found in Section II one can derive the following necessary and sufficient conditions for the design to be round to nearest, even:

$$\begin{array}{l} \text{if } ad - 2^k < 0 \\ \quad \dfrac{a(d-1) - 2b - 1}{2\lfloor (2^{n+1} + d - 3)/2d \rfloor} < ad - 2^k < a\left(\dfrac{d+1}{2}\right) - b \\ \text{if } ad - 2^k > 0 \\ \quad a\left(\dfrac{d-1}{2}\right) - b \leq ad - 2^k < \dfrac{a(d+1) - 2b}{2\lfloor (2^{n+1} + d - 1)/2d \rfloor} \end{array}$$

Faithful rounding requires that the exact answer be returned if it is representable, otherwise it is permitted to return either of the two answers which are immediately above and below the true result. This means that for $x = md$ where $0 \leq m \leq \lfloor 2^n/d \rfloor$ the correct answer of m must be returned:

$$m = \left\lfloor \frac{amd + b}{2^k} \right\rfloor$$

Otherwise the constraint that either the answer above or below can be returned can be summarised as:

$$0 \leq \frac{ax+b}{2^k} - \left\lfloor \frac{x}{d} \right\rfloor < 2$$

Proceeding in a similar manner to Section II one can derive the following necessary and sufficient for the design to be faithfully rounded:

$$\begin{array}{ll} \lfloor 2^n/d \rfloor (2^k - ad) \leq b < 2^k & \text{if } ad - 2^k < 0 \\ \lfloor 2^n/d \rfloor (ad - 2^k) < 2^k - b & \text{if } ad - 2^k > 0 \end{array}$$

IV. Hardware Heuristic

It is assumed that minimal hardware implementations will be formed when the number of partial product bits in $ax + b$ is minimal. This count is driven by the width of the partial product array which is $n + k$ in length. So we will choose to minimise k in the first instance. It will turn out that having done this there will be no freedom in the value of a. However there will be an interval in which b may reside. In an effort to minimise the number of partial product bits we first find the set of valid b values whose Hamming weight is minimal and then of those choose the one with smallest value, working on the assumption that this will limit the effect on the partial product array height which is normally highest in the middle of the array. In the following we define the function which finds this value for numbers in the interval $[a, b]$ as minh(a, b) and can be computed as follows where a and b are p bits in length:

```
int minh(int a, int b)
c = 0
for i = p − 1 down to 0 begin
  if a[i] == b[i] then c[i] = a[i];   a[i] = 0;
  else c += 2^⌈log2 a⌉;   break;
  endif
end
return c
```

V. Optimal Round Towards Zero Scheme

We can now apply the hardware heuristic to the necessary and sufficient conditions for a multiply-add scheme to perform RTZ rounding. We split this process on the whether $ad - 2^k$ is positive or negative:

A. Optimal RTZ Scheme when $ad - 2^k > 0$

In this case we require:

$$ad - 2^k < \frac{a-b}{\lfloor 2^n/d \rfloor}$$

Now note that the right hand side is strictly decreasing in b. So for any valid a, b and k we can always set $b = 0$ and the condition will be met. Thus our condition reduces to:

$$\begin{aligned}(ad - 2^k)\lfloor 2^n/d \rfloor &< a \\ \frac{2^k}{d} < a &< \frac{2^k \lfloor 2^n/d \rfloor}{d\lfloor 2^n/d \rfloor - 1} \end{aligned} \tag{2}$$

Given that a must be an integer, we have a formula for k_{opt}, the smallest possible feasible value of k:

$$\begin{aligned} k_{opt} &= \min\left(k : \frac{1}{2^k}\left\lceil \frac{2^k}{d} \right\rceil < \frac{\lfloor 2^n/d \rfloor}{d\lfloor 2^n/d \rfloor - 1}\right) \\ &= \min\left(k : \frac{2^k}{(-2^k) \mod d} > d\left\lfloor \frac{2^n}{d} \right\rfloor - 1\right) \end{aligned}$$

