Correctly Rounded Multiplication by Rational Constants via Multiply-Add

***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. Certain applications require multiplication by a constant rational, such situations arise in filtering and number format conversion. This paper shows that greater hardware efficiency can be achieved if multiplication by a constant rational is not considered as constant multiplication followed by constant division. This paper presents the conditions which allow multiply-add schemes to perform correctly rounded unsigned and signed invariant rational multiplication under three common rounding modes. We propose a heuristic to explore the space of implementations meeting the conditions we derive. Logic synthesis experiments show that an area reduction of up to 35% can be achieved compared to existing correctly rounded approaches.**

***Index Terms*—constant multiplication, constant division, correctly rounded, parallel architecture**

I. Introduction and Background

Creating bespoke hardware for multiplication by a constant rational arises in a variety of situations, *e.g.* bespoke filtering, base conversions, certain caching algorithms. The previous work has mainly focused on software implementations, due to the lack of native integer division instructions within existing hardware [1]. Where hardware implementations are considered they are invariably iterative in nature. Certain divisors have been expanded into infinite products which translate into multiple shift and add instructions [2], [3]. The periodicity in the binary expansion of p/q has been exploited to reduce the number of additions required in an iterative implementation, which is most appropriate when the periodic binary portion of p/q is short, [4]. High radix iterative methods for constant division utilising small look-up tables for each iteration has been proposed, [5]. An alternative, put forward in [6], [7], [8], proposed replacing 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 [9] show how only n bit multiply-add operations are required and are thus optimal from a software perspective. The work in [9] 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 [10]. In [11] a multiply-add scheme is considered and optimised for constant division, however work in [11] only considered x/d, rather than the more general px/q.

In this paper we seek parallel hardware for computing invariant rational multiplication using a parallel multiply-add scheme, which is guaranteed to be correctly rounded. Our stated problem is to find integers a, b and k such that:

$$Round\left(\frac{px}{q}\right) = \left\lfloor \frac{ax+b}{2^k} \right\rfloor$$

While $a/2^k$ is expected to be an approximation to p/q, the value of k is difficult to determine while guaranteeing the rounding condition. This equation needs to hold for a signed two's complement or unsigned n bit input x and for any rational p/q. Without loss of generality, it will be assumed that $p \in \mathbb{Z}$, $q \in \mathbb{N}$ and p & q are coprime. It will also be assumed that q is not a power of 2, as that special case can be solved trivially. In this paper we will address three common integer rounding modes, floor (round towards negative infinity), ceiling (round towards positive infinity) and round to nearest-up (RTU).

The contributions of this paper are:

- necessary and sufficient conditions for the multiply-add schemes guaranteed to meet floor, ceiling, and RTU rounding conditions for both signed and unsigned inputs.
- optimal parameter selection with respect to a hardware heuristic, simple enough to be embedded directly into a piece of HDL.
- experimental synthesis comparison of the three rounding schemes and two signages against previous work.

II. Motivating Example

The following is an example of a multiply-add scheme which implements multiplication by $7/9$:

$$\left\lfloor \frac{7x}{9} \right\rfloor = \left\lfloor \frac{199x + 28}{2^8} \right\rfloor \quad x \in [0, 255]$$

There is no freedom in the choice of the constant 28. If either 27 or 29 were used, this identity would fail to hold. This shows that a naive exploration of the design space will fail to find this feasible architecture. Moreover, the techniques found in previous work such as [11] and [9] would first require the creation of $7x$ and then a constant division by 9 scheme. These reduce to the following two schemes:

$$\left\lfloor \frac{7x}{9} \right\rfloor = \left\lfloor \frac{455(7x) + 455}{2^{12}} \right\rfloor = \left\lfloor \frac{1821(7x)}{2^{14}} \right\rfloor \quad x \in [0, 255]$$

So the application of previous approaches will result in multiplying and adding unnecessarily large constants. Implementing this architecture would result in hardware that would waste power, speed and area. By decomposing the need to multiply by a constant rational in terms of a constant integer multiplication and a constant integer division a designer may be designing less efficient hardware. This paper considers the consequences of multiplying by the constant rational p/q as a single operation.

