Robert SMYK[1], **Maciej CZYŻAK**[2], **Zenon ULMAN**[2]
[1] PAŃSTWOWA WYŻSZA SZKOŁA ZAWODOWA W ELBLĄGU, INSTUTUT INFORMATYKI
[2] POLITECHNIKA GDAŃSKA

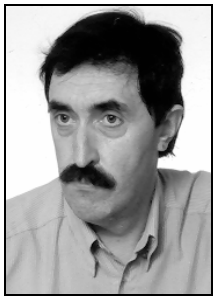
Design of a complex multiplier based on the convolution with the use of the polynomial residue number system

M.Sc. Robert SMYK

He received his M.S. degree in control engineering in 2002 from the Gdansk University of Technology. Since 2002 he has pursued his PhD study in the Department of Electrical and Control Engineering. Since 2004 he has been an instructor in the Department of Informatics at the Elblag Higher Professional College (PWSZ Elbląg). His research area includes algoritms of digital signal processing and computer arithmetic.

e-mail: r.smyk@ely.pg.gda.pl

Ph.D. Maciej CZYŻAK



He received his Ph.D. in 1985 from the Gdansk University of Technology(GUT). From 1984 to 1990 he was a researcher and lecturer at the Institute of Informatics, Faculty of Electronics, GUT. Since 1994 he has been a lecturer of informatics at the Faculty of Electrical and Control Engineering ,GUT. His research interests are in fast digital signal processing, computer arithmetic and VLSI design.

e-mail: m.czyzak@ely.pg.gda.pl

D.Sc. Ph.D. Zenon ULMAN

He received his Ph.D. in 1979 from the Gdansk University of Technology(GUT). From 1972 he has been a researcher and lecturer at the Faculty of Electrical and Control Engineering, GUT. In 2000 he obtained the Doctor of Science degree from the Technical University of Wroclaw. His scientific interests include computer arithmetic, number systems and fast digital signal processing.

e-mail: z.ulman@ely.pg.gda.pl

Abstract

The complex multiplication is one of the basic operations in digital signal processing. In this work the design procedure of the complex multiplier based on the well-known decomposition algorithm of Skavantzos and Stouraitis is presented. The algorithm makes use of encoding *n*-bit numbers as polynomials of degree 7 in the ring of polynomials modulo (x^8-1) with $n/4$-bit coefficients. The complex multiplication is carried out as an eight point cyclic convolution. The design procedure is illustrated by the computational example and design of a small multiplier.

Keywords: digital signal processing, complex multiplication, polynomial residue number system.

Projektowanie mnożnika zespolonego oparte na splocie z użyciem wielomianowego systemu resztowego

Streszczenie

Mnożenie zespolone jest jedną z podstawowych operacji w cyfrowym przetwarzaniu sygnałów. W niniejszej pracy przestawiono metodę projektowania mnożników zespolonych opartą na znanym algorytmie dekompozycji Skavantzosa and Stouraitisa. W algorytmie tym stosuje się kodowanie liczb *n*-bitowych jako wielomianów stopnia 7 w pierścieniu wielomianów modulo (x^8-1) ze współczynnikami $n/4$-bitowymi. Mnożenie zespolone jest następnie realizowane jako 8-punktowy splot cykliczny. Proponowaną metodę projektowania zilustrowano przykładem obliczeniowym oraz przykładowym projektem mnożnika.

Słowa kluczowe: cyfrowe przetwarzanie sygnałów, mnożenie zespolone, wielomianowy system resztowy.

1. Introduction

The complex multiplication is one of the basic arithmetic operations in digital signal processing. The most important design goals while designing complex multipliers can be maximization of the throughput while maintaining a possibly small area, or minimization of the area-time complexity. The design issues also encompass the power dissipation in VLSI multipliers. Using a direct approach a binary complex $n \times n$ bit multiplier can be based on four $n \times n$ bit binary multipliers and two adders. There also exists an algebraic transform proposed by Blahut [1], that allows to save one real multiplication at the cost of two more additions and one subtraction. The theoretical lower bound on the delay for an $n \times n$-bit binary multiplication was given by Winograd [2] and for the area-time complexity by Brent and Kung [3], but in practice they have small impact on multiplier design. The binary multiplication can be implemented very effectively using only shifts and additions, but the delay may be prohibitive. A common alternative to this solution is the generation of partial products with a subsequent addition using a two-dimensional full-adder(FA) array and a final two-operand binary adder. Such an array is usually transformed into square that gives an array multiplier. The delay of such a multiplier consists roughly of the delay of the FA array that is proportional to n and the carry-propagate adder delay. The binary adder is used commonly in the ripple-carry or carry look-ahead form. A faster but less regular multiplier can be obtained by summing the partial product with the use of the Wallace tree [5]. In this case the delay is proportional to log n. The complex multiplier considered in this paper has a different structure from those based on binary multipliers discussed above. Here the complex n-bit numbers are encoded as polynomials of degree seven in the rings of polynomial modulo (x^8-1). The complex multiplication is performed as an 8-point cyclic convolution. In the first step n-bit complex numbers are decomposed into $n/4$-bit segments. Using these segments the coefficients of the input polynomials are calculated. In the second step all 64 possible products of polynomial coefficients are calculated. In this step $n/4 \times n/4$, $n/4 \times n/4+1$ and $n/4+1 \times n/4+1$ multipliers are needed. Using these products, the coefficients of the circular convolution are computed with the subsequent calculation of the real and imaginary parts of the complex product. In this paper, $n = 4$ is assumed. The aim of this work is to show the design procedure of the complex multiplier based on the above presented principle and consider the issue of its time-hardware complexity. This algorithm may be useful as a design tool for large multipliers, but the characteristic features of its hardware realization can also be determined using small n. Therefore we shall use $n = 4$. This work is a revised and extended version of [6].

2. The complex multiplication algorithm for n-bit numbers

Consider multiplication $z = x \cdot y$ of two n-bit complex numbers with $x = x_r + jx_i$ and $y = y_r + jy_i$. In order to decrease the

wordlength the real and imaginary parts are decomposed into *1*-bit segments in the following manner [4]

$$x = \left(x_{r3}2^{3n/4} + x_{r2}2^{n/2} + x_{r1}2^{n/4} + x_{r0}\right) + \\ + j\left(x_{i3}2^{3n/4} + x_{i2}2^{n/2} + x_{i1}2^{n/4} + x_{i0}\right), \tag{1}$$

$$y = \left(y_{r3}2^{3n/4} + y_{r2}2^{n/2} + y_{r1}2^{n/4} + y_{r0}\right) + \\ + j\left(y_{i3}2^{3n/4} + y_{i2}2^{n/2} + y_{i1}2^{n/4} + y_{i0}\right). \tag{2}$$

After such decomposition the complex numbers can be expressed as 7th order polynomials. This makes possible to represent the multiplication of complex numbers as a multiplication of polynomials. The complex number x can be defined in the polynomial form as follows [4]:

$$W(k) = w_0 2^{3n/4}k^0 + \frac{\sqrt{2}}{2}w_1 2^{n/2}k^1 + w_2 2^{3n/4}k^2 + \frac{\sqrt{2}}{2}w_3 2^{n/2}k^3 + \\ + w_4 2^{n/4}k^4 + \frac{\sqrt{2}}{2}w_5 2^0 k^5 + w_6 2^{n/4}k^6 + \frac{\sqrt{2}}{2}w_7 2^0 k^7, \tag{3}$$

with $k = \sqrt{2}/2 + j\sqrt{2}/2$.

The polynomial coefficients can be expressed as linear combinations of the individual segments

$$\begin{aligned} w_0 &= x_{r3}, \quad w_1 = x_{r2} + x_{i2}, \\ w_2 &= x_{i3}, \quad w_3 = -x_{r2} + x_{i2}, \\ w_4 &= -x_{r1}, \quad w_5 = x_{r0} - x_{i0}, \\ w_6 &= -x_{i1}, \quad w_7 = x_{r0} - x_{i0}. \end{aligned} \tag{4}$$

These coefficients can be computed by equating the respective expressions:

$$w_0 2^{3n/4}\left(\frac{\sqrt{2}}{2} + j\frac{\sqrt{2}}{2}\right)^0 + w_2 2^{3n/4}\left(\frac{\sqrt{2}}{2} + j\frac{\sqrt{2}}{2}\right)^2 = \\ = x_{r3}2^{3n/4} + jx_{i3}2^{3n/4}, \tag{5a}$$

$$\frac{\sqrt{2}}{2}w_1 2^{n/2}\left(\frac{\sqrt{2}}{2} + j\frac{\sqrt{2}}{2}\right)^1 + \frac{\sqrt{2}}{2}w_3 2^{n/2}\left(\frac{\sqrt{2}}{2} + j\frac{\sqrt{2}}{2}\right)^3 = \\ = x_{r2}2^{n/2} + jx_{i2}2^{n/2}, \tag{5b}$$

$$w_4 2^{n/4}\left(\frac{\sqrt{2}}{2} + j\frac{\sqrt{2}}{2}\right)^4 + w_6 2^{n/4}\left(\frac{\sqrt{2}}{2} + j\frac{\sqrt{2}}{2}\right)^6 = \\ = x_{r1}2^{n/4} + jx_{i1}2^{n/4}, \tag{5c}$$

$$\frac{\sqrt{2}}{2}w_5\left(\frac{\sqrt{2}}{2} + j\frac{\sqrt{2}}{2}\right)^5 + \frac{\sqrt{2}}{2}w_7\left(\frac{\sqrt{2}}{2} + j\frac{\sqrt{2}}{2}\right)^7 = x_{r1} + jx_{i1}. \tag{5d}$$

The same scheme of decomposition for the second complex number y is used. The respective polynomial obtained for y will be denoted as $V(k)$. Then the product of x and y can be computed as

$$Q(k) = \left|W(k)\cdot V(k)\right|_{x^8-1}, \tag{6}$$

which is equivalent to 8-point circular convolution, where

$$Q(k) = q_0 k^0 + \frac{\sqrt{2}}{2}q_1 k^1 + q_2 k^2 + \frac{\sqrt{2}}{2}q_3 k^3 + q_4 k^4 + \\ + \frac{\sqrt{2}}{2}q_5 k^5 + q_6 k^6 + \frac{\sqrt{2}}{2}q_7 k^7. \tag{7}$$

and

$$Q_{re} = q_0 + q_1/2 - q_3/2 - q_4 - q_5/2 + q_7/2, \tag{8a}$$

$$Q_{im} = q_1/2 + q_2 + q_3/2 - q_5/2 - q_6 - q_7/2. \tag{8b}$$

The relationships (8a) and (8b) can be obtained by inserting $k = \sqrt{2}/2 + j\sqrt{2}/2$ into (7).

The coefficients of $Q(k)$ are the entries on the main diagonal of the matrix Q defined as

$$Q = Z \cdot G^T \tag{9}$$

with

$$G = \begin{bmatrix} 2^6 & 2^1 & 2^4 & 2^1 & 2^2 & 2^1 & 2^4 & 2^1 \\ 2^5 & 2^5 & 2^3 & 2^3 & 2^1 & 2^1 & 2^3 & 2^3 \\ 2^6 & 2^3 & 2^6 & 2^1 & 2^2 & 2^{-1} & 2^2 & 2^1 \\ 2^5 & 2^5 & 2^5 & 2^5 & 2^1 & 2^1 & 2^1 & 2^1 \\ 2^4 & 2^3 & 2^6 & 2^3 & 2^4 & 2^{-1} & 2^2 & 2^{-1} \\ 2^3 & 2^3 & 2^5 & 2^5 & 2^3 & 2^3 & 2^1 & 2^1 \\ 2^4 & 2^1 & 2^4 & 2^3 & 2^4 & 2^1 & 2^4 & 2^{-1} \\ 2^3 & 2^3 & 2^3 & 2^3 & 2^3 & 2^3 & 2^3 & 2^3 \end{bmatrix} \tag{10}$$

and

$$Z = W \cdot V, \tag{11}$$

where $W = \left\lfloor w_{|i-j|_8} \right\rfloor$, $i, j = 0,1,2,...,7.$ and V is a diagonal matrix $diag(v_0, v_1, ..., v_7)$ with v_i being the elements of the coefficient vector of the complex number y, appearing as w_i in (5).

3. Realization of the algorithm for *n* = 4

The algorithm consists of the following steps:

i) computation of w_i and v_i, $i = 0,1,...,7$,

ii) calculation of the matrix Z (products $w_i \cdot v_j$, $i, j = 0, 1, 2,..., 7$),

iii) computation of q_i, $i = 0, 1, 2,...,7$,

iv) determination of Q_{re} and Q_{im} by (8a) and (8b), respectively.

3.1. Computation of w_i and v_i

The coefficients w_i, v_i, $i = 0,1,...,7$ can be computed as given in (4), x_{kr}, x_{ki}, $k = 0,1,2,...,7$, are 1-bit. Thus w_i and v_i may be at most 2-bit with the extra sign bit. The sign-magnitude form is assumed. The coefficients are computed using logic functions. The logic equations are given only for w_i, $i = 0,1,...,7$, since for v_i they have the same form with x_{rk}, x_{ik} replaced with y_{rk}, y_{ik}.

a) $w_0^{(0)} = x_{r3}$,

b) $w_1^{(1)} = x_{r2} \cdot x_{i2}$, $w_1^{(0)} = x_{r2} \oplus x_{i2}$,

c) $w_2^{(0)} = x_{r3}$,

d) $w_3^{(1)} = x_{r2} \cdot \overline{x}_{i2}$, $w_3^{(0)} = x_{r2} \oplus x_{i2}$,

e) $w_4^{(1)} = 1$, $w_4^{(0)} = x_{r1}$,

f) $w_5^{(2)} = 1$, $w_5^{(1)} = x_{0r} \oplus x_{0i}$, $w_5^{(0)} = x_{r0} \cdot x_{i0}$,

g) $w_6^{(1)} = 1$, $w_6^{(0)} = x_{i1}$,

h) $w_7^{(1)} = \overline{x}_{r0} \cdot x_{i0}$, $w_7^{(0)} = x_{r0} \oplus x_{i0}$.

3.2. Calculation of the matrix *Z*

In this step all products $w_i \cdot v_j$, i, j=0, 1, 2,...,7 have to be computed. The operands can be classified in the following manner

i) *1*-bit unsigned w_0, w_2

ii) *2*-bit unsigned w_1

iii) 1-bit signed w_3, w_4, w_6, w_7

iv) 2-bit signed w_5

Due to the sign-magnitude representation, the signs can be multiplied separately, hence three types of simple multipliers are needed, namely 1-bit multiplier (Type I), 2×1-bit multiplier (Type II), a 2×2-bit multiplier (Type III), signed 1-bit multiplier (Type IV) and signed 2×2-bit multiplier (Type V). The respective logic functions have the following form:

Type I: (1×1-bit unsigned arguments)

$$h_I^{(0)} = a^{(0)}b^{(0)}$$

Type II: (2×1-bit unsigned arguments)

$$h_{II}^{(1)} = a^{(0)}b^{(1)}, \quad h_{II}^{(0)} = a^{(0)}b^{(0)}$$

Type III: (2×2-bit unsigned arguments)

$$h_{III}^{(2)} = a^{(1)}b^{(1)}\overline{a}^{(0)}, \quad h_{III}^{(1)} = a^{(1)}\overline{a}^{(0)}b^{(0)} + a^{(1)}a^{(0)}b^{(1)},$$

$$h_{III}^{(0)} = \overline{a}^{(1)}a^{(0)}b^{(0)},$$

Type IV: (1×1-bit signed arguments)

$$h_{IV}^{(1)} = sign\,1 \oplus sign\,2, \quad h_{IV}^{(0)} = a^{(0)}b^{(0)},$$

Type V: (2×1-bit signed arguments)

$$h_{IV}^{(1)} = sign\,1 \oplus sign\,2, \quad h_I^{(0)} = a^{(0)}b^{(0)},$$

The sizes and signs of $w_i \cdot v_j$ for all combinations of *i* and *j* are given in Table 1.

Tab. 1. The bit sizes and signs of the partial products
Tab. 1. Rozmiary w bitach i znaki iloczynów częściowych

	w_0	w_1	w_2	w_3	w_4	w_5	w_6	w_7
v_0	1,+	2,+	1,+	1,±	1,-	2,-	1,-	1,±
v_1	2,+	3,+	2,+	2,±	2,-	3,±	2,-	2,±
v_2	1,+	2,+	1,+	1,±	1,-	2,-	1,-	1,±
v_3	1,±	2,±	1,±	1,±	1,±	2,±	1,±	1,±
v_4	1,-	2,-	1,-	1,±	1,+	2,+	1,+	1,±
v_5	2,-	3,±	2,±	2,±	2,+	3,+	1,±	2,±
v_6	1,-	2,-	1,-	1,±	1,+	1,±	1,+	1,±
v_7	1,±	2,±	1,±	1,±	1,±	2,±	1,±	1,±

3.3. Computation of q_i, *i* = 0, 1, 2,...,7

The computation of q_i, i =0, 1, 2,...,7. by (9) requires 8-operand signed addition for each *i*. The operands are at most 3-bit with the sign and varying weights as it is seen in the rows of the matrix G. The computation of q_i involves the terms with a fixed sign and terms with a variable sign. The use of 2's complement is ineffective due to the small number of nonzero bits in the individual terms hence the grouping of addends is performed in such a manner, that two groups are formed, one with a fixed positive sign and the other with a fixed negative sign. The variable sign terms are attached to both groups and they are redirected to the proper group in dependence of the sign. This redirection is performed by simple demultiplexers controlled by the sign of the term.

The general formula for the computation of q_i can be written as

$$q_i = \sum_{j=0}^{8} z_{ij} \cdot g_{ij} \tag{12}$$

Hence, for instance, for q_0 we receive the sums of the positive and negative terms of q_0, q_0^+, q_0^-, respectively:

$$q_0^+ = (\overline{S}(z_{0,1}) \cdot |z_{0,1}| + \overline{S}(z_{0,3}) \cdot |z_{0,3}| + \overline{S}(z_{0,5}) \cdot |z_{0,5}| + \\ + \overline{S}(z_{0,7})) \cdot 2^1 + z_{0,4} \cdot 2^2 + z_{0,0} \cdot 2^6 \tag{13a}$$

$$q_0^- = (S(z_{0,1}) \cdot |z_{0,1}| + S(z_{0,3}) \cdot |z_{0,3}| + S(z_{0,5}) \cdot |z_{0,5}| + \\ + S(z_{0,7})) \cdot 2^1 + (z_{0,2} + z_{0,6}) \cdot 2^4 \tag{13b}$$

In the above formulas the dashed symbols represent the negation and $S(\cdot)$-denotes the sign of the term in parentheses with 0 for the positive sign and 1 for the negative sign. After the computation of (13a) and (13b), q_0^- is transformed into its 2's complement form and the $q_0^+ + q_0^-$ addition is made. The structure of an such adder is shown in Fig 1.

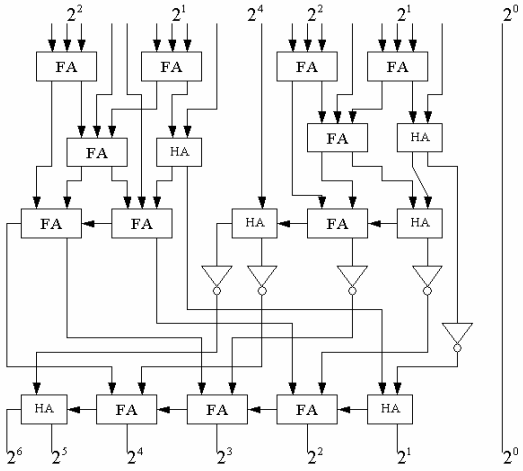


Fig. 1. The full adder (FA) array for the computation of q_0

Rys. 1. Sumator do obliczania współczynnika q_0

The adders for Q_{re} and Q_{im} - as given by (8a) and (8b), respectively, have been designed in the standard form [5] of 6-operand Wallace trees and the final carry-propagate adder has the ripple-carry form.

4. Numerical example

Assume two 4-bit complex numbers x=5+j12 and y=12+j9, with the product $z = x \cdot y = -39 + j177$.

As shown above the numbers can be represented in the form of polynomials $W(k)$ and $V(k)$ with the coefficients computed by (4). The product of these polynomials allows to obtain the $Q(k)$ polynomial that represents the product of two complex numbers. In the first step we have to decompose in accordance with (1) and (2) the real and imaginary parts of both numbers. For n=4 such decomposition leads to the segmentation of real and imaginary parts of both numbers into 1-bit segments. For the assumed complex numbers we obtain

$$x = 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 + j(1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0)$$
$$y = 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0 + j(1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0)$$

Thus we get

$$x_{r3}=0,\quad x_{r2}=1,\quad x_{r1}=0,\quad x_{r0}=1$$
$$x_{i3}=1,\quad x_{i2}=0,\quad x_{i1}=1,\quad x_{i0}=1$$
$$y_{r3}=1,\quad y_{r2}=1,\quad y_{r1}=0,\quad y_{r0}=0$$
$$y_{i3}=1,\quad y_{i2}=0,\quad y_{i1}=0,\quad y_{i0}=1$$

The coefficients of $W(k)$ and $V(k)$ are

$$w_0 = x_{r3} = 0,\quad w_1 = x_{r2} + x_{i2} = 1,$$
$$w_2 = x_{i3} = 1,\quad w_3 = -x_{r2} + x_{i2} = -1,$$
$$w_4 = -x_{r1} = 0,\quad w_5 = x_{r0} - x_{i0} = -2,$$
$$w_6 = -x_{i1} = -1,\quad w_7 = x_{r0} - x_{i0} = 0.$$

and

$$v_0 = y_{r3} = 1,\quad v_1 = y_{r2} + y_{i2} = 1,$$
$$v_2 = y_{i3} = 1,\quad v_3 = -y_{r2} + y_{i2} = -1,$$
$$v_4 = -y_{r1} = 0,\quad v_5 = y_{r0} - y_{i0} = -1,$$
$$v_6 = -y_{i1} = 0,\quad v_7 = y_{r0} - y_{i0} = -1.$$

The circular matrix W has the following form

$$W = \begin{bmatrix} 0 & 0 & -1 & -2 & 0 & -1 & 1 & 1 \\ 1 & 0 & 0 & -1 & -2 & 0 & -1 & -1 \\ 1 & 1 & 0 & 0 & -1 & -2 & 0 & -1 \\ -1 & 1 & 1 & 0 & 0 & -1 & -2 & 0 \\ 0 & -1 & 1 & 1 & 0 & 0 & -1 & -2 \\ -2 & 0 & -1 & 1 & 1 & 0 & 0 & -1 \\ -1 & -2 & 0 & -1 & 1 & 1 & 0 & 0 \\ 0 & -1 & -2 & 0 & -1 & 1 & 1 & 0 \end{bmatrix}$$

and the $V(k)$ coefficients have the values

$$(v_0, v_1, v_2, v_3, v_4, v_5, v_6, v_7) = (1,\ 1,\ 1,\ -1,\ 0,\ -1,\ 0,\ -1).$$

In the next step by using (11) we obtain the matrix Z

$$Z = \begin{bmatrix} 0 & 0 & -1 & 2 & 0 & 1 & 0 & -1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & -1 \\ 1 & 1 & 0 & 0 & 0 & 2 & 0 & 1 \\ -1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & -1 & 1 & -1 & 0 & 0 & 0 & 2 \\ -2 & 0 & -1 & -1 & 0 & 0 & 0 & 1 \\ -1 & -2 & 0 & 1 & 0 & -1 & 0 & 0 \\ 0 & -1 & -2 & 0 & 0 & -1 & 0 & 0 \end{bmatrix}$$

Then making use of (9) we obtain the coefficients of $Q(k)$

$$Q = (q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7) =$$
$$= (-12,\ 32,\ 75,\ 34,\ 49,\ -78,\ -14,\ -32).$$

Finally by (8a) and (8b), we get the real and imaginary parts of the product

$$Q_{re} = q_0 + q_1/2 - q_3/2 - q_4 - q_5/2 + q_7/2 = -39,$$

and

$$Q_{im} = q_1/2 + q_2 + q_3/2 - q_5/2 - q_6 - q_7/2 = 177.$$

5. Analysis of hardware amount and delay

The multiplier in the proposed configuration consists of four main blocks. The first block (CC) computes the w_i and v_i coefficients, $i = 0,1,2,...,7$, the second block (WV) performs the multiplications $w_i v_j$, $i, j = 0,1,2,...,7$, the third block (Q) calculates the q_i coefficients, $i = 0,1,2,...,7$ and finally the fourth one (PR) determines Q_{re} and Q_{im}. The hardware amount of the 4-bit multiplier can be expressed in the following manner:

$$A_{MULT} = A_{CC} + A_{WV} + A_U + A_{PR}. \tag{14}$$

The area of CC block, A_{CC} is neglected as its area is very small. The *WV* block requires 4 2×2 multipliers (7.98 *GE*) , 16 - 2×1 multipliers(2.66 *GE*) and 16 - 1×1 multipliers (1.33 *GE*) , and 22 . 1×1 signed multipliers (4.33 *GE*) and 6 2×1 signed multipliers(5.66 GE). In total, the *WV* block uses 224.88 *GE* which corresponds to 24.98 A_{FA} (with A_{FA}=9*GE* [7]). The adders that compute q_i, *i*=1,2,..7, require approximately 1120 A_{FA} and the two adders for the final calculation of Q_{re} and Q_{im}, call for 78 A_{FA}. Hence,we finally obtain about 222.98 A_{FA}.

The 4×4 complex multiplier delay at the logical level can here be estimated in the following manner:

$$t_{MULT} = t_{ACC} + t_{WV} + t_Q + t_{PR} \tag{15}$$

where $t_{CC} \cong 0.7 t_{FA}$, $t_{WV} = t_{INV} + 2\cdot t_{NAND3} \cong 0.7 t_{FA}$, $t_{PR} = 8 t_{FA}$, $t_q = \max(t_{q_i}) = 10\, t_{FA}$.

Finally we obtain $t_{MULT} = 19.4\ t_{FA}$.

6. Conclusions

The design procedure of a complex multiplier based on Skavantzos and Stouraitis decomposition algorithm is presented. For larger complex multipliers this algorithm represents a decomposition tool, *i.e.* instead of 4 $n\times n$-bit binary multipliers, 64 $n/4\times n/4$-bit, $n/4+1\times n/4$ or $n/4+1\times n/4+1$-bit multipliers can be applied. The obtained results indicate that the hardware structure may be more regular than in the case of direct decomposition of the large multiplier. This is due to the realization of addition because after computation of partial multiplication results, irrespective of the multiplier size, 8 multi-operand parallel additions and next two 6-operand additions in parallel have to be carried out. The multiplier implementation with the use of the presented procedure becomes more effective with the growth of *n*, since the part connected with the realization of partial multiplications becomes dominant. The algorithm is not suitable for small *n*, because of relatively large number of additions.

7. References

[1] R. E. Blahut: Fast Algorithms for Digital Signal Processing, Addison-Wesley, 1987.

[2] S. Winograd :On the time required to perform multiplication, Journal of the ACM, vol 14, pp. 793-802.

[3] R.P. Brent, H.T. Kung: The chip complexity of binary arithmetic, Proceedings of the 12th Annual ACM Symposium on the Theory of Computers, 1980.

[4] A.Skavantzos, T. Stouraitis: Decomposition of Complex Multipliers Using Polynomial Encoding, IEEE Transactions on Computers, Vol. 41, No. 10, October 1992.

[5] K:Hwang.: Computer Arithmetic, Wiley, 1979.

[6] Smyk R., Czyżak M., Ulman Z. : Complex multiplier based on polynomial residue number system, IC-SPETO 2005, XXVIII Międzynarodowa Konferencja z Podstaw Elektrotechniki i Teorii Obwodów, Gliwice-Ustroń, 11-14.05. 2005, s. 215-219.

[7] Samsung Electronics: Standard Cell Logic Library STDL 130, 2005.

Artykul recenzowany