So by the construction of k_{opt}, $a = \lceil 2^{k_{opt}}/d \rceil$ is valid as it satisfies (2). Now consider that k_{opt} is the smallest valid k hence:

$$\begin{aligned} \frac{1}{2^{k_{opt}-1}}\left\lceil \frac{2^{k_{opt}-1}}{d} \right\rceil &\geq \frac{\lfloor 2^n/d \rfloor}{d\lfloor 2^n/d \rfloor - 1} \\ 2\left\lceil \frac{2^{k_{opt}-1}}{d} \right\rceil &\geq \frac{2^{k_{opt}}\lfloor 2^n/d \rfloor}{d\lfloor 2^n/d \rfloor - 1} \\ \left\lceil \frac{2^{k_{opt}}}{d} \right\rceil + 1 &\geq \frac{2^{k_{opt}}\lfloor 2^n/d \rfloor}{d\lfloor 2^n/d \rfloor - 1} \end{aligned}$$

Hence $a = \lceil 2^{k_{opt}}/d \rceil + 1$ is invalid as it violates (2). We may conclude that the there is only one valid value for a when $k = k_{opt}$. We can now state that the design which minimises k and satisfies $ad - 2^k > 0$ is unique and is defined by:

$$\boxed{\begin{aligned} k^+_{opt} &= \min\left(k : \frac{2^k}{(-2^k) \mod d} > d\left\lfloor \frac{2^n}{d} \right\rfloor - 1\right) \\ a^+_{opt} &= \left\lceil \frac{2^{k^+_{opt}}}{d} \right\rceil \\ b^+_{opt} &= 0 \end{aligned}}$$

B. Optimal RTZ Scheme when $ad - 2^k < 0$

In this case we need:

$$-\frac{b+1}{\lfloor 2^n/d \rfloor} < ad - 2^k < a - b$$

Hence b must necessarily reside in the following interval:

$$b \in \left[(2^k - ad)\lfloor 2^n/d \rfloor, 2^k + a - ad - 1\right]$$

This interval must be non empty so:

$$\begin{aligned} 2^k + a - ad &> (2^k - ad)\lfloor 2^n/d \rfloor \\ \frac{2^k}{d} > a &> \frac{2^k(\lfloor 2^n/d \rfloor - 1)}{d\lfloor 2^n/d \rfloor - d + 1} \end{aligned}$$

Given that a must be an integer we have a formula for k_{opt}:

$$\begin{aligned} k_{opt} = \min&\left(k : \frac{1}{2^k}\left\lfloor \frac{2^k}{d} \right\rfloor > \frac{\lfloor 2^n/d \rfloor - 1}{d\lfloor 2^n/d \rfloor - d + 1}\right) \\ \min&\left(k : \frac{2^k}{2^k \mod d} > d\left\lfloor \frac{2^n}{d} \right\rfloor - d + 1\right) \end{aligned}$$

In a similar manner to the previous section it can be shown that $a = \lfloor 2^{k_{opt}}/d \rfloor$ is valid but $a = \lfloor 2^{k_{opt}}/d \rfloor - 1$ is not valid. We can now state that the design which minimises k and satisfies $ad - 2^k < 0$ is unique in k and a and is defined by:

$$\boxed{\begin{aligned} k^-_{opt} &= \min\left(k : \frac{2^k}{2^k \mod d} > d\left\lfloor \frac{2^n}{d} \right\rfloor - d + 1\right) \\ a^-_{opt} &= \left\lfloor \frac{2^{k^-_{opt}}}{d} \right\rfloor \\ b^-_{opt} &= \text{minh}\left((2^{k^-_{opt}} - a^-_{opt}d)\left\lfloor \frac{2^n}{d} \right\rfloor, 2^{k^-_{opt}} - a^-_{opt}(d-1) - 1\right) \end{aligned}}$$

It can be shown that $k^+_{opt} \neq k^-_{opt}$, hence we can unambiguously choose whichever scheme has smallest k.