Our aim is to create necessary and sufficient conditions on a, b and k such that an architecture will correctly implement the desired functionality. In order to succinctly establish the conditions for a variety of rounding modes and unsigned and signed inputs, we will consider the problem of determining a, b and k such that:

$$\left\lfloor \frac{px + r}{q} \right\rfloor = \left\lfloor \frac{ax + b}{2^k} \right\rfloor \quad \text{for all} \quad x \in [x_{min}, x_{max}]$$

Judicious choice of r, x_{min} and $x_{max} \in \mathbb{Z}$ will result in a variety of rounding modes and integer input formats. Optimal a, b and k will depend on the given values of p, q, r, x_{min} and x_{max}. Finding an analytic solution to this problem has required an additional constraint, namely $x_{max} - x_{min} \geq 3(q-1)$.

III. Hardware Heuristic

The hardware heuristic we adopt here is that adopted by [11] which first seeks to minimise the width of the constant multiplication array and then attempts to minimise the height. The width is minimised by minimising k. It turns out, this results in no freedom in the value of a. However there will be an interval in which b may reside. Within the set of valid b, the

values which are expected to contribute the least to the partial product array height will be those with minimal Hamming weight and minimal value. The following function computes this value for an interval $[A, B]$ where A and B are p bits in length:

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

So optimal or minimal values of a, b and k will be determined with respect to this hardware implementation cost heuristic.

IV. Determining Minimal Values of a, b and k

A. Solution Existence

1) Set of Feasible k: Given that $\left\lfloor \frac{ax+b}{2^k} \right\rfloor = \left\lfloor \frac{(2a)x+(2b)}{2^{k+1}} \right\rfloor$, then if k is feasible then so is $k+1$. The set of feasible values of k is thus of the form: $[k_{\min}, \infty)$, where $k_{\min}$ is this minimal value of k.

We will also exploit the fact that $k_{\min} > 0$. This can be shown by demonstrating that $k = 0$ is infeasible. Given that it is being assumed that $x_{max} - x_{min} \geq 3(q-1)$ and that $q \in \mathbb{N}$ which is not a power of 2 then $x_{max} - x_{min} \geq 3(q-1) > q$. This implies that x can take the values x_{min} and $x_{min} + q$. If $k = 0$ this would mean:

$$\left\lfloor \frac{p(x_{min}+q)+r}{q} \right\rfloor - \left\lfloor \frac{px_{min}+r}{q} \right\rfloor = (a(x_{min}+q)+b) - (ax_{min}+b)$$

$$p = aq$$

But p and q are coprime, so this is a contradiction, hence $k_{\min} > 0$. Future calculations involve 2^k, which can now be assumed be an integer.

2) Existence of feasible k: By increasing k and with an appropriate choice of a, $a/2^k$ can get arbitrarily close to p/q and $b/2^k$ can get arbitrarily close to r/q. So there exists a choice of a, b and k such that the following holds for all $x \in [x_{min}, x_{max}]$:

$$\frac{r}{q} \leq \left(\frac{a}{2^k} - \frac{p}{q}\right)x + \frac{b}{2^k} < \frac{r+1}{q}$$

$$\text{Rearranging:} \quad 0 \leq \frac{ax+b}{2^k} - \frac{px+r}{q} < \frac{1}{q}$$

Finally, we note that since p, x and r are all integers, $\frac{px+r}{q}$ is a multiple of $\frac{1}{q}$, so the fact that $\frac{ax+b}{2^k} - \frac{px+r}{q}$ is less than $\frac{1}{q}$, implies their integer parts are identical:

$$\left\lfloor \frac{px+r}{q} \right\rfloor = \left\lfloor \frac{ax+b}{2^k} \right\rfloor \quad \forall x \in [x_{min}, x_{max}]$$

B. Determining a given $k = k_{\min}$

1) a is odd when $k = k_{\min}$: This is proved by contradiction, assume a were even when $k = k_{\min}$, then:

$$\left\lfloor \frac{ax+b}{2^{k_{\min}}} \right\rfloor = \left\lfloor \frac{\frac{a}{2}x + \left\lfloor \frac{b}{2} \right\rfloor}{2^{k_{\min}-1}} \right\rfloor$$

This would imply that $k_{\min}$ was not minimal, which is a contradiction. Conclude that a is odd when $k = k_{\min}$.

2) The Solution Space is Convex: Consider the relaxation of the integral constraints on a and b by taking the values of

$\alpha, \beta \in \mathbb{R}$ which satisfy the following:

$$\left\lfloor \frac{px+r}{q} \right\rfloor = \lfloor \alpha x + \beta \rfloor \quad x \in [x_{min}, x_{max}]$$

The $2D$ space formed in $\mathbb{R}^2$ by these (α, β) values will now be shown to be convex. Suppose the solution space is non-empty such that there exists α_1, α_2, β_1 and $\beta_2 \in \mathbb{R}$ for which:

$$\left\lfloor \frac{px+r}{q} \right\rfloor = \lfloor \alpha_1 x + \beta_1 \rfloor = \lfloor \alpha_2 x + \beta_2 \rfloor \quad \forall x \in [x_{min}, x_{max}]$$

Firstly, defining the fractional parts X_1 and $X_2 \in [0, 1)$:

$$\alpha_1 x + \beta_1 = \left\lfloor \frac{px+r}{q} \right\rfloor + X_1 \quad \alpha_2 x + \beta_2 = \left\lfloor \frac{px+r}{q} \right\rfloor + X_2$$

Now let, for some $\lambda \in [0, 1]$:

$$\alpha = (1-\lambda)\alpha_1 + \lambda\alpha_2 \qquad \beta = (1-\lambda)\beta_1 + \lambda\beta_2$$

Now consider:

$$\begin{aligned} &\lfloor \alpha x + \beta \rfloor = \lfloor (1-\lambda)(\alpha_1 x + \beta_1) + \lambda(\alpha_2 x + \beta_2) \rfloor \\ &= \left\lfloor \frac{px+r}{q} \right\rfloor + \lfloor (1-\lambda)X_1 + \lambda X_2 \rfloor \\ &= \left\lfloor \frac{px+r}{q} \right\rfloor \end{aligned}$$

Conclude that the $2D$ space of feasible (α, β) is convex. Hence if there exist integers a and k such that: $\alpha_1 < \frac{a}{2^k} < \alpha_2$ for feasible α_1 and α_2, then by the convexity of the solution space there exists β such that:

$$\left\lfloor \frac{px+r}{q} \right\rfloor = \left\lfloor \frac{a}{2^k} x + \beta \right\rfloor = \left\lfloor \frac{ax + \lfloor 2^k \beta \rfloor}{2^k} \right\rfloor$$

Hence a and k are feasible solutions to the original problem.

3) $k = k_{\min}$ implies that a is unique: Proceeding by contradiction, it is known that a is odd when k is minimal. If a is not unique, then there exists two distinct feasible odd values of a, $a_1 < a_2$. However, this implies that there exists a feasible even value for a such that $a_1 < a < a_2$ for a minimal k. This is a contradiction with the necessary oddness of a, so we can conclude that when k is minimal, a is unique and odd; this value will be called $a_{\min}$.

4) Determining $a_{\min}$: Given $k_{\min}$, we will bound the possible values of $a_{\min}$. Suppose:

$$\begin{aligned} a_{\min} &> \left\lceil \frac{p2^{k_{\min}}}{q} \right\rceil \\ \frac{p}{q} &< \frac{1}{2^{k_{\min}}} \left\lceil \frac{p2^{k_{\min}}}{q} \right\rceil < \frac{a_{\min}}{2^{k_{\min}}} \end{aligned}$$

But by the section on convexity, $a = \left\lceil \frac{p2^{k_{\min}}}{q} \right\rceil$ must also be a feasible solution for a. This contradicts the uniqueness of $a_{\min}$. Similarly if we assume that:

$$\begin{aligned} a_{\min} &< \left\lfloor \frac{p2^{k_{\min}}}{q} \right\rfloor \\ \frac{a_{\min}}{2^{k_{\min}}} &< \frac{1}{2^{k_{\min}}} \left\lfloor \frac{p2^{k_{\min}}}{q} \right\rfloor < \frac{p}{q} \end{aligned}$$

But by the section on convexity, $a = \left\lfloor \frac{p2^{k_{\min}}}{q} \right\rfloor$ must also be a feasible solution for a. This contradicts the uniqueness of $a_{\min}$. Conclude that:

$$\left\lfloor \frac{p2^{k_{\min}}}{q} \right\rfloor \leq a_{\min} \leq \left\lceil \frac{p2^{k_{\min}}}{q} \right\rceil$$

These two integer bounds are consecutive integers, hence given the minimal value of k, then:

$$a_{\min} = \text{odd one of} \quad \left\lfloor \frac{p2^{k_{\min}}}{q} \right\rfloor \quad \text{and} \quad \left\lceil \frac{p2^{k_{\min}}}{q} \right\rceil$$

C. Determining $k_{\min}$

It will be first assumed that $a_{\min} = \left\lfloor \frac{2^{k_{\min}} p}{q} \right\rfloor$ and the associated minimum value for k will be derived, $k_{\min}^{-}$. The calculations for $a_{\min} = \left\lceil \frac{2^{k_{\min}} p}{q} \right\rceil$ proceed similarly. The smallest of these $k_{\min}$ values produces the optimal scheme.

1) Assuming that $a_{\min} = \left\lfloor \frac{2^{k_{\min}} p}{q} \right\rfloor$: We require

$$\left\lfloor \frac{px+r}{q} \right\rfloor = \left\lfloor \frac{ax+b}{2^k} \right\rfloor \quad \forall \text{ integer } x \in [x_{min}, x_{max}]$$

which implies:

$$0 \leq \frac{ax+b}{2^k} - \left\lfloor \frac{px+r}{q} \right\rfloor < 1$$
$$0 \leq (aq - p2^k)x + 2^k(px + r \mod q) + bq - r2^k < q2^k$$

Substituting and simplifying $a = \left\lfloor \frac{2^k p}{q} \right\rfloor$:

$$0 \leq 2^k(px + r \mod q) - x(p2^k \mod q) + bq - r2^k < q2^k \tag{1}$$

Consider the part involving x, $f(x) = 2^k(px + r \mod q) - x(p2^k \mod q)$. Now f has the following property:

$$\forall x \quad f(x+q) = f(x) - q(p2^k \mod q) < f(x)$$

Hence the values that produce the maximum and minimum of f must lie in the intervals $[x_{min}, x_{min} + q - 1]$ and $[x_{max} - q + 1, x_{max}]$ respectively. To proceed with finding the extrema of f we will assume that the drift term, $-x(p2^k \mod q)$, is ignorable with respect to the modulo term, which can be assumed if we ensure $\frac{2^k}{(p2^k \mod q)} \geq q - 1$. With this assumption, the actual values for extrema of f are the values for the extrema of the modulo term of f in the respective intervals, these are:

$$\omega^-_{\max} = x_{min} + (-x_{min} - p_q^{-1}(r+1) \mod q)$$
$$\omega^-_{\min} = x_{max} - (x_{max} + p_q^{-1} r \mod q)$$

Where p_q^{-1} is the multiplicative inverse of p modulo q (note that this exists as p and q are coprime). Substituting $\omega^-_{\max / \min}$ back into equation (1) requires the following two inequalities to hold:

$$q2^k > 2^k(q-1) - \omega^-_{\max}(p2^k \mod q) + bq - r2^k$$
$$0 \leq -\omega^-_{\min}(p2^k \mod q) + bq - r2^k$$

Rearranging:

$$\frac{r2^k + \omega^-_{\min}(p2^k \mod q)}{q} \leq b < \frac{(r+1)2^k + \omega^-_{\max}(p2^k \mod q)}{q}$$

Notice that the values of $\omega^-_{\min}$ and $\omega^-_{\max}$ imply that the upper and lower bounds of these inequalities are integers. For this interval to be non empty, it is required that:

$$r2^k + \omega^-_{\min}(p2^k \mod q) < (r+1)2^k + \omega^-_{\max}(p2^k \mod q)$$
$$\frac{2^k}{p2^k \mod q} > \omega^-_{\min} - \omega^-_{\max}$$

Therefore, $k^-_{\min}$ is the minimum such value which satisfies this:

$$k^-_{\min} = \min\left(k : \frac{2^k}{(p2^k \mod q)} > \omega^-_{\min} - \omega^-_{\max}\right)$$

With the associated value of b satisfying:

$$b^-_{\min} \in \left[\frac{r2^k + \omega^-_{\min}(p2^k \mod q)}{q}, \frac{(r+1)2^k + \omega^-_{\max}(p2^k \mod q)}{q} - 1\right]$$

Now in determining the extrema of f, we assumed that $\frac{2^k}{(p2^k \mod q)} \geq q - 1$. This assumption will hold if we force:

$$\omega^-_{\min} - \omega^-_{\max} \geq q - 1$$
$$x_{max} - x_{min} - (x_{max} + p_q^{-1} r \mod q) - (-x_{min} - p_q^{-1}(r+1) \mod q) \geq q - 1$$

Bounding the modulo terms produces a sufficient but non necessary condition for this inequality to hold: $x_{max} - x_{min} \geq 3(q-1)$.

