Design and Implementation of Complex Multiply Add and Other Similar Operators

Pouya Dormiani and Miloš D. Ercegovac

Computer Science Department, UCLA, Los Angeles, CA 90025, USA

ABSTRACT

In this paper we present an implementation of a series of complex-valued operators defined in Ref. 2: Complex-Multiply-Add (CMA), Complex-Sum of Squares (CSS), and Complex-Sum of Products (CSP). The preceding paper[2] defined these operators at an algorithmic level, for which we now provide actual hardware performance metrics through detailed discussion of their implementation for an Altera Stratix II[17] FPGA device. In addition to discussing these designs in particular, we present our methodology and choice of tools to create a pragmatic generator.

Keywords: Complex arithmetic, multiply-add, sum of products, sum of squares, FPGA implementation, design generator.

1. INTRODUCTION

With the exception of complex addition and multiplication,[1,12,16] complex online SVD,[11] complex operations are typically not implemented in hardware. Recently, hardware-oriented methods for complex division and square root have been introduced.[6–8] The complex operators designed in this paper all derive from a single concept, namely the complex-valued evaluation-method (CE-method),[9] which uses a commonly known isomorphism in conjunction with the real-valued E-method[3].[4] We only present portions of the theory which are relevant to understanding the implementation. The main characteristic of the E-method is its simplicity where easily computable recurrences are devised to feed each other in lock-step to solve systems of linear equations. These recurrences, a generalization of the SRT division[13] have the following matrix-vector form in the radix 2:

$$\mathbf{w}^{(j)} = 2 \times [\mathbf{w}^{(j-1)} - \mathbf{A}\mathbf{d}^{(j-1)}] \tag{1}$$

with $\mathbf{w}^{(j)}$ being the j^{th} iteration residual vector (n-tuple), $\mathbf{A}$ being an $n \times n$ matrix of coefficients whose form dictates the expression to be evaluated, and $\mathbf{d}$ is a digit-vector with $d_i^{(j)} \in \{-1, 0, 1\}$. The matrix $\mathbf{A}$ must be strictly diagonally dominant to ensure non singularity and recurrence convergence. Partially expanded, the recurrence is of the form

$$w_i^{(j)} = 2 \times \left[w_i^{(j-1)} - d_i^{(j-1)} + \sum_{\substack{k=0 \\ k \neq i}}^{n-1} a_{ik} d_k^{(j-1)} \right], \; i = 0 \ldots n-1 \tag{2}$$

Adopting the original notation in[3] allows for better understanding of selection and the convergence conditions relating to the E-method. For this reason, let us redefine the recurrence in (2) as

$$z_i^{(j-1)} = w_i^{(j-1)} - d_i^{(j-1)} \tag{3a}$$

$$w_i^{(j)} = 2 \times \left[z_i^{(j-1)} + \sum_{\substack{k=0 \\ k \neq i}}^{n-1} a_{ik} d_k^{(j-1)} \right], \; i = 0, \ldots, n-1 \tag{3b}$$

upon which we base our discussions of selection and convergence. In the discussions that follow we refer to $w_i^{(j)}$ as the *residual* and $z_i^{(j)}$ as the *adjusted residual*.

Advanced Signal Processing Algorithms, Architectures, and Implementations XVIII,
edited by Franklin T. Luk, Proc. of SPIE Vol. 7074, 70740J, (2008) · 0277-786X/08/$18 · doi: 10.1117/12.793478

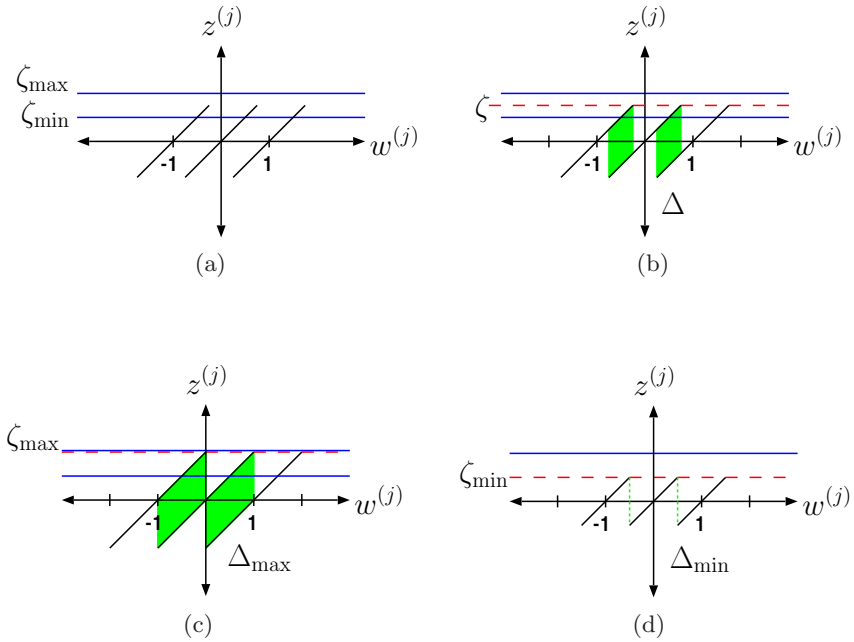


Figure 1: Robertson's diagrams displaying the role of bounds placed on the residual and adjusted residual as well as the overlap of selection intervals. (a) A plot of relationships between $w^{(j)}$ and $z^{(j)}$ for all possible values of $d^{(j)}$ (b) Demonstrates overlaps between selection regions (shown in green) for which both choices of $d^{(j)}$ keeps the adjusted residual and residual contained in the next iteration. (c) Maximum allowable overlap. (d) Minimum allowable overlap.