VI. Optimal Rounding Schemes

We can apply the techniques found in Section V to the other rounding schemes, the results can be summarised as follows:

$$\begin{aligned} (k, a, b) = \left(k^+ < k^-\right) \quad &? \quad \left(k^+, a^+, \text{minh}\left(Y^+\left(k^+, a^+\right)\right)\right) \\ &: \quad \left(k^-, a^-, \text{minh}\left(Y^-\left(k^-, a^-\right)\right)\right) \end{aligned}$$

Where

$$\begin{aligned} k^+ &= \min\left(k : \frac{2^k}{(-2^k) \mod d} > X^+\right) \\ k^- &= \min\left(k : \frac{2^k}{2^k \mod d} > X^-\right) \\ a^+ &= \left\lceil \frac{2^{k^+}}{d} \right\rceil \qquad a^- = \left\lfloor \frac{2^{k^-}}{d} \right\rfloor \end{aligned}$$

and the definition of $X^\pm$ and $Y^\pm$ can be found in Table 1.

VII. Extension to Division of Two's Complement Numbers

So far it has been assumed that $x \geq 0$. Note that the three rounding modes we have been using are (or can be assumed in the case of FR) to be odd functions. So for x in some bounded negative interval we can find some a and b by proceeding as follows:

$$\begin{aligned} Round\left(\frac{x}{d}\right) &= -Round\left(\frac{-x}{d}\right) \\ &= -\left\lfloor \frac{a(-x)+b}{2^k} \right\rfloor \\ &= (ax - b + 2^k - 1) >> k \end{aligned}$$

TABLE I
$X^{\pm}$ AND $Y^{\pm}$ FOR RTZ, RTE AND FR.

	RTZ	FR
X^+	$d\lfloor 2^n/d \rfloor - 1$	$\lfloor 2^n/d \rfloor$
X^-	$d\lfloor 2^n/d \rfloor - d + 1$	$\lfloor 2^n/d \rfloor$
$Y^+(k,a)$	$(0,0)$	$(0,0)$
$Y^-(k,a)$	$((2^k - ad)\lfloor 2^n/d \rfloor,$ $2^k - a(d-1) - 1)$	$((2^k - ad)\lfloor 2^n/d \rfloor, 2^k - 1)$
	RTE	
X^+	$d\lfloor (2^{n+1} - d - 1)/(2d) \rfloor - 1$	
X^-	$d\lfloor (2^{n+1} - d - 3)/(2d) \rfloor + 1$	
$Y^+(k,a)$	$(a(d-1)/2 + (2^k - ad),$ $a(d-1)/2 + (2^k - ad)\lfloor (2^{n+1} + d - 1)/(2d) \rfloor - 1)$	
$Y^-(k,a)$	$(a(d-1)/2 + (2^k - ad)\lfloor (2^{n+1} + d - 3)/(2d) \rfloor,$ $a(d+1)/2 + 2^k - ad - 1)$	

Now because $Round(0/d) = 0$ then $b < 2^k$ and so $-b + 2^k - 1$ is in fact the bitwise inversion of the k bits of b namely $b[k-1:0]$. We may conclude that if x is an n bit two's complement number then:

$$Round\left(\frac{x}{d}\right) = (ax + b[k-1:0] \oplus x[n-1]) >> k$$

where $\oplus$ here is a bitwise XOR of the k bits of b and k, a, b is constructed by the schemes presented to work for $x \in [0, 2^{n-1}]$.

VIII. Experimental Benchmarks

We can compare the implementation costs of RTZ, RTE, FR with the stated prior work in this area [7]. Synthesis experiments were performed for $n = 16$ and d taking all odd values in the interval [3,49] using Synopsys Design Compiler 2009.06-SP5 in ultra mode using the TSMC 65nm library Tcbn65lpwc. Firstly the synthesis was performed which sought to minimise the logic delay and secondly minimise the area by setting the maximum delay of 2ns and allowing the tool to perform gate sizing. The results can be found in Figure 2. Dividing by a power of 2 utilises no hardware resource; note how the delay and area drops dramatically when d is around a power of 2 in these figures (note only odd values of d have been plotted). The FR design exhibits up to a 20% speed improvement and 50% area improvement over the previous work. The RTZ design, exhibits up to a 10% speed and 16% area improvement over the previous work, but due either to synthesis noise or an ill tuned hardware heuristic, can sometimes perform worse than the previous work.

IX. Conclusion

This paper has explored the full architecture space of implementing constant integer addition using a multiply-add implementation for three rounding modes. The proof framework can be applied to any rounding mode or input interval and an extension to two's complement numbers has been presented. Synthesis results show 20% delay and 50% area improvements over the previous work.

Acknowledgment

The authors would like to thank Imagination Technologies for supporting and reviewing this research.

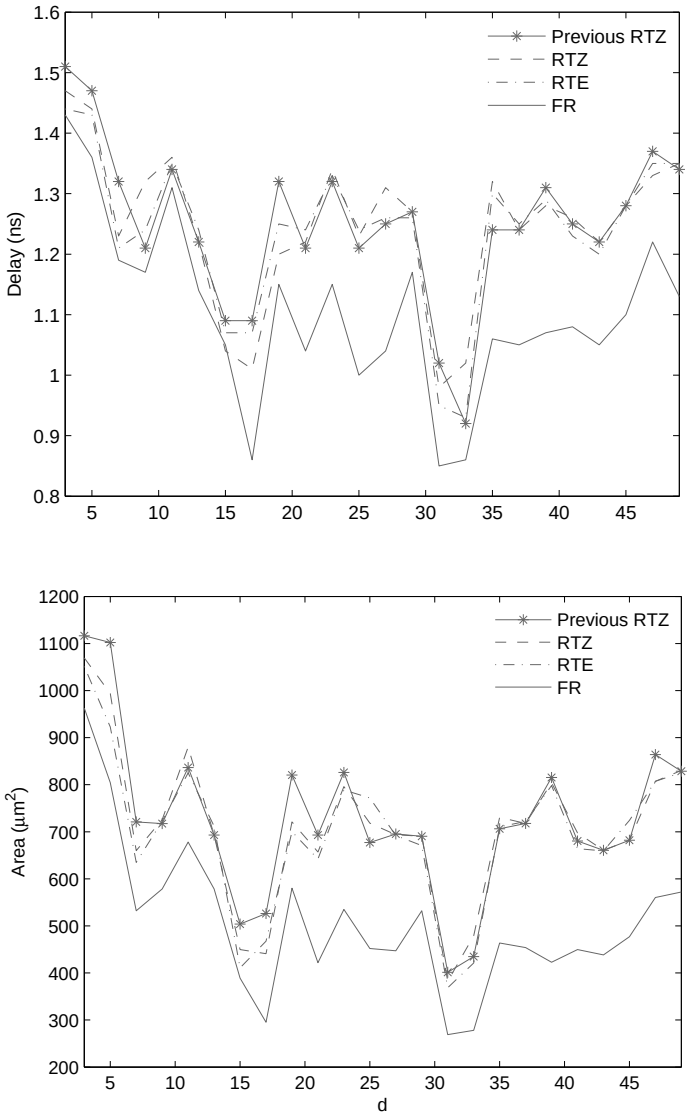

Fig. 2. Delay and Area Synthesis Comparisons.

References

[1] N. Möller and T. Granlund, "Improved division by invariant integers," *IEEE Transactions on Computers*, vol. 60, no. 2, pp. 165 –175, Feb. 2011.

[2] S.-Y. Li, "Fast constant division routines," *IEEE Transactions on Computers*, vol. C-34, no. 9, pp. 866 –869, Sept. 1985.

[3] P. Srinivasan and F. Petry, "Constant-division algorithms," *IEE Proceedings on Computers and Digital Techniques*, vol. 141, no. 6, pp. 334 –340, Nov 1994.

[4] J.-M. Muller, A. Tisserand, B. de Dinechin, and C. Monat, "Division by constant for the ST100 DSP microprocessor," in *17th IEEE Symposium on Computer Arithmetic*, June 2005, pp. 124 – 130.

[5] T. Granlund and P. L. Montgomery, "Division by invariant integers using multiplication," in *In Proceedings of the SIGPLAN '94 Conference on Programming Language Design and Implementation*, 1994, pp. 61–72.

[6] R. Alverson, "Integer division using reciprocals," in *10th IEEE Symposium on Computer Arithmetic*, Jun 1991, pp. 186 –190.

[7] A. Robison, "N-bit unsigned division via n-bit multiply-add," in *17th IEEE Symposium on Computer Arithmetic*, June 2005, pp. 131 – 139.

[8] D. H. Jacobsohn, "A combinatoric division algorithm for fixed-integer divisors," *IEEE Transactions on Computers*, vol. C-22, no. 6, pp. 608 –610, June 1973.