UNIWERSYTET TECHNOLOGICZNO-PRZYRODNICZY
IM. JANA I JĘDRZEJA ŚNIADECKICH
W BYDGOSZCZY

WYDZIAŁ TELEKOMUNIKACJI
I ELEKTROTECHNIKI

ZESZYTY NAUKOWE NR 253

TELEKOMUNIKACJA I ELEKTRONIKA 12

BYDGOSZCZ – 2009

UNIWERSYTET TECHNOLOGICZNO-PRZYRODNICZY
IM. JANA I JĘDRZEJA ŚNIADECKICH
W BYDGOSZCZY

ZESZYTY NAUKOWE NR 253

TELEKOMUNIKACJA I ELEKTRONIKA 12

BYDGOSZCZ – 2009

REDAKTOR NACZELNY
prof. dr hab. inż. Janusz Prusiński

REDAKTOR DZIALOWY
dr inż. Sławomir Cieślik

OPRACOWANIE TECHNICZNE
mgr inż. Daniel Morzyński

ISSN 1899-0088

Wydawnictwa Uczelniane Uniwersytetu Technologiczno-Przyrodniczego
ul. Ks. A. Kordeckiego 20, 85-225 Bydgoszcz, tel. (052) 3749482, 3749426
e-mail: wydawucz@utp.edu.pl http://www.utp.edu.pl/~wyd

Wyd. I. Nakład 80 egz. Ark. aut. 3,5. Ark. druk. 4,5.
Zakład Małej Poligrafii UTP Bydgoszcz, ul. Ks. A. Kordeckiego 20

Contents

UNIWERSYTET TECHNOLOGICZNO-PRZYRODNICZY
IM. JANA I JĘDRZEJA ŚNIADECKICH W BYDGOSZCZY
ZESZYTY NAUKOWE NR 253
TELEKOMUNIKACJA I ELEKTRONIKA 12 (2009), 5-20

ON INFLUENCE OF FEEDBACK ON HARMONICS IN MILDLY NONLINEAR ANALOG CIRCUITS

Andrzej Borys

Institute of Telecommunications, Faculty of Telecommunications and Electrical Engineering
University of Technology and Life Sciences (UTP)
ul. Kaliskiego 7, 85-789 Bydgoszcz, Poland

Summary: First, it is shown that applying the linear feedback increases the order of nonlinearity of the whole mildly nonlinear amplifier in comparison with that characterizing the amplifier without this feedback. Second, the impact of this fact on harmonics, which appear in the feedback amplifier driven by a single complex harmonic signal, is analyzed in detail. Finally, the associated model of the feedback amplifier is derived. It is shown that using this model, and only then, it is possible to interpret correctly the means of harmonic distortion calculations in weakly nonlinear amplifiers proposed by Palumbo and Pennisi in one of their recent papers.

Keywords: harmonic distortion analysis, mildly nonlinear circuits and systems, feedback, influence of feedback on harmonics, Volterra series

1. INTRODUCTION

The fact that feedback reduces nonlinear distortion is always pointed out in papers on mildly nonlinear circuits or systems, of which structure contains such a (linear) feedback. More precisely, and restricting ourselves here to consideration of only nonautonomous analog circuits like, for example, weakly nonlinear amplifiers, this is so expressed: the second and third order harmonic distortion factors or the second and third order intermodulation distortion factors of these amplifiers are considerably reduced by applying the linear negative feedback in their structures, in comparison to the structures without feedback. The amount of reduction of the harmonic or intermodulation distortion factors can be calculated from expressions, which can be found in the literature on these topics. The feedback return ratio and the feedback return difference, well known quantities in the linear feedback theory [1], play a prominent role in them.

With regard to the context sketched shortly above, the pioneering work done by Narayanan [2] in this area should be mentioned. The results of his investigations were published in 1970. Since then numerous papers, and even books, devoted to the analysis and reducing nonlinear distortion in weakly nonlinear analog circuits and systems, appeared. Because of their huge number, we will not list all of them here. Amongst them, we mention only the recent one. It is a paper by Palumbo and Pennisi [3] on the analysis of high-frequency harmonic distortion in weakly nonlinear feedback amplifi-

ers. We refer here, in a course of presenting new results, to the method, some expressions, and results published in [3] (of course, we could do this also with regard to others, as for example, to those given by Narayanan [2]).

In all the papers mentioned above, only the advantageous effect of feedback is emphasized. That is the fact that it reduces nonlinear distortion in a circuit. Two disadvantageous effects of introducing feedback (which are of general nature and occur also when the negative type of feedback is applied) were not perceived at all. These are the following:

1. The order of nonlinearity of the whole circuit, that is of the weakly nonlinear circuit to which a linear feedback was applied, increases.
2. In consequence, new harmonics appear in the circuit containing feedback in comparison with this circuit without feedback when they are driven by a sinusoidal signal.

One of the main objectives of this paper is to explain in detail how and why the order of nonlinearity of a weakly nonlinear circuit increases after applying to it a linear feedback, and what is the mechanism of appearance of additional harmonics in the latter circuit.

2. INCREASE OF ORDER OF NONLINEARITY BY INTRODUCING LINEAR FEEDBACK

Consider a weakly nonlinear amplifier as shown in Fig. 1(a). Let its input-output characteristic be described by a nonlinear operator H.

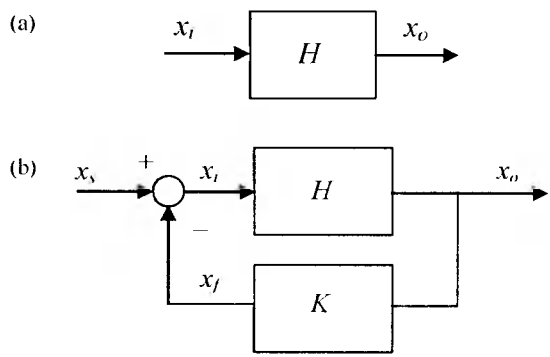


Fig. 1. Weakly nonlinear amplifier: (a) in configuration without feedback. (b) in closed-loop configuration containing linear feedback

Rys. 1. Nieliniowy wzmacniacz (z małymi nieliniowościami): (a) bez sprzężenia zwrotnego. (b) w pętli z liniowym sprzężeniem zwrotnym

When a linear feedback represented by a block K is applied to this amplifier, we get a closed-loop configuration illustrated in Fig. 1(b).