D. Assuming that $a_{\min} = \left\lceil \frac{2^{k_{\min}} p}{q} \right\rceil$

Proceeding as in the previous case, the values for ω^+ are:

$$\omega^+_{\max} = x_{max} - (x_{max} + p_q^{-1}(r+1) \mod q)$$

$$\omega^+_{\min} = x_{min} + (-x_{min} - p_q^{-1} r \mod q)$$

The resultant

$$k^+_{\min} = \min\left(k : \frac{2^k}{(-p2^k \mod q)} > \omega^+_{\max} - \omega^+_{\min}\right)$$
$$b^+_{\min} \in \left[\frac{r2^k - \omega^+_{\min}(-p2^k \mod q)}{q}, \frac{2^k(r+1) - \omega^+_{\max}(-p2^k \mod q)}{q} - 1\right]$$

E. Choosing minimal b

Once $k_{\min}$ has been determined for both cases, there is still a choice in the value of b, b will reside in an interval $[b_{\min}, b_{\max}]$. The heuristic used will be that presented in Section III.

V. Optimal Implementation Schemes

We now turn to how the result can be used to determine optimal implementations for floor, ceiling, and RTU rounding conditions for both signed and unsigned inputs by judicious choice of x_{min}, x_{max} and r.

Unsigned $x \in [0, 2^n - 1]$ set $x_{min} = 0, x_{max} = 2^n - 1$

Signed $x \in [-2^{n-1}, 2^{n-1} - 1]$ set $x_{min} = -2^{n-1}, x_{max} = 2^{n-1} - 1$

$$Floor\left(\frac{px}{q}\right) = \left\lfloor \frac{px}{q} \right\rfloor \quad \text{set} \quad r = 0$$

$$Ceiling\left(\frac{px}{q}\right) = \left\lceil \frac{px}{q} \right\rceil = \left\lfloor \frac{px + q - 1}{q} \right\rfloor \quad \text{set} \quad r = q - 1$$

$$RTU\left(\frac{px}{q}\right) = \left\lfloor \frac{px}{q} + \frac{1}{2} \right\rfloor = \left\lfloor \frac{px + \lfloor q/2 \rfloor}{q} \right\rfloor \quad \text{set} \quad r = \lfloor q/2 \rfloor$$

We can now present the optimal solution to determining $a, b, k \in \mathbb{Z}$ such that:

$$\text{Round}\left(\frac{px}{q}\right) = \left\lfloor \frac{ax + b}{2^k} \right\rfloor$$

For an unsigned/signed $n \in \mathbb{N}$ bit input x for rounding modes floor, ceiling and RTU assuming that $p \in \mathbb{Z}$, $q \in \mathbb{N}$, p and q coprime, q is not a power of 2 and $x_{max} - x_{min} \geq 3(q-1)$, which reduces to $n \geq \log_2(3q-2)$ in this case (if n violates this constraint an implementation using $n = \lceil \log_2(3q-2) \rceil$ can be created, although the resulting design may not have an optimal value of k).

Inputs $p \in \mathbb{Z} \quad q, n \in \mathbb{N}$

Signage (Unsigned/Signed)

Rounding (Floor/Ceiling/RTU)

Outputs (a, b, k)

Find $p_q^{-1} \in \mathbb{Z}$ such that $(p_q^{-1} p \mod q) = 1$

$x_{min} = \text{Unsigned?} \quad 0 \quad : \quad \text{Signed?} \quad -2^{n-1}$

$x_{max} = \text{Unsigned?} \quad 2^n - 1 \quad : \quad \text{Signed?} \quad 2^{n-1} - 1$

$r = \text{Floor?} \quad 0 \quad : \quad \text{Ceiling?} \quad q - 1 \quad : \quad \text{RTU?} \quad \lfloor q/2 \rfloor$

$\omega^-_{\max} = x_{min} + (-x_{min} - p_q^{-1}(r+1) \mod q)$

$\omega^-_{\min} = x_{max} - (x_{max} + p_q^{-1} r \mod q)$

$\omega^+_{\max} = x_{max} - (x_{max} + p_q^{-1}(r+1) \mod q)$

$\omega^+_{\min} = x_{min} + (-x_{min} - p_q^{-1} r \mod q)$

$k^\pm = \min\left(k : \frac{2^k}{(\mp p2^k \mod q)} > \pm\left(\omega^\pm_{\max} - \omega^\pm_{\min}\right)\right)$

$a^+ = \left\lceil \frac{p2^{k^+}}{q} \right\rceil \quad a^- = \left\lfloor \frac{p2^{k^-}}{q} \right\rfloor$

$b^\pm_{\min} = \frac{r2^{k^\pm} \mp \omega^\pm_{\min}(\mp p2^{k^\pm} \mod q)}{q}$

$b^\pm_{\max} = \frac{(r+1)2^{k^\pm} \mp \omega^\pm_{\max}(\mp p2^{k^\pm} \mod q)}{q} - 1$

$b^\pm = \text{minh}(b^\pm_{\min}, b^\pm_{\max})$

return $(k^- < k^+)? \quad (a^-, b^-, k^-) \quad : \quad (a^+, b^+, k^+)$

VI. Experimental Results

The equations presented in the previous section were embedded in a single piece of parameterisable HDL. Parameterisable code was also created for the previous relevant methods found in the literature, [9] and [11]. Logic synthesis experiments were performed using Synopsys' Design Compiler 2013.12-SP2 in ultra mode using the TSMC 28nm library tcbn28hpmbwp35ss0p81v0c for a variety of bit widths, input

signages and rounding modes. The logic synthesis implements the multiply-add scheme by creating and reducing a single partial product array for $ax + b$. The multiplication is realised by creating a canonical signed digit representation of the constant a. A variety of frequencies was targeted; once the frequency is set, the tool performed gate sizing to reduce the area. A selection of area/delay points were extracted which enabled the construction of Pareto curves. The designs used, where appropriate, for comparison included the naive use of the native HDL operations $\times$,$+$ and $/$, [9] and [11] where px/q is encoded as a constant multiplication followed by a constant division. In order to demonstrate the effectiveness of the proposed method over all previous approaches, synthesis experiments were performed for $n = 32$, $p = 341$ and $q = 845$ for all rounding modes and unsigned inputs. The resultant area vs delay Pareto curves for all three experiments can be found in Figures 1,2 and 3. Note that the curves for [9] and [11] are coincident in Figure 1.

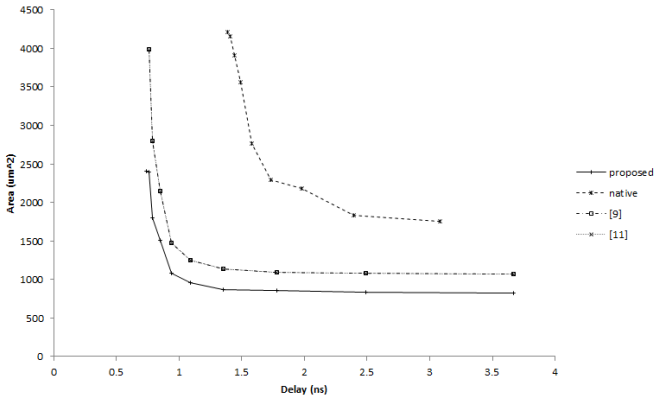

Fig. 1. Area vs Delay for Floor rounding with an Unsigned 32 bit input

For unsigned 32 bit inputs, this new approach is typically $20 - 35\%$ smaller and just as fast as the previous approaches. Compared to native RTL operators, this new approach is between $55 - 85\%$ smaller for common delays and the fastest produced designs offer a 47% reduction in critical delay.

In order to get an understanding of how the proposed

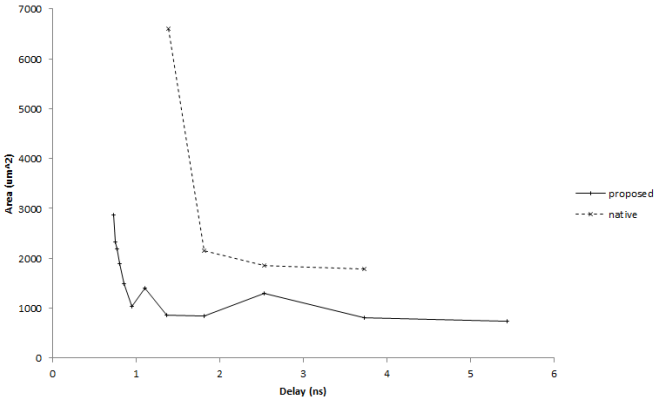

Fig. 2. Area vs Delay for Ceiling rounding with an Unsigned 32 bit input

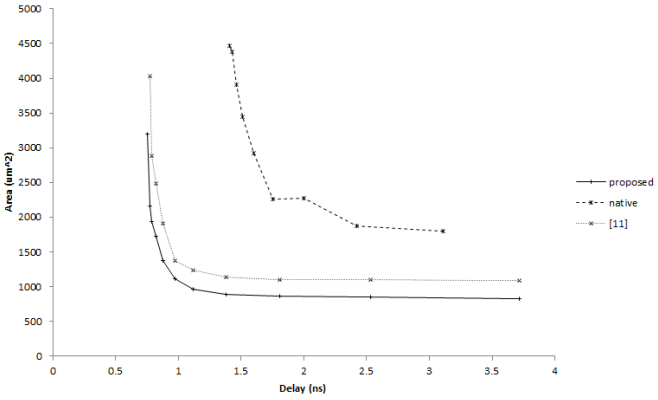

Fig. 3. Area vs Delay for RTU rounding with an Unsigned 32 bit input

method compares in general, a range of 16 p and q values were selected and logic synthesis was performed for $n = 32$, unsigned input, floor rounding mode and a delay of 0.9ns. Figure 4 shows a bar chart comparing the area of the proposed method against the best of the previous methods, [11]. On average, the designs produced by the new approach were 8% smaller than those produced by the best of the previous approaches with a peak at 25% area reduction.

VII. Conclusion and Future Work

This paper has explored the full architecture space of implementing multiplication by a constant rational using a multiply-add implementation for three rounding modes, supporting unsigned or signed inputs. The proof framework can be applied to a variety of rounding modes and used on a general input

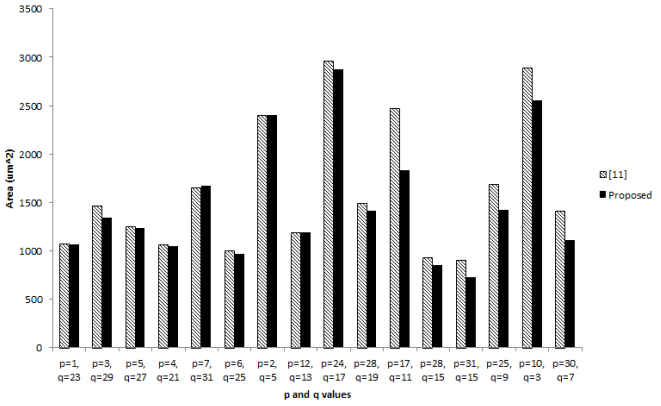

Fig. 4. Area comparisons for proposed vs the best of previous approaches.

interval. Synthesis results demonstrate an area reduction of up to 35% over previous work. Proof has been provided that there can be no better multiply-add scheme with respect to the heuristic used. Future work includes removing the constraint, $x_{max} - x_{min} \geq 3(q-1)$.

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] F. de Dinechin, "Multiplication by rational constants," *Circuits and Systems II: Express Briefs, IEEE Transactions on*, vol. 59, no. 2, pp. 98–102, Feb 2012.

[5] F. de Dinechin and L.-S. Didier, "Table-based division by small integer constants," in *Reconfigurable Computing: Architectures, Tools and Applications*, ser. Lecture Notes in Computer Science, O. Choy, R. Cheung, P. Athanas, and K. Sano, Eds. Springer Berlin Heidelberg, 2012, vol. 7199, pp. 53–63.

[6] 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.

[7] 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.

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

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

[10] 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.

[11] T. Drane, W. Cheung, and G. Constantinides, "Correctly rounded constant integer division via multiply-add," in *Circuits and Systems (ISCAS), 2012 IEEE International Symposium on*, May 2012, pp. 1243–1246.