1.1 Digit Selection Function

We now discuss in detail derivation of the selection function used in the design. The approach is similar to the derivation of selection functions for digit-recurrence algorithms.[5] Once a recurrence, e.g., Eqn. (3a), has been established, all ensuing analysis is dependent upon the selection function as it generates $d^{(j)}$ in every iteration. Plotting eqn. (3a) for different values of $d^{(j)}$ results in the Robertson's diagram shown in Fig. 1(a).

Define a constant ζ to bound the adjusted residual

$$|z^{(j)}| \leq \zeta, \quad \zeta_{\min} \leq \zeta < \zeta_{\max} \tag{4}$$

It can be seen from Fig. 1(a) that if $\zeta < 1/2$, then the continuity condition of the selection function is unsatisfied as it will leave values of $w^{(j)}$ for which no choice of $d^{(j)}$ yields $|z^{(j)}| < 1/2$. Likewise if $\zeta \geq 1$, then $w^{(j)}$ could exceed the selection range when $rz^{(j)}$ is computed in the recurrence step, and therefore not permit selection of any digit. Therefore, $\zeta_{\min} = 1/2$ and $\zeta_{\max} = 1$, from which we obtain the bound,

$$|w^{(j)}| \leq 1 + \zeta \tag{5}$$

Now let

$$\left| \sum_{\substack{k=0 \\ k \neq i}}^{n-1} a_{ik}\, d_k^{(j-1)} \right| \leq \alpha, \; i = 0, \ldots, n-1 \tag{6}$$

for which we now use the recurrence in eqn. (3b) to determine the bound on α.

$$|w^{(j)}| = 2 \left| z^{(j-1)} + \sum_{\substack{k=0 \\ k \neq i}}^{n-1} a_{ik}\, d_k^{(j)} \right| \leq 2(\zeta + \alpha) \Rightarrow \alpha \leq \frac{1}{2}(1 - \zeta) \tag{7}$$

Again, from the selection diagrams in Fig. 1 we see the relationship between ζ and the quantity of overlap between selection regions denoted Δ, given by,

$$\Delta = 2\zeta - 1,\ 0 \leq \Delta < 1 \tag{8}$$

Now that all the parameters in the design have been established, we can see there are multiple values ζ, Δ and α for which the recurrence in eqn.(3) will converge. One way of describing this selection mechanism is via eqn. (9).

$$S(x) = \begin{cases} \text{sign } x \times \lfloor |x| + \frac{1}{2} \rfloor & , |x| \leq 1 \\ \text{sign } x \times \lfloor |x| \rfloor & , \text{otherwise} \end{cases} \tag{9}$$

2. DESIGN

In order to design CMA, CSS, and CSP operators, we first develop several elementary units used by all operators. To identify these elementary units, the following summary of the complex operators[2] is used. Some similar designs for online multiply-add were considered in Ref. 14.

2.1 CMA

This operator computes $y = ab + c$. In the complex domain, the variables are $y = y^r + iy^i$, $a = a^r + ia^i$, $b = b^r + ib^i$, and $c = c^r + ic^i$. The mapping in this case is,

$$\begin{pmatrix} 1 & 0 & -a^r & a^i \\ 0 & 1 & -a^i & -a^r \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} s_0^r \\ s_0^i \\ s_1^r \\ s_1^i \end{pmatrix} = \begin{pmatrix} c^r \\ c^i \\ b^r \\ b^i \end{pmatrix}$$

so that $s_0^r = y^r$ and $s_0^i = y^i$. With residual recurrences,

$$\begin{aligned} w_{0,r}^{(0)} &= c^r, \quad w_{0,r}^{(j)} = 2\left[w_{0,r}^{(j-1)} - d_{0,r}^{(j-1)} + a^r d_{1,r}^{(j-1)} - a^i d_{1,i}^{(j-1)}\right] \\ w_{0,i}^{(0)} &= c^i, \quad w_{0,i}^{(j)} = 2\left[w_{0,i}^{(j-1)} - d_{0,i}^{(j-1)} + a^i d_{1,r}^{(j-1)} + a^r d_{1,i}^{(j-1)}\right] \\ w_{1,r}^{(0)} &= b^r, \quad w_{1,r}^{(j)} = 2\left[w_{1,r}^{(j-1)} - d_{1,r}^{(j-1)}\right] \\ w_{1,i}^{(j)} &= b^i, \quad w_{1,i}^{(j)} = 2\left[w_{1,i}^{(j-1)} - d_{1,i}^{(j-1)}\right] \end{aligned} \tag{10}$$

2.2 CSP

This operator computes $y = ab + ce + f$. In the complex domain, $a = a^r + ia^i$, $b = b^r + ib^i$, etc. The mapping in this case is,

$$\begin{pmatrix} 1 & 0 & -a^r & a^i & -c^r & c^i \\ 0 & 1 & -a^i & -a^r & -c^i & -c^r \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} s_0^r \\ s_0^i \\ s_1^r \\ s_1^i \\ s_2^r \\ s_2^i \end{pmatrix} = \begin{pmatrix} f^r \\ f^i \\ b^r \\ b^i \\ e^r \\ e^i \end{pmatrix}$$

with residual recurrences,

$$\begin{aligned} w_{0,r}^{(0)} &= f^r, \quad w_{0,r}^{(j)} = 2\left[w_{0,r}^{(j-1)} - d_{0,r}^{(j-1)} + a^r d_{1,r}^{(j-1)} - a^i d_{1,i}^{(j-1)} + c^r d_{2,r}^{(j-1)} - c^i d_{2,i}^{(j-1)}\right] \\ w_{0,i}^{(0)} &= f^i, \quad w_{0,i}^{(j)} = 2\left[w_{0,i}^{(j-1)} - d_{0,i}^{(j-1)} + a^i d_{1,r}^{(j-1)} + a^r d_{1,i}^{(j-1)} + c^i d_{2,r}^{(j-1)} + c^r d_{2,i}^{(j-1)}\right] \\ w_{1,r}^{(0)} &= b^r, \quad w_{1,r}^{(j)} = 2\left[w_{1,r}^{(j-1)} - d_{1,r}^{(j-1)}\right] \\ w_{1,i}^{(j)} &= b^i, \quad w_{1,i}^{(j)} = 2\left[w_{1,i}^{(j-1)} - d_{1,i}^{(j-1)}\right] \\ w_{2,r}^{(j)} &= e^r, \quad w_{2,r}^{(j)} = 2\left[w_{2,r}^{(j-1)} - d_{2,r}^{(j-1)}\right] \\ w_{2,i}^{(j)} &= e^i, \quad w_{2,i}^{(j)} = 2\left[w_{2,i}^{(j-1)} - d_{2,i}^{(j-1)}\right] \end{aligned} \tag{11}$$

2.3 CSS

This operator computes $y = a^2 + b^2$. In the complex domain, $a = a^r + ia^i$, and $b = b^r + ib^i$. The mapping in this case is

$$\begin{pmatrix} 1 & 0 & -a^r & a^i & -b^r & b^i \\ 0 & 1 & -a^i & -a^r & -b^i & -b^r \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} s_0^r \\ s_0^i \\ s_1^r \\ s_1^i \\ s_2^r \\ s_2^i \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ a^r \\ a^i \\ b^r \\ b^i \end{pmatrix}$$

with residual recurrences,

$$\begin{aligned}
w_{0,r}^{(0)} = 0, \quad & w_{0,r}^{(j)} = 2\left[w_{0,r}^{(j-1)} - d_{0,r}^{(j-1)} + a^r d_{1,r}^{(j-1)} - a^i d_{1,i}^{(j-1)} + b^r d_{2,r}^{(j-1)} - b^i d_{2,i}^{(j-1)}\right] \\
w_{0,i}^{(0)} = 0, \quad & w_{0,i}^{(j)} = 2\left[w_{0,i}^{(j-1)} - d_{0,i}^{(j-1)} + a^i d_{1,r}^{(j-1)} + a^r d_{1,i}^{(j-1)} + b^i d_{2,r}^{(j-1)} + b^r d_{2,i}^{(j-1)}\right] \\
w_{1,r}^{(0)} = a^r, \quad & w_{1,r}^{(j)} = 2\left[w_{1,r}^{(j-1)} - d_{1,r}^{(j-1)}\right] \\
w_{1,i}^{(j)} = a^i, \quad & w_{1,i}^{(j)} = 2\left[w_{1,i}^{(j-1)} - d_{1,i}^{(j-1)}\right] \\
w_{2,r}^{(j)} = b^r, \quad & w_{2,r}^{(j)} = 2\left[w_{2,r}^{(j-1)} - d_{2,r}^{(j-1)}\right] \\
w_{2,i}^{(j)} = b^i, \quad & w_{2,i}^{(j)} = 2\left[w_{2,i}^{(j-1)} - d_{2,i}^{(j-1)}\right]
\end{aligned} \tag{12}$$

2.4 Elementary Units

The recurrences used in the operators identify three types of generic recurrences to be implemented, namely,

$$\textbf{ZEU: } w^{(j)} = 2\left[w^{(j-1)} - d^{(j-1)}\right] \tag{13}$$

$$\textbf{PEU: } w^{(j)} = 2\left[w^{(j-1)} - d^{(j-1)} + \sigma_1 d_{\sigma_1}^{(j-1)} + \sigma_2 d_{\sigma_2}^{(j-1)}\right] \tag{14}$$

$$\textbf{REU: } w^{(j)} = 2\left[w^{(j-1)} - d^{(j-1)} + \sigma_1 d_{\sigma_1}^{(j-1)} + \sigma_2 d_{\sigma_2}^{(j-1)} + \sigma_3 d_{\sigma_3}^{(j-1)} + \sigma_4 d_{\sigma_4}^{(j-1)}\right] \tag{15}$$

where σ_i is a constant and $d_{\sigma_i}^{(j)}$ is the digit which multiplies that constant. Note that to obtain negative weighted values, the digits can be negated easily, so the elementary unit will simply implement the sum of all $\sigma_i d_{\sigma_i}^{(j)}$.

When implementing these recurrences, several design factors are crucial for performance considerations, one of which is to compute the addition of terms in the recurrence using a redundant scheme to avoid carry propagation and reduce the cycle time. This requires that the residual be stored in a redundant form, and also affects the selection mechanism as an estimate $\widehat{w}^{(j)}$ of the residual must now be be obtained from the redundant value to perform selection.

A look at some values for ζ, Δ and α explicitly shows their relationship and allows us to make an appropriate choice of parameters for the design. Table 1 enumerates some of these values.

Scheme	$1/2 \le \zeta < 1$	$\Delta = 2\,\zeta - 1$	$\alpha = 1/2(1-\zeta)$
1	17/32	1/16	15/64
2	9/16	1/8	7/32
3	5/8	1/4	3/16
4	3/4	1/2	1/8
5	7/8	3/4	1/16
6	15/16	14/16	1/32

Table 1: Some different values for ζ, Δ and α are enumerated to give a better perspective of their relationship.

Each of the schemes in Table 1 dictates a certain design choice. Note that ζ and α are are both a consequence of the choice in Δ which dictates the overlap between selection intervals and the allowable error in the estimate of the residual $\widehat{w}^{(j)}$. Their values determine necessary scaling of the operands. Recalling the bound

$$\left| w^{(j)} - \widehat{w}^{(j)} \right| \leq \frac{\Delta}{2} \tag{16}$$

which ensures that $\widehat{w}^{(j)}$ will select a correct digit, we first discuss the effects of the redundant encoding of the residual on the estimate $\widehat{w}^{(j)}$. The estimate of the residual should be obtained by truncation of the redundant residual $w_R^{(j)}$ with t fractional digits, on which a short CPA is used to get conventional form, i.e. $\widehat{w}^{(j)} = CPA(\lfloor w_R^{(j)} \rfloor_t)$ where we define $\lfloor x \rfloor_t$ as x truncated to t fractional bits. The choice of redundant encoding will affect the error $|w^{(j)} - \widehat{w}^{(j)}|$ as discussed below,

Carry-save: Since a carry-save encoding has only a positive truncated magnitude, we define errors $\epsilon_{\max} = \max(w^{(j)} - \lfloor w_R^{(j)} \rfloor_t) = 2^{-t+1} - ulp$, $\epsilon_{\min} = \min(w^{(j)} - \lfloor w_R^{(j)} \rfloor_t) = 0$. Then error of truncation is therefore not symmetric, with $\epsilon_{\min} \leq \epsilon \leq \epsilon_{\max}$, $0 \leq |w^{(j)} - \widehat{w}^{(j)}| < \epsilon_{\max} = 2^{-t+1}$.

Borrow-save: A borrow-save encoding means that both a positive and a negative error are possible after truncation. Define errors, $\epsilon_{\min} = \min(w^{(j)} - \lfloor w_R^{(j)} \rfloor_t) = -2^{-t} + ulp$, $\epsilon_{\max} = \max(w^{(j)} - \lfloor w_R^{(j)} \rfloor_t) = 2^{-t} - ulp$ then we can see that the error in truncation is symmetric, $\epsilon_{\min} \leq \epsilon \leq \epsilon_{\max}$, $|w^{(j)} - \widehat{w}^{(j)}| < \epsilon = 2^{-t}$.

These errors and Eqn. (16) dictate the number of fractional positions to which the redundant residual should be truncated, i.e. $\max(|\epsilon_{\min}|, |\epsilon_{\max}|) \leq \Delta/2$. That is, for a carry-save encoding of the residual: $t \geq 2 - \log_2(\Delta)$, and for a borrow save implementation, $t \geq 1 - \log_2(\Delta)$. In the designs that follow we have used Scheme 3 as the choice of implementation parameters. At the time of this writing, the different schemes have still not been evaluated but the currently implemented scheme was chosen on the basis that it avoids the extreme points in the schemes: a small overlap requires a larger CPA to calculate the estimate, but will allow for more relaxed bounds on ζ and α, while a larger overlap requires a less precise estimate (and shorter CPA) while constraining ζ and α. Given that $\Delta = 1/4$, the residual will have to be truncated to four fractional positions ($t = 4$) in order to ensure correct selection. This means a CPA of length seven (since $2|w^{(j)}| < 26/8$, and requires at least three integer positions) must be employed to compute $\widehat{w}^{(j)}$.

The original selection function in Eqn. (9) requires a full precision comparison of $w^{(j)}$ to constants $-1/2$ and $1/2$ in order to perform selection. Bounds L_k and U_k are used to describe the minimum and maximum values for which digit k can be selected—bounds U_{-1}, L_0, U_0 and L_1 have been shown for a non-overlapping selection case in Fig. 2. Given $\Delta = 1/4$, Fig. 2 shows the permissible overlapping regions as shaded, and the actual overlapping regions as striped. The motivation behind overlapping selection regions is not so that either digit could be selected; that is merely a consequence and may allow optimization of the selection function. It is to allow a low-precision estimate of the residual to suffice for selection purposes, thus removing the constraint for full precision evaluation of the residual before selection is performed. In fact we may altogether ignore the fact that overlapping regions allow multiple selection choices and use $-1/2$ and $1/2$ as breakpoints for selection values as dictated by eqn. (9) if this results in a more efficient implementation of the selection function.

Given that we have a carry-save encoding of the residual, the estimate can only have a positive error of less than $1/8$. Since the error can only be positive, then the error can only affect the upper bounds U_k. This has again been demonstrated in the figure, by shifting rightward U_0 and U_{-1} for the overlapping regions by $1/8$ (note that it is actually shifted by less than $1/8$ so the closed ended interval U_{-1} becomes open). From the overlapping regions we can see that 111001 ($-7/16$) allows for selection $d^{(j)}$ of both -1 and 0; likewise, 001000 ($1/2$), and 001001 ($9/16$) both allow for selection $d^{(j)}$ of 1 and 0.

One could find the minimal logic expressions implementing this selection scheme but for an FPGA implementation this is rather useless because LUTs (Look Up Tables) are used to implement functions, and for the particular architecture of choice a 6-input LUT is available as a design atom allowing for an efficient direct implementation.

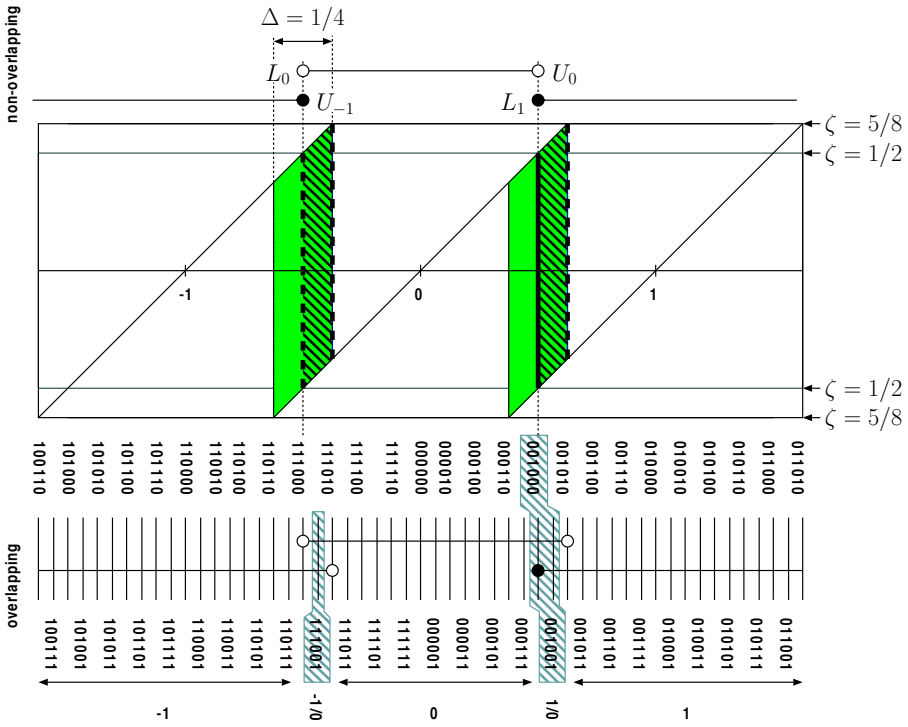


Figure 2: Robertson's diagram for $\zeta = 5/8$, and $\Delta = 1/4$

2.5 ZEU

The Zero-term Evaluation Unit (ZEU) implements,

$$w^{(j)} = 2\left[w^{(j-1)} - d^{(j-1)}\right] \tag{17}$$

$$d^{(j)} = S(w^{(j)}) \tag{18}$$

where S is the selection mechanism discussed in section 2.4. Since this recurrence contains no terms to be added to the residual and only adjusts the residual by $d^{(j-1)}$, then the residual can be kept in non-redundant form.

2.5.1 Design

The design is based on a "bit-slice" approach whereby the unit is partitioned into slices of several bits to allow for a uniform and systematic hardware layout. Even for FPGA designs, this can have an effect if each slice is laid out efficiently with their interconnection kept in mind. We have not laid out any of these components by hand, and we relied on Altera Quartus[17] to perform such operations. Even so, there is a very practical use for such a slice-based design. When the CAD tool fits the net list to the device, the algorithms employed utilize heuristics and random seeds which can alter the fitting for each compilation. This can be very frustrating for a designer which intends to keep the performance of the initial fitting unaltered if, for example, the precision of the design needs to be extended–this is further exacerbated if moving the previously fitted unit will violate timing constraints with respect to other units. For this reason, partitioning the unit into slices allows the designer to fit a particular slice and retain the routing and fitting even if the precision must be extended. This becomes clearer after we present the design.

Figure 3 shows the different slices in the ZEU design, which constitutes the most-significant-slice, (MSS) the repeat-slice (RS) and the least-significant-slice (LSS). The MSS shown in Fig. 3(a) contains the critical path for the ZEU, as its the only unit in which signals traverse logic prior to being registered. The "Adjustment" block in the MSS simply subtracts the value of $d^{(j-1)}$ from the integer portion of the residual $w^{(j-1)}$, i.e.

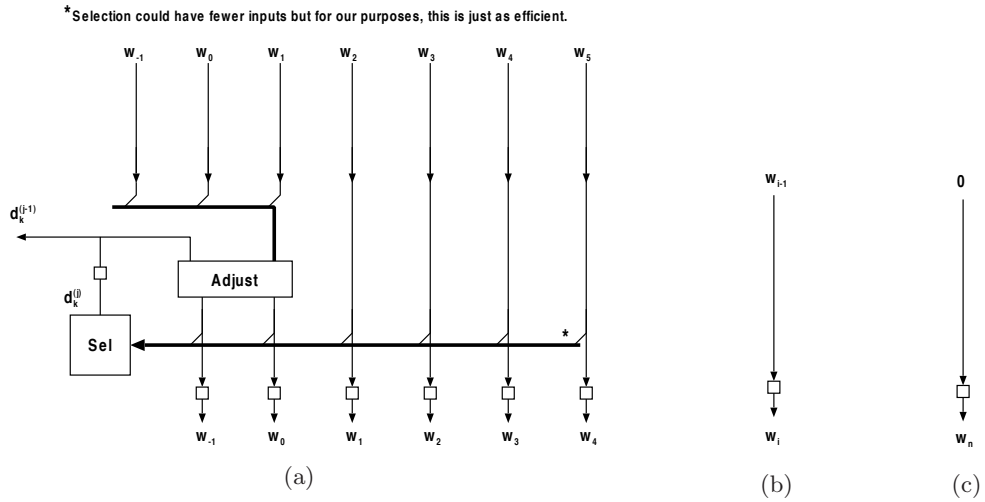


Figure 3: ZEU slices. The small boxes shown denote registers which are initialized with $w^{(0)}$. (a) most significant slice, this unit is only required once at the most significant position (b) repeat slice, repeats enough times to meet the required precision. (c) least significant slice is only required once at the least significant position.

$W_{-1}W_0 = W_{-2}W_{-1}W_0 - 2\,d$. This is performed via a table lookup as shown in Table 2. A similar adjustment is described in.[15] The selection function performs selection as described in section 2.4.

$W_{-2}W_{-1}W_0$	$d^{(j)}$	$W_{-1}W_0$
000	0	00
001	0	01
110	0	10
111	0	11
000	-1	10
001	-1	11
010	-1	00
011	-1	01
100	1	10
101	1	11
110	1	00
111	1	01

Table 2: Adjustment of the residual by the selected digit.

2.5.2 Performance

Here we present the area and delay of the ZEU unit with varying degrees of precision (fractional). As pointed out in 2.5.1 any variations in the delay is due to different fitting results obtained from each run. We find it more constructive to report typical delay results from compilation than to retain the place and route results of the first MSS and report equal delays for all precisions.

Precision	8	16	32	64	128
Clock Freq. (Mhz)	535.62	491.88	530.79	510.2	422.3
ALUTs	8	8	8	8	8
Regs	12	20	36	68	132

Table 3: Performance of the ZEU unit for Altera Stratix II,[17] compiled with Altera Quartus[17] 8.0.

Starting with the ALUTs reported, we see that this parameter remains constant for all precisions, this is because the only logic in the ZEU is contained in the MSS for which there is only one instance. The register usage increases linearly with precision as expected. As discussed the delays vary for each compilation, however, if one takes the best reported delay from all instances, and retains the place and route results from that run, then delay for all precisions could perform as well. Of course, this is assuming that no single slice and its interconnection to adjacent slices is placed and routed so poorly as to change the critical path.

2.6 PEU

The Polynomial Evaluation Unit (PEU) derives its name from the fact that it is the recurrence used to evaluate a complex polynomial as described in 10. Please refer to Section 2.5 for an overall view of the design and performance results.

2.6.1 Design

The PEU implements

$$w^{(j)} = 2\left[w^{(j-1)} - d^{(j-1)} + \sigma_1 d_{\sigma_1}^{(j-1)}\right] \tag{19}$$

However, in this implementation a carry-save encoding of the residual is used, and the sum performed in the recurrence is actually done in redundant form. That is, eqn. (19) is actually implemented as

$$(ws^{(j)}, wc^{(j)}) = 2\left[ws^{(j-1)} + wc^{(j-1)} - d^{(j-1)} + \sigma_1 d_{\sigma_1}^{(j-1)} + \sigma_2 d_{\sigma_2}^{(j-1)}\right] \tag{20}$$

$$d^{(j)} = S(\widehat{w}^{(j)}) \tag{21}$$

$$\widehat{w}^{(j)} = CPA\left(\lfloor ws^{(j)}, wc^{(j)} \rfloor_4\right) \tag{22}$$

This means that the recurrence requires a [4:2] reduction module. Inputs σ_1 and σ_2 must also be have their multiples generated (i.e. $\sigma_i d_{\sigma_i}^{(j)}$). The implementations for the MSS, RS and LSS are shown in figures 4(a), 4(b) and 4(c) respectively.

2.6.2 Performance

Here we present the delay and area of the PEU unit for varying values of precision (fractional) in Table 4:

Precision	8	16	32	64	128
Clock Freq. (Mhz)	258.93	265.11	274.80	250.31	241.60
ALUTs	57	105	205	394	778
Regs	15	31	63	127	255

Table 4: Performance of the PEU unit for Altera Stratix II,[17] compiled with Altera Quartus[17] 8.0.

2.7 REU

The Rational Evaluation Unit (REU) derives its name from being the recurrence used to compute rational functions. This unit implements,

$$ws^{(j)}, wc^{(j)} = 2\left[ws^{(j-1)} + wc^{(j-1)} - d^{(j-1)} + \sigma_1 d_{\sigma_1}^{(j-1)} + \sigma_2 d_{\sigma_2}^{(j-1)} + \sigma_3 d_{\sigma_3}^{(j-1)} + \sigma_4 d_{\sigma_4}^{(j-1)}\right] \tag{23}$$

$$d^{(j)} = S(\widehat{w}^{(j)}) \tag{24}$$

$$\widehat{w}^{(j)} = CPA\left(\lfloor ws^{(j)}, wc^{(j)} \rfloor_4\right) \tag{25}$$

which uses a [6:2] reduction module.

2.7.1 Design

The design of the REU follows that of the ZEU and PEU as described in Sections 2.5.1 and 2.6.1 respectively. Figures 5(a), 5(b), and 5(c) show the MSS, RS and LS of the REU respectively.

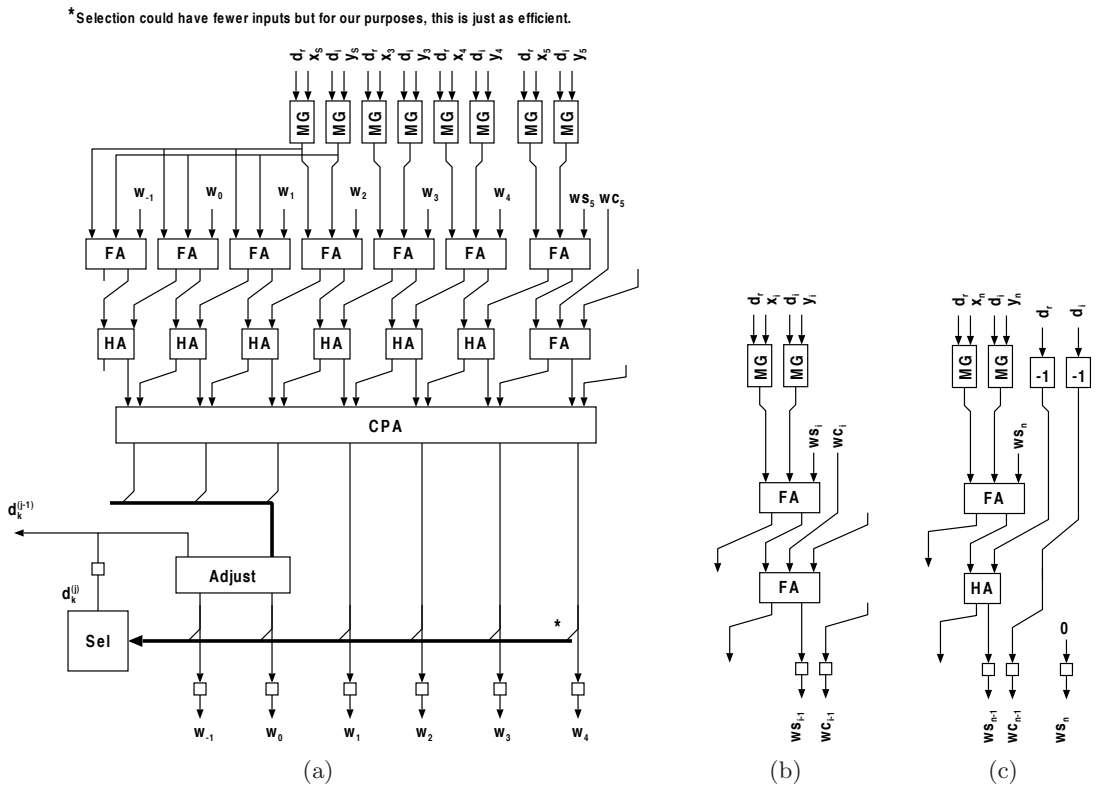


Figure 4: PEU slices. (a) the most-significant-slice, (b) repeat-slice, (c) the least-significant-slice. In the figure shown σ_1 has been labeled as the x input and σ_2 as the y input.

2.7.2 Performance

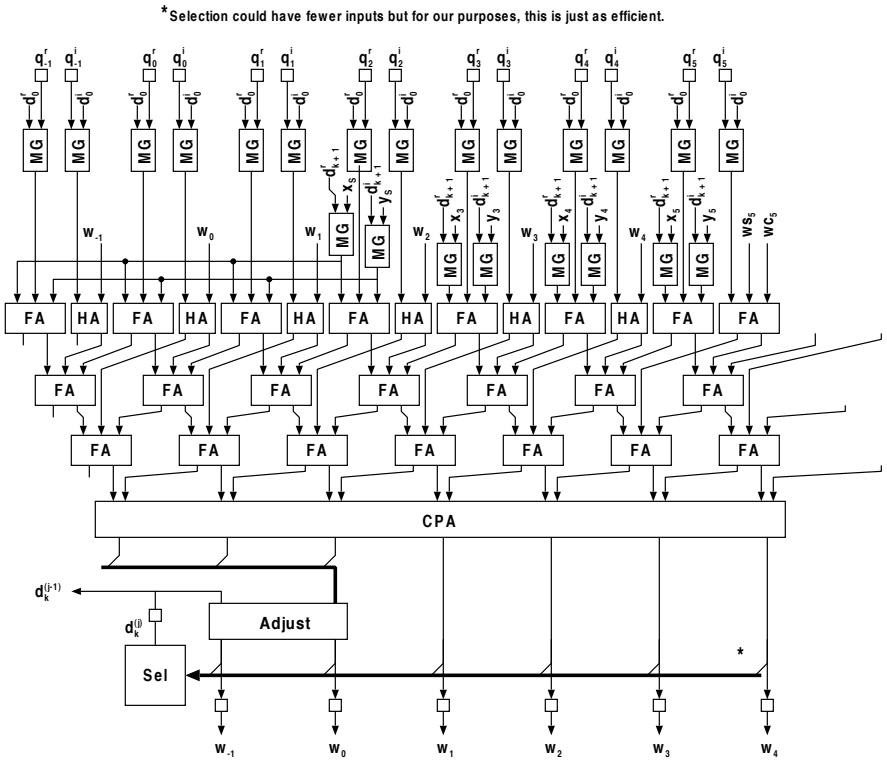
Precision	8	16	32	64	128
Clock Freq. (Mhz)	235.18	229.36	230.63	224.01	220.7
ALUTs	120	208	382	729	1435
Regs	15	31	63	127	255

Table 5: Performance of the REU unit for Altera Stratix II,[17] compiled with Altera Quartus[17] 8.0.

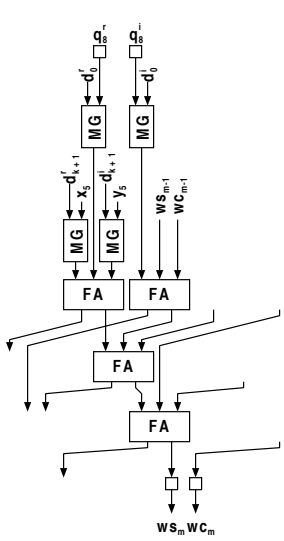
3. METHODOLOGY

The designs implemented in this paper are written using VHDL and parametrized to allow for generating different instances of the design based on specifications. The problem with generative features of VHDL is that it can over-complicate the code and leave the source base rather unreadable for other designers. More so, the VHDL generative features have limited capability, and depend on generic parameters to entities which also have limited capability with regard to how they can be initialized. There are obviously work-arounds and recipes which designers have utilized to get the desired effect, but this makes the source difficult to read and often times the real functionality becomes buried in other code.

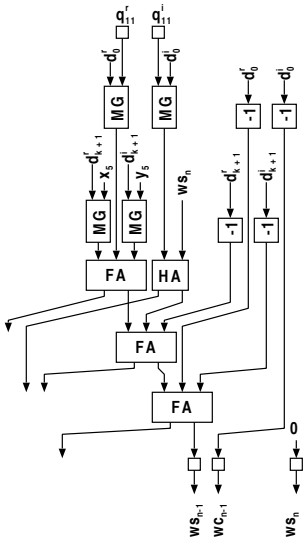
Another desired capability is to co-simulate the arithmetic at an algorithmic level to check and verify simulation results. Of course, again, VHDL is *capable* of delivering this functionality, but there are much nicer and more pragmatic tools available which can dramatically simplify a designer's life as well as reduce "reinventing the wheel".



(a)



(b)



(c)

Figure 5: REU slices. (a) most-significant-slice (b) repeat-slice (c) least-significant-slice. In this diagram, σ_1, σ_2, σ_3 and σ_4 have been labeled as x, y, q^r and q^i

In order to achieve both these desired effects, we have utilized the Python[18] programming language, NumPy,[20] a Python numerical package for scientific computing as well as Mako,[19] a Python template engine to create pragmatic generators. Using Python and NumPy, one can easily obtain algorithmic solutions to the design being implemented. A Python script can then use the Mako library to interface these results, as well as design parameters to a VHDL template file which after being rendered becomes conventional VHDL.

From our experience using these sets of tools, we have obtained much cleaner VHDL code, and spent a lot less time grappling with obscure bugs.

4. RESULTS

We now summarize performance results for the complex operators CMA, CSS, and CSP.

4.1 CMA

Precision	8	16	32	64	128
Clock Freq. (Mhz)	209.69	189.68	199.12	205.97	217.16
ALUTs	141	237	434	814	1582
Regs	94	174	334	654	1296
Delay / Operation (ns)	42.92	89.62	165.73	315.58	594.03

Table 6: Performance of the CMA unit for Altera Stratix II,[17] compiled with Altera Quartus[17] 8.0.

4.2 CSS

Precision	8	16	32	64	128
Clock Freq. (Mhz)	171.56	184.84	188.11	181.09	181.72
ALUTs	277	454	802	1495	2902
Regs	116	212	404	788	1556
Delay / Operation (ns)	52.46	91.97	175.43	358.94	709.88

Table 7: Performance of the CSS unit for Altera Stratix II,[17] compiled with Altera Quartus[17] 8.0.

4.3 CSP

Precision	8	16	32	64	128
Clock Freq. (Mhz)	161.11	173.16	171.32	188.75	185.53
ALUTs	281	457	806	1497	2906
Regs	118	214	406	790	1558
Delay/Operation (ns)	55.86	98.18	192.62	339.07	695.31

Table 8: Performance of the CSP unit for Altera Stratix II,[17] compiled with Altera Quartus[17] 8.0.

Comparisons with alternative designs using conventional arithmetic operations are in progress.

5. SUMMARY AND FUTURE WORK

In this paper we have presented designs and implementations for a series of complex operators: multiply-add, sum-of-products and sum-of-squares for the Altera Stratix II[17] architecture. The precision independent critical-path of these operators is a very noteworthy feature for long precision operations. The actual clocking frequency may vary based on the quality of the place and route achieved.

Future work entails designing high-radix implementations of such designs to utilize built-in FPGA features such as fast carry-paths. In addition, very long precision designs and designs which can time-multiplex resources are being considered.

We also plan to perform comparisons with designs based on conventional arithmetic approaches.

REFERENCES

[1] T. Aoki, H. Amada, and T. Higuchi. Real/complex reconfigurable arithmetic using redundant complex number systems. *Proc. 13th IEEE Symposium on Computer Arithmetic*, pp.200-207, 1997.

[2] M.D. Ercegovac and J.M Muller, "Complex Multiply-Add and Other Related Operators" , Proceedings of the SPIE Conference, 2007.

[3] M.D. Ercegovac. *A general method for evaluation of functions and computation in a digital computer*. PhD thesis, Dept. of Computer Science, University of Illinois, Urbana-Champaign, 1975.

[4] M.D. Ercegovac. A General Hardware-oriented Method for Evaluation of Functions and Computations in a Digital Computer. *IEEE Trans. Comp.*, C-26(7):667–680, 1977.

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

[6] M.D. Ercegovac and J.-M. Muller. Complex Division with Prescaling of Operands. *IEEE International Conference on Application-Specific Systems, Architectures and Processors*, pp. 293-303, 2003.

[7] M.D. Ercegovac and J.-M. Muller, Design of a complex divider. *Proc. SPIE on Advanced Signal Processing Algorithms, Architectures, and Implementations XII*, pp. 51-59, 2004.

[8] M.D. Ercegovac and J.-M. Muller. Complex Square Root with Operand Prescaling. *IEEE International Conference on Application-Specific Systems, Architectures and Processors*, pp. 293-303, 2004.

[9] M.D. Ercegovac and J.-M. Muller, Solving Systems of Linear Equations in Complex Domain : Complex E-Method. LIP Report No. 2007-2, École Normale Supérieure de Lyon, France.

[10] M.D. Ercegovac and J.-M. Muller, A Hardware-Oriented Method for Evaluating Complex Polynomials. *IEEE International Conference on Application-Specific Systems, Architectures and Processors*, 2007.

[11] R.D. McIlhenny, *Complex Number On-line Arithmetic for Reconfigurable Hardware: Algorithms, Implementations, and Applications*, Ph.D. Dissertation, Computer Science Department, University of California, 2002.

[12] V. Oklobdzija, D. Villeger and T. Soulas, An Integrated Multiplier for Complex Numbers. *J. of VLSI Signal Processing*, vol.7, no. 3, pp.213-222, May 1994.

[13] J.E. Robertson, A new class of digital division methods. *IRE Transactions on Electronic Computers,* EC-7(3):88-92.

[14] A.F. Tenca, M.D. Ercegovac. Design of high-radix digit slices for online computations. In *SPIE Conference on High-Speed Computing, Digital Signal Processing, and Filtering Using Reconfigurable Logic*, Bellingham, 1996.

[15] D. Tullsen and M.D. Ercegovac. Design and implementation of an on-line algorithm. In *Proc. SPIE Conference on Real-Time Signal Processing*, San Diego, August 1986.

[16] B.W.Y. Wei, H. Du, and H. Chen, A Complex-Number Multiplier Using Radix-4 Digits. *Proc. 12th IEEE Symposium on Computer Arithmetic*, pp. 84-90, 1995

[17] http://www.altera.com/

[18] http://www.python.org/

[19] http://www.makotemplates.org/

[20] http://numpy.scipy.org/