In Fig. 1(a), the input and output signals x_I and x_O, respectively, of a continuous time t are related to each other through the operator H as

$$x_O(t) = H\left(x_I(t)\right) \quad \text{or shortly} \quad x_O = H\left(x_I\right). \tag{1}$$

In the feedback configuration of Fig. 1(b), we have two additional equations

$$x_i = x_s - x_f \tag{2a}$$

and

$$x_f = K(x_o) \tag{2b}$$

where now the input signal to the whole amplifier is denoted by x_s. Moreover, x_f means the feedback return signal supplied to the input of amplifier H by the linear feedback block K. The input-output characteristic of this block is described by a linear operator K.

Substituting (2a) and (2b) into (1) gives

$$x_o = H(x_s - K(x_o)). \tag{3}$$

Equation (3) is obviously an implicit form of the input-output characteristic of the whole (feedback) amplifier. The task is to find its explicit form that is an operator H_f defined as

$$x_o = H_f(x_s). \tag{4}$$

Now, to understand better the problem of increase of the order of nonlinearity in the closed-loop, we consider first a weakly nonlinear amplifier in Fig. 1(b) of which both the components: the open-loop amplifier H and the linear feedback block K are frequency independent. Furthermore, assume that the amplifier H is described exactly by a third order polynomial. That is its description (exact) is in the form

$$x_o = H(x_i) = a_1 x_i + a_2 x_i^2 + a_3 x_i^3 \tag{5}$$

where the coefficients a_1, a_2, and a_3 are real numbers. This means that the nonlinearity of the open-loop amplifier is of the third order.

Note that the above assumptions allow us to assume that the behavior of the operator H_f can be also described in form of a polynomial. However, we cannot assume a'priori that it will be a polynomial of the third degree (order), too. We must assume, generally, an infinite power series what will be evident from the course of further derivations. That is

$$x_o = H_f(x_s) = a_{1f} x_s + a_{2f} x_s^2 + a_{3f} x_s^3 + a_{4f} x_s^4 + a_{5f} x_s^5 + \ldots \tag{6}$$

where the coefficients a_{1f}, a_{2f}, a_{3f}, a_{4f}, a_{5f}, and so on are real numbers with the letter "f" in their subscripts standing for "feedback". Moreover, rewrite (2b) for a frequency independent linear feedback as

$$x_f = k \cdot x_o \tag{7}$$

where k is a real number.

Using the description (5) for H and substituting (6) and (7) into (3), we obtain

$$
\begin{aligned}
&a_{1f}x_s + a_{2f}x_s^2 + a_{3f}x_s^3 + a_{4f}x_s^4 + a_{5f}x_s^5 + .. = \\
&= a_1\left[x_s - k\left(a_{1f}x_s + a_{2f}x_s^2 + a_{3f}x_s^3 + a_{4f}x_s^4 + a_{5f}x_s^5 + ..\right)\right] + \\
&+ a_2\left[x_s - k\left(a_{1f}x_s + a_{2f}x_s^2 + a_{3f}x_s^3 + a_{4f}x_s^4 + a_{5f}x_s^5 + ..\right)\right]^2 + \\
&+ a_3\left[x_s - k\left(a_{1f}x_s + a_{2f}x_s^2 + a_{3f}x_s^3 + a_{4f}x_s^4 + a_{5f}x_s^5 + ..\right)\right]^3 .
\end{aligned}
\tag{8}
$$

Then, equating to each other the expressions of the same order (degree) – that is such which contain the same power of the variable x_s – on both sides of (8), we get successively

$$a_{1f}x_s = a_1\left(x_s - ka_{1f}x_s\right) \tag{9a}$$

$$a_{2f}x_s^2 = -a_1 k a_{2f}x_s^2 + a_2\left(x_s - ka_{1f}x_s\right)^2 \tag{9b}$$

$$a_{3f}x_s^3 = -a_1 k a_{3f}x_s^3 - 2a_2\left(x_s - ka_{1f}x_s\right)ka_{2f}x_s^2 + a_3\left(x_s - ka_{1f}x_s\right)^3 \tag{9c}$$

$$
\begin{aligned}
a_{4f}x_s^4 = &-a_1 k a_{4f}x_s^4 - 2a_2\left(x_s - ka_{1f}x_s\right)ka_{3f}x_s^3 + \\
&+ a_2\left(ka_{2f}x_s^2\right)^2 - 3a_3\left(x_s - ka_{1f}x_s\right)^2 ka_{2f}x_s^2
\end{aligned}
\tag{9d}
$$

$$
\begin{aligned}
a_{5f}x_s^5 = &-a_1 k a_{5f}x_s^5 - 2a_2\left(x_s - ka_{1f}x_s\right)ka_{4f}x_s^4 + 2a_2k^2a_{2f}a_{3f}x_s^5 + \\
&+ 3a_3\left(x_s - ka_{1f}x_s\right)\left(ka_{2f}x_s^2\right)^2 - 3a_3\left(x_s - ka_{1f}x_s\right)^2 ka_{3f}x_s^3
\end{aligned}
\tag{9e}
$$

and so on. From these equations, after eliminating x_s and performing a number of algebraic manipulations, we obtain

$$a_{1f} = \frac{a_1}{1 + ka_1} \tag{10a}$$

$$a_{2f} = \frac{a_2}{\left(1 + ka_1\right)^3} \tag{10b}$$

$$a_{3f} = \frac{a_3}{\left(1 + ka_1\right)^4} - \frac{2a_2{}^2k}{\left(1 + ka_1\right)^5} \tag{10c}$$

$$a_{4f} = \frac{5a_2{}^3k^2}{\left(1 + ka_1\right)^7} - \frac{a_2a_3k\left(5 + 3ka_1\right)}{\left(1 + ka_1\right)^6} \tag{10d}$$

$$a_{5f} = -\frac{14a_2{}^4k^3}{\left(1 + ka_1\right)^9} + \frac{a_2{}^2a_3k^2\left(21 + 6ka_1\right)}{\left(1 + ka_1\right)^8} - \frac{3a_3{}^2k}{\left(1 + ka_1\right)^7} \tag{10e}$$

and so on.

Note from (10d) and (10e) that the coefficients a_{4f} and a_{5f} are, generally, non-zero. The same regards also the further coefficients a_{6f}, a_{7f}, and so on, in the expansion (6). So really the degree (order) of the polynomial describing an amplifier after

adding to it a linear feedback is different from that describing the same amplifier but without feedback. It changed here from three in (5) to infinity in (6).

So the conclusion is that the linear feedback has an influence upon the order of nonlinearity that is incorporated in the amplifier's closed-loop description. Furthermore, observe that if the amplifier's open-loop description as (5) would be in form of an infinite power series we did not see the above effect. Probably of this reason, this effect was not perceived before.

At this point, we pay also the reader's attention to the fact that the radius of convergence of the (finite) power series (5) is infinite, and it does not mean that the same holds for the (infinite) power series (6), too. Probably, the latter has a finite radius of convergence and it must be found. From the form of expressions (10a-e), we see that it is not a simple task.

A more general case of a weakly nonlinear amplifier in Fig. 1(b), of which both the components: the open-loop amplifier H and the linear feedback block K are frequency dependent, is also, as we will see, more cumbersome. Assume then that the amplifier H is described (in the time domain) by a Volterra series [4, 5] consisting of only first three components. That is its description (exact) will be, analogously to (5), in the following form

$$x_o(t)=H\left(x_i(t)\right)=x_o^{(1)}(t)+x_o^{(2)}(t)+x_o^{(3)}(t)=\int_{-\infty}^{\infty}h^{(1)}(\tau)x_i(t-\tau)d\tau++\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}h^{(2)}(\tau_1,\tau_2)\cdot$$
$$\cdot x_i(t-\tau_1)x_i(t-\tau_2)d\tau_1 d\tau_2+\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}h^{(3)}(\tau_1,\tau_2,\tau_3)x_i(t-\tau_1)x_i(t-\tau_2)x_i(t-\tau_3)d\tau_1 d\tau_2 d\tau_3\ . \tag{11}$$

The terms $x_o^{(1)}(t)$, $x_o^{(2)}(t)$, and $x_o^{(3)}(t)$ in (11), which are the respective components of $x_o(t)$, we call the amplifier partial responses of the first, second, and third order, respectively. This order is with respect to a variable representing the signal (here, in (11), x_i). Furthermore, by $h^{(1)}(t)$, $h^{(2)}(t_1,t_2)$, and $h^{(2)}(t_1,t_2,t_3)$ we denote, respectively, the first (linear), second, and third order, nonlinear impulse responses of the amplifier without feedback.

The equivalent of (6) will be now an infinite Volterra series as

$$\begin{aligned}
x_o(t)&=H_f\left(x_s(t)\right)=x_o^{(1)}(t)+x_o^{(2)}(t)+x_o^{(3)}(t)+x_o^{(4)}(t)+x_o^{(5)}(t)+\ldots=\\
&=\int_{-\infty}^{\infty}h_f^{(1)}(\tau)x_s(t-\tau)d\tau+\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}h_f^{(2)}(\tau_1,\tau_2)x_s(t-\tau_1)x_s(t-\tau_2)d\tau_1 d\tau_2+\\
&+\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}h_f^{(3)}(\tau_1,\tau_2,\tau_3)\left(\prod_{k=1}^{3}x_s(t-\tau_k)d\tau_k\right)+\\
&+\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}h_f^{(4)}(\tau_1,\tau_2,\tau_3,\tau_4)\left(\prod_{k=1}^{4}x_s(t-\tau_k)d\tau_k\right)+\\
&+\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}h_f^{(5)}(\tau_1,\tau_2,\tau_3,\tau_4,\tau_5)\left(\prod_{k=1}^{5}x_s(t-\tau_k)d\tau_k\right)+\ldots\ .
\end{aligned} \tag{12}$$

And the equivalent of (7) will be the linear convolution having the form

$$x_f(t) = K(x_o(t)) = \int_{-\infty}^{\infty} k(\tau) x_o(t-\tau)\, d\tau \,. \tag{13}$$

In (12) and (13), the letter "f" in subscripts stands, as before, for "feedback". Furthermore, $h_f^{(1)}(t)$, $h_f^{(2)}(t_1,t_2)$, $h_f^{(3)}(t_1,t_2,t_3)$, $h_f^{(4)}(t_1,t_2,t_3,t_4)$, $h_f^{(5)}(t_1,t_2,t_3,t_4,t_5)$, and so on are, respectively, the first (linear), second, third, fourth, fifth order, and next orders, nonlinear impulse responses of the feedback amplifier. Moreover, the function $k(t)$ in (13) is the (linear) impulse response of the now frequency dependent block K in Fig. 1(b).

Substituting x_o and x_f given by (12) and (13), respectively, into (3), and using also in (3) the formula (11) for the operator H, we get an equivalent of (8) for this far more complicated case with the amplifier and feedback in Fig. 1(b) being frequency dependent. In the next step, we proceed with the resulting equation similarly as before. That is we equate to each other expressions of the same order (degree) occurring on its both sides. More precisely, we equate expressions in which the number of appearances of the variable x_s is the same. As a result, we get the equivalent of equations (9a-e).

Further procedure leading to getting the equivalents of (10a-e) is quite involved because of occurrence of a huge number of multidimensional integrals in equivalents of (9a-e). In such situations, the method usually used, which is dated back to the appearance of the work [2], is carrying out the transformation of equations to the frequency domain by the use of the multidimensional Fourier (or Laplace) transforms. In consequence, one obtains the nonlinear transfer functions of a mildly nonlinear circuit. In the case of calculation of nonlinear transfer functions of the fourth or fifth order, and of next orders, for a circuit of Fig. 1(b), it is needed to carry out a huge number of algebraic manipulations to get the final results. The higher the order, the larger is the number of manipulations.

We will not do this here (details of these calculations will be eventually published later). The important for us here is one result that follows from these calculations. That is, similarly as in the case of (10a-e), we get (generally) nonzero nonlinear transfer functions of the fourth, fifth, and of higher orders for the circuit in Fig. 1(b). And finally, it follows from the latter that the nonlinear impulse responses related with them (through the inverse transforms) are nonzero functions, too.

So the conclusions, which can be drawn from the above, are similar to those presented a while before for the structure of Fig. 1(b) considered to be purely frequency independent. First, the Volterra series describing a mildly nonlinear amplifier after adding to it a linear feedback (with memory) has a different number of components from that describing the same amplifier but without feedback. This number changed here from three in (11) to infinity in (12).

Second, which is related to the previous conclusion, we observe that the linear feedback (with memory) has an influence upon the order of nonlinearity that is represented in the amplifier's closed-loop description by a Volterra series through the highest partial response in it. Furthermore, observe that if the amplifier's open-loop description as (11) would be in form of a Volterra series containing the infinite number of

components, we did not see the above effect. Probably of this reason, this effect was not reported in the literature up to now.

3. FEEDBACK AND POWER SERIES-LIKE MODEL OF WEAKLY NONLINEAR AMPLIFIER WITH MEMORY

In [3], Palumbo and Pennisi have developed a power series-like model for weakly nonlinear amplifiers with memory in configuration (without feedback) of Fig. 1(a). It has the following form

$$x_o = a_1(j\omega)x_i + a_2(j\omega)x_i^2 + a_3(j\omega)x_i^3 \tag{14}$$

in the notation of [3]. In (14), $\omega = 2\pi f$ means the angular frequency with f denoting the usual frequency variable. Moreover, $j = \sqrt{-1}$ and the frequency-dependent coefficients $a_1(j\omega)$, $a_2(j\omega)$, and $a_3(j\omega)$ are the coefficients in this (truncated) power series-like description of a mildly nonlinear amplifier with memory.

One thing is very important of which we must be aware when applying the model given by (14). This is the fact that it is valid only for input signals of the form

$$x_i(t) = A_i \exp(j\omega t) \tag{15}$$

where A_i is a real number and means the amplitude of this complex harmonic signal. In other words, (14) does not represent a Volterra series valid for any input signal, but only for such that having the form $x_i(t) = A_i \exp(j\omega t)$.

The above is evident from the derivation of (14) which can be done by substituting $x_i(t)$ given by (15) into the finite Volterra series consisting of only first three components (such as that in (11)) and performing next the needed operations in it, as shown in the following

$$x_o(t) = \int_{-\infty}^{\infty} h^{(1)}(\tau)A_i \exp(j\omega(t-\tau))d\tau + \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h^{(2)}(\tau_1,\tau_2)A_i^2 \exp(j\omega(t-\tau_1))\cdot$$

$$\cdot\exp(j\omega(t-\tau_2))d\tau_1 d\tau_2 + \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h^{(3)}(\tau_1,\tau_2,\tau_3)A_i^3 \exp(j\omega(t-\tau_1))\exp(j\omega(t-\tau_2))\cdot$$

$$\cdot\exp(j\omega(t-\tau_3))d\tau_1 d\tau_2 d\tau_3 = A_i \exp(j\omega t)\int_{-\infty}^{\infty} h^{(1)}(\tau)\exp(-j\omega\tau))d\tau + A_i^2 \exp(j2\omega t)\cdot \tag{16}$$

$$\cdot\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h^{(2)}(\tau_1,\tau_2)\exp(-j\omega(\tau_1+\tau_2))d\tau_1 d\tau_2 + A_i^3 \exp(j3\omega t)\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h^{(3)}(\tau_1,\tau_2,\tau_3)\cdot$$

$$\cdot\exp(-j3\omega(\tau_1+\tau_2+\tau_3))d\tau_1 d\tau_2 d\tau_3 .$$

Applying in (16) the notion of n-dimensional Fourier transforms [5], here for n = 1, 2 or 3, and choosing the same frequency point f at each dimension, we arrive finally at

$$x_o(t) = H^{(1)}(f) A_i \exp(j2\pi ft) + \\ + H^{(2)}(f,f) A_i^2 \exp(j2\pi 2ft) + \\ + H^{(2)}(f,f,f) A_i^3 \exp(j2\pi 3ft) \tag{17}$$

where $H^{(1)}(f)$, $H^{(2)}(f,f)$, and $H^{(3)}(f,f,f)$ denote the Fourier transforms (mentioned above) of $h^{(1)}(t)$, $h^{(2)}(t_1,t_2)$, and $h^{(3)}(t_1,t_2,t_3)$, respectively, at the same frequency point f chosen at each dimension. At this point, note also that $H^{(1)}(f)$, $H^{(2)}(f,f)$, and $H^{(3)}(f,f,f)$ are called the circuit nonlinear transfer functions of the first (linear one), of the second, and of the third order, accordingly. Furthermore, $f = \omega/(2\pi)$ in (17).

Observe that (17) is identical with (14) when we identify $a_1(j\omega)$ with $H^{(1)}(f)$, $a_2(j\omega)$ with $H^{(2)}(f,f)$, and $a_3(j\omega)$ with $H^{(3)}(f,f,f)$.

In this paper, we call shortly the signal of the form given by (15) a harmonic at frequency ω (or equivalently f). From (17), we see that an amplifier in the structure of Fig. 1(a), having the description given by (11), and driven by a single harmonic at frequency f has three (and only three) harmonics, at frequencies f, $2f$, and $3f$, at its output. Observe that this number is identical with the number of components in the finite Volterra series (11). Moreover, the number three staying at $3f$ (at the highest frequency of these harmonics) is equal to the order of nonlinearity incorporated in the amplifier description (identical with the highest order among partial responses in the Volterra series).

Note now that we have a quite different situation when the linear feedback (with memory) as in Fig. 1(b) is applied to the amplifier characterized above. Then, we must use for the whole amplifier, as shown in the previous section, an infinite Volterra series of the form (12). And it is clear from the derivation underlying (17) that, in this case, after substituting into (12) the input signal denoted now as

$$x_s(t) = A_s \exp(j\omega t) \tag{18}$$

where A_s is a real number and means the amplitude of this harmonic signal, and performing afterwards the needed operations in it, we get finally an equivalent of (17) as

$$x_o(t) = H_f^{(1)}(f) A_s \exp(j2\pi ft) + H_f^{(2)}(f,f) A_s^2 \exp(j2\pi 2ft) + \\ + H_f^{(3)}(f,f,f) A_s^3 \exp(j2\pi 3ft) + H_f^{(4)}(f,f,f,f) A_s^4 \exp(j2\pi 4ft) + \\ + H_f^{(5)}(f,f,f,f,f) A_s^5 \exp(j2\pi 5ft) + \ldots . \tag{19}$$

In (19), similarly as before, $H_f^{(1)}(f)$, $H_f^{(2)}(f,f)$, $H_f^{(3)}(f,f,f)$, $H_f^{(4)}(f,f,f,f)$, $H_f^{(5)}(f,f,f,f,f)$, and so on are, respectively, the Fourier transforms of $h_f^{(1)}(t)$, $h_f^{(2)}(t_1,t_2)$, $h_f^{(3)}(t_1,t_2,t_3)$, $h_f^{(4)}(t_1,t_2,t_3,t_4)$, $h_f^{(5)}(t_1,t_2,t_3,t_4,t_5)$, and of the next nonlinear impulse responses. They are calculated here at the same frequency point f chosen at each dimension. Furthermore, these functions are called the nonlinear transfer functions of the corresponding orders of the feedback amplifier.

Observe from (19) that, opposite to the previous case, we have at the output of the same amplifier, which is put now into the feedback structure of Fig. 1(b), an infinite number of harmonics, having frequencies f, $2f$, $3f$, $4f$, $5f$, and so on. Note further that this infinite number of harmonics corresponds with the infinite number of components in the Volterra series (12). This (infinite) number is equal to the order of nonlinearity incorporated in the description (12) (i.e. identical with the highest order of the partial response in the Volterra series which is here infinite).

4. RESCUE FOR PALUMBO AND PENNISI'S MEANS OF MODELING WEAKLY NONLINEAR FEEDBACK AMPLIFIERS

It follows evidently from the results of the previous two sections that the power series-like model for weakly nonlinear amplifiers expressed by (14) cannot be applied to study the feedback structure of Fig. 1(b). More precisely:

1. It is not a proper model for the whole circuit with feedback as represented by Fig. 1(b). Then, as we have shown, expression (19) must be used instead of (17) (which is identical with (14)).
2. It is not also a proper model for modeling the input-output behavior of the weakly nonlinear amplifier H in Fig. 1(b) because the input signal at its input is now a sum of an infinite number of harmonics (not a single harmonic of the form $x_f(t) = A_f \exp(j2\pi ft)$).

Nevertheless, it has been used in the above context in an approach developed by Palumbo and Pennisi in [3] for calculation of harmonic distortion in weakly nonlinear feedback amplifiers. See equations: (15) in [3] and (10) in [6] – with regard to point 1, and equations: (17) in [3] and (A3) in [6] – with regard to point 2, for example.

We derive here a model that enables to obtain correctly the results presented by Palumbo and Pennisi in [3]. It is an associated model incorporating some simplifications with regard to the original formulation. So, for that reason, it can lead to results that are not necessarily identical with those one gets with the use of the original model. And, concluding, the approach from [3] for feedback amplifier should be perceived similarly.

To achieve our goal mentioned above, we proceed now in the following way: First, we postulate the form as shown in Fig. 2 for an associated model we look for. Then, we check whether it really describes correctly the results presented by Palumbo and Pennisi in [3].

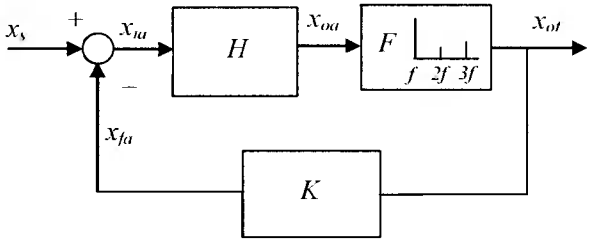


Fig. 2. Associated model of weakly nonlinear amplifier in closed-loop configuration containing linear feedback

Rys. 2. Stowarzyszony model nieliniowego wzmacniacza w pętli z liniowym sprzężeniem zwrotnym

In Fig. 2, an ideal filter F plays a role of a filter that allows only the harmonics at frequencies f, $2f$, and $3f$ at the output of H to pass to the output of the whole amplifier. It fully rejects all the other harmonics. In consequence, under the assumption of the input signal to the whole amplifier x_s being given by (18), we get such a situation in the loop of Fig. 2 that the signals x_{ia}, x_{of}, and x_{fa} contain exclusively harmonics of the frequencies f, $2f$, and $3f$. Other harmonics occur in the circuit of Fig. 2 only at the output of the amplifier H. That is they are components of the signal x_{oa} (besides the harmonics at frequencies f, $2f$, and $3f$).

It follows from the above and the results of previous sections that the signals x_{ia}, x_{oa}, and x_{fa} in the associated model of Fig. 2 are not identical with the corresponding signals x_i, x_o, and x_f of the original model of Fig. 1(b). For that reason, we use an additional letter "a" in their subscript notation. Of course, this additional letter should be also put at the subscript of the symbol of filter F output signal occurring in Fig. 2. However, that more consistent notation x_{ofa} has been abbreviated in this case to x_{of}, to avoid too long subscripts, and in such a form is used in the paper.

Observe now that the signal x_{ia}, as containing three harmonics, can be expressed in the following way

$$x_{ia}(t) = A_{1i}\exp(j2\pi ft) + A_{2i}\exp(j2\pi 2ft) + A_{3i}\exp(j2\pi 3ft) \tag{20}$$

where A_{1i}, A_{2i}, and A_{3i} are real numbers and mean the amplitudes of the corresponding harmonics at frequencies f, $2f$, and $3f$, respectively.

As we already know the form given by (14) for a power series-like model cannot be used to model the amplifier H in Fig. 1(b) or in Fig. 2. Remember that it is so because (14) is valid only for input signals being single harmonics. Extension of applicability of the power series-like model is however possible. We do this by finding each time its specific form, for a particular input signal. So, for H in the original model of Fig. 1(b), a counterpart of (14) must be derived assuming the input signal in form of an infinite sum of harmonics at frequencies f, $2f$, $3f$, $4f$, $5f$, and so on. Further, in the case of H in the associated model of Fig. 2, the situation is simpler in comparison with the latter. Then, we have to consider in the calculations the input signal consisting of only three harmonics (at frequencies f, $2f$, and $3f$) as in (20).

Now, we will derive a power series-like formula regarding the latter case. To this end, we use a specific form of the Volterra series that enables to express its components (partial responses) through the circuit nonlinear transfer functions – for more details see [5]). So applying it to the series (11), we get the following expressions for its components:

$$x_o^{(1)}(t) = \int_{-\infty}^{\infty} H^{(1)}(f_1) X_i(f_1) \exp(j2\pi f_1 t) df_1 \tag{21a}$$

$$x_o^{(2)}(t) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} H^{(2)}(f_1, f_2) X_i(f_1) X_i(f_2) \exp(j2\pi(f_1 t + f_2 t)) df_1 df_2 \tag{21b}$$

$$\begin{aligned} x_o^{(3)}(t) = &\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} H^{(3)}(f_1, f_2, f_3) X_i(f_1) X_i(f_2) \cdot \\ &\cdot X_i(f_3) \exp(j2\pi(f_1 t + f_2 t + f_3 t)) df_1 df_2 df_3 \end{aligned} \tag{21c}$$

where $X_i(f_x)$, $x = 1, 2, 3$, means the Fourier transform of the input signal, and the sets of frequency variables: $\{f_1\}$, $\{f_1, f_2\}$, and $\{f_1, f_2, f_3\}$ occurring in (21a), (21b), and (21c), respectively, form the corresponding one-, two-, and three-dimensional frequency spaces.

The Fourier transform of the input signal to the amplifier H in Fig. 2 is given by

$$X_{ia}(f_x) = A_{1i}\delta(f_x - f) + A_{2i}\delta(f_x - 2f) + A_{3i}\delta(f_x - 3f) \tag{22}$$

where δ means the Dirac impulse and f_x is the current frequency in the Fourier transform.

Specializing the general expressions (21a), (21b), and (21c) to the case of H in the structure of Fig. 2 means introduction in them $X_{ia}(f_x)$ instead of $X_i(f_x)$, and adding the letter "a" at the subscripts by $x_o^{(1)}(t)$, $x_o^{(2)}(t)$, and $x_o^{(3)}(t)$. Carrying out afterwards the standard algebraic manipulations, exploiting the sifting property of the Dirac impulse and using the symmetric nonlinear transfer functions in them, we get finally

$$\begin{aligned} x_{oa}^{(1)}(t) = &H^{(1)}(f) A_{1i} \exp(j2\pi ft) + H^{(1)}(2f) \cdot \\ &\cdot A_{2i} \exp(j2\pi 2ft) + H^{(1)}(3f) A_{3i} \exp(j2\pi 3ft) \end{aligned} \tag{23a}$$

$$\begin{aligned} x_{oa}^{(2)}(t) = &H^{(2)}(f, f) A_{1i}^2 \exp(j2\pi 2ft) + 2H^{(2)}(f, 2f) \cdot \\ &\cdot A_{1i} A_{2i} \exp(j2\pi 3ft) + \Big[2H^{(2)}(f, 3f) A_{1i} A_{3i} \exp(j2\pi 4ft) + \\ &+ H^{(2)}(2f, 2f) A_{2i}^2 \exp(j2\pi 4ft) \Big] + 2H^{(2)}(2f, 3f) A_{2i} A_{3i} \exp(j2\pi 5ft) + \\ &+ H^{(2)}(3f, 3f) A_{3i}^2 \exp(j2\pi 6ft) \end{aligned} \tag{23b}$$

$$x_{oa}^{(3)}(t) = H^{(3)}(f,f,f)A_{1i}^3 \exp(j2\pi 3ft) + 3H^{(3)}(f,f,2f)A_{1i}^2 A_{2i} \exp(j2\pi 4ft) +$$
$$+\Big[3H^{(3)}(f,f,3f)A_{1i}^2 A_{3i} \exp(j2\pi 5ft) + 3H^{(3)}(f,2f,2f)\cdot$$
$$\cdot A_{1i}A_{2i}^2 \exp(j2\pi 5ft)\Big] + \Big[6H^{(3)}(f,2f,3f)A_{1i}A_{2i}A_{3i} \exp(j2\pi 6ft) +$$
$$+H^{(3)}(2f,2f,2f)A_{2i}^3 \exp(j2\pi 6ft)\Big] + \Big[3H^{(3)}(f,3f,3f)A_{1i}A_{3i}^2 \exp(j2\pi 7ft) + \quad (23c)$$
$$+3H^{(3)}(2f,2f,3f)A_{2i}^2 A_{3i} \exp(j2\pi 7ft)\Big] + 3H^{(3)}(2f,3f,3f)\cdot$$
$$\cdot A_{2i}A_{3i}^2 \exp(j2\pi 8ft) + H^{(3)}(3f,3f,3f)A_{3i}^3 \exp(j2\pi 9ft)\,.$$

Denoting the components of $x_{ia}(t)$ in (20) as $x_{1i}(t) = A_{1i}\exp(j2\pi ft)$, $x_{2i}(t) = A_{2i}\exp(j2\pi 2ft)$, and $x_{3i}(t) = A_{3i}\exp(j2\pi 3ft)$, respectively, applying them afterwards in (23a), (23b), and (23c), and summing the partial responses, we obtain the following

$$x_{oa} = a_1(j\omega)x_{1i} + a_2(j\omega)x_{2i} + a_3(j\omega)x_{3i} + a_{11}(j\omega)x_{1i}^2 + 2a_{12}(j\omega)x_{1i}x_{2i}$$
$$+\Big[2a_{13}(j\omega)x_{1i}x_{3i} + a_{22}(j\omega)x_{2i}^2\Big] + 2a_{23}(j\omega)x_{2i}x_{3i} + a_{33}(j\omega)x_{3i}^2 + \quad (24)$$
$$+a_{111}(j\omega)x_{1i}^3 + 3a_{112}(j\omega)x_{1i}^2x_{2i} + \ldots\ldots + a_{333}(j\omega)x_{1i}^3$$

where the expressions describing the coefficients in the resulting multivariate polynomial can be easily determined by comparison of (24) with (23a), (23b) or (23c). So we have $a_1(j\omega) = H^{(1)}(f)$, $a_2(j\omega) = H^{(1)}(2f)$, $a_3(j\omega) = H^{(1)}(3f)$, $a_{11}(j\omega) = H^{(2)}(f,f)$, $a_{12}(j\omega) = 2H^{(2)}(f,2f)$, $a_{13}(j\omega) = 2H^{(2)}(f,3f)$, ... , $a_{33}(j\omega) = H^{(2)}(3f,3f)$, $a_{111}(j\omega) = H^{(3)}(f,f,f)$, $a_{112}(j\omega) = 3H^{(3)}(f,f,2f)$, ... , $a_{333}(j\omega) = H^{(3)}(3f,3f,3f)$. (By the way, note that the principle of indexing the coefficients of the multivariate polynomial in (24) is a little bit different than that used in (14).)

Expression (24) is a direct counterpart of (14) for correct modeling the input-output behavior of the amplifier H in the associated model of Fig. 2. We see however that it is much more complicated than (14), and thereby very heavy to use. So we resign from this form in further derivations. In what follows, we prefer to use the other form of the power series-like model that is summarized in equations (23a-c).

Note now that for the structure of Fig. 2 we can write the following two equations (in the operator form)

$$x_{fa} = x_s - x_{ia} \quad (25a)$$

and

$$x_{fa} = KFHx_{ia} \quad (25b)$$

where F stands for the mapping which is carried out by the ideal filter F of Fig. 2, according to the rule described beneath this figure.

Using (25a) in (25b), we get

$$x_s - x_{ia} = KFHx_{ia} = KFx_{oa} \ . \tag{26}$$

Further, knowing that the signal $x_{oa}(t) = (Hx_{ia})(t)$ at the output of the amplifier H in Fig. 2 is equal to the sum $x_{oa}^{(1)}(t) + x_{oa}^{(2)}(t) + x_{oa}^{(3)}(t)$ with its components given by (23a-c), and applying to it the filtering rule of the filter F, we arrive for the signal $(Fx_{oa})(t)$ at

$$\begin{aligned}(Fx_{oa})(t) &= H^{(1)}(f)A_{1i}\exp(j2\pi ft) + H^{(1)}(2f)A_{2i}\exp(j2\pi 2ft) + \\ &+ H^{(1)}(3f)A_{3i}\exp(j2\pi 3ft) + H^{(2)}(f,f)A_{1i}^2\exp(j2\pi 2ft) + 2H^{(2)}(f,2f)\cdot \\ &\cdot A_{1i}A_{2i}\exp(j2\pi 3ft) + H^{(3)}(f,f,f)A_{1i}^3\exp(j2\pi 3ft)\ .\end{aligned} \tag{27}$$

The linear feedback block K (as a linear circuit with memory) transfers the signal (27) to its output according to the following formula

$$\begin{aligned}(KFx_{oa})(t) &= K(f)H^{(1)}(f)A_{1i}\exp(j2\pi ft) + K(2f)H^{(1)}(2f) \\ &\cdot A_{2i}\exp(j2\pi 2ft) + K(3f)H^{(1)}(3f)A_{3i}\exp(j2\pi 3ft) + K(2f)\cdot \\ &\cdot H^{(2)}(f,f)A_{1i}^2\exp(j2\pi 2ft) + 2K(3f)H^{(2)}(f,2f)A_{1i}A_{2i}\cdot \\ &\cdot\exp(j2\pi 3ft) + K(3f)H^{(3)}(f,f,f)A_{1i}^3\exp(j2\pi 3ft)\ .\end{aligned} \tag{28}$$

$K(f_x)$ on the right-hand side of (28) means the transfer function of the linear feedback block K, calculated at the corresponding frequencies $f_x = f$, $2f$ or $3f$.

Introducing (18), (20), and (28) into (26) gives

$$\begin{aligned}&A_s\exp(j2\pi ft) - A_{1i}\exp(j2\pi ft) - A_{2i}\exp(j2\pi 2ft) - A_{3i}\exp(j2\pi 3ft) = \\ &= K(f)H^{(1)}(f)A_{1i}\exp(j2\pi ft) + K(2f)H^{(1)}(2f)A_{2i}\exp(j2\pi 2ft) + K(3f)\cdot \\ &\cdot H^{(1)}(3f)A_{3i}\exp(j2\pi 3ft) + K(2f)H^{(2)}(f,f)\cdot A_{1i}^2\exp(j2\pi 2ft) + \\ &+ 2K(3f)H^{(2)}(f,2f)A_{1i}A_{2i}\exp(j2\pi 3ft) + K(3f)H^{(3)}(f,f,f)A_{1i}^3\exp(j2\pi 3ft)\ .\end{aligned} \tag{29}$$

The next step is to equate to each other the expressions staying by the exponents of the same frequency on both sides of (29). As a result, we get

$$A_s - A_{1i} = K(f)H^{(1)}(f)A_{1i} \tag{30a}$$

$$-A_{2i} = K(2f)H^{(1)}(2f)A_{2i} + K(2f)H^{(2)}(f,f)A_{1i}^2 \tag{30b}$$

$$\begin{aligned}-A_{3i} &= K(3f)H^{(1)}(3f)A_{3i} + 2K(3f)\cdot \\ &\cdot H^{(2)}(f,2f)A_{1i}A_{2i} + K(3f)H^{(3)}(f,f,f)A_{1i}^3\ .\end{aligned} \tag{30c}$$

Now, solving (30a) for A_{1i}, afterwards solving (30b) for A_{2i} and using A_{1i} from the previous step, and finally solving (30c) for A_{3i} and using A_{1i} and A_{2i} from the previous two steps, we obtain successively

$$A_{1i} = \frac{A_s}{1 + K(f) H^{(1)}(f)} \tag{31a}$$

$$A_{2i} = \frac{-K(2f) H^{(2)}(f,f) A_s^2}{\left[1 + K(2f) H^{(1)}(2f)\right]\left[1 + K(f) H^{(1)}(f)\right]^2} \tag{31b}$$

$$\begin{aligned} A_{3i} = & \frac{K(3f) A_s^3}{\left[1 + K(3f) H^{(1)}(3f)\right]\left[1 + K(f) H^{(1)}(f)\right]^2} \cdot \\ & \cdot \left\{ -H^{(3)}(f,f,f) + \frac{2H^{(2)}(f,f) K(2f) H^{(2)}(f,2f)}{\left[1 + K(2f) H^{(1)}(2f)\right]} \right\} . \end{aligned} \tag{31c}$$

Having the expressions determining the amplitudes A_{1i}, A_{2i}, and A_{3i} as the functions of the amplitude A_s, we can eliminate them from (27). This leads to

$$\begin{aligned} x_{of}(t) = (Fx_{oa})(t) = & a_{1f}(j2\pi f) A_s \exp(j2\pi ft) + \\ & + a_{2f}(j2\pi 2f) A_s^2 \exp(j2\pi 2ft) + a_{3f}(j2\pi 3f) A_s^3 \exp(j2\pi 3ft) \end{aligned} \tag{32}$$

with the functions $a_{1f}(j2\pi f)$, $a_{2f}(j2\pi f)$, and $a_{3f}(j2\pi f)$ given by

$$a_{1f}(j2\pi f) = \frac{H^{(1)}(f)}{1 + K(f) H^{(1)}(f)} \tag{33a}$$

$$a_{2f}(j2\pi f) = \frac{H^{(2)}(f,f)}{\left[1 + K(f) H^{(1)}(f)\right]^2 \left[1 + K(2f) H^{(1)}(2f)\right]} \tag{33b}$$

$$\begin{aligned} a_{3f}(j2\pi f) = & \frac{H^{(3)}(f,f,f)}{\left[1 + K(f) H^{(1)}(f)\right]^3 \left[1 + K(3f) H^{(1)}(3f)\right]} \cdot \\ & \cdot \left\{ 1 - \frac{2H^{(2)}(f,f) H^{(2)}(f,2f)}{H^{(1)}(2f) H^{(3)}(f,f,f)} \cdot \frac{K(2f) H^{(1)}(2f)}{\left[1 + K(2f) H^{(1)}(2f)\right]} \right\} . \end{aligned} \tag{33c}$$

Note now that introducing in (33a-c) the following notation for the nonlinear transfer functions: $H^{(1)}(f)=a_1(j2\pi f)$, $H^{(2)}(f,f)=a_2(j2\pi f)$, and $H^{(3)}(f,f,f)=$ $=a_3(j2\pi f)$ we get the expressions identical with those given in [3] for the so-called *closed-loop nonlinear coefficients*. Therefore, we conclude that (32) is identical with the corresponding power series-like description for the whole amplifier that has been used in [3] (see equation (15) in [3]). And this ends the proof of the statement that the proper model for the results presented by Palumbo and Pennisi [3] for the feedback amplifier is the associated model in Fig. 2.

(On the occasion, we put the reader's attention to the fact that a small correction is needed in equation (19) in [3] because, generally, $H^{(2)}(f,f)\neq H^{(2)}(f,2f)$.)

In summary, having in mind the results of this and of the previous section, we can say that the means of calculation of harmonic distortion in weakly nonlinear feedback amplifies developed in [3] relies upon the approximation of the original model in Fig. 1(b) by the associated model in Fig.2. It is assumed that writing

$$x_{of}(t)=(Fx_{oa})(t)\cong x_o(t) \tag{34}$$

makes sense. The validity of (34) has been checked in [3] by carrying out calculations for some CMOS amplifiers and their comparison with the results of exact simulations.

BIBLIOGRAPHY

[1] S.S. Hakim, 1966: Feedback Circuit Analysis. Wiley, New York.

[2] S. Narayanan, 1970: Application of Volterra series to intermodulation distortion analysis of transistor feedback amplifiers. IEEE Trans. Circuit Theory, vol. CT-17, pp. 518-527.

[3] G. Palumbo, S. Pennisi, 2003: High-frequency harmonic distortion in feedback amplifiers: analysis and applications. IEEE Trans. Circuits and Systems-I: Fundamental Theory and Applications, vol. 50, pp. 328-340.

[4] E. Bedrosian, S.O. Rice, 1971: The output properties of Volterra systems (nonlinear systems with memory) driven by harmonic and Gaussian inputs. Proceedings of the IEEE, vol. 59, pp. 1688-1707.

[5] J.J. Bussgang, L. Ehrman, J.W. Graham, 1974: Analysis of nonlinear systems with multiple inputs. Proceedings of the IEEE, vol. 62, pp. 1088-1119.

[6] S.O. Cannizzaro, G. Palumbo, S. Pennisi, 2006: Effects of nonlinear feedback in the frequency domain. IEEE Trans. Circuits and Systems-I: Fundamental Theory and Applications, vol. 53, pp. 225-234.

O WPŁYWIE SPRZĘŻENIA ZWROTNEGO NA SKŁADOWE HARMONICZNE W UKŁADACH ANALOGOWYCH Z MAŁYMI NIELINIOWOŚCIAMI

Streszczenie

W pracy pokazano, że zastosowanie liniowego sprzężenia zwrotnego w układzie wzmacniacza analogowego pracującego w zakresie tzw. małych nieliniowości powoduje zwiększenie rzędu nieliniowości wykazywanej przez ten wzmacniacz w porównaniu ze wzmacniaczem bez sprzężenia. Przeanalizowano wpływ powyższego zjawiska na składowe harmoniczne wyższych rzędów, powstające przy pobudzeniu wzmacniacza pojedynczym sygnałem harmonicznym. Ponadto wyprowadzono model stowarzyszony wzmacniacza ze sprzężeniem zwrotnym. Pokazano, że wykorzystując ten model, i tylko wtedy, można zinterpretować w sposób prawidłowy metodę obliczeń zniekształceń harmonicznych we wzmacniaczach analogowych pracujących w zakresie tzw. małych nieliniowości, która to została zaproponowana przez Palumbo i Pennisi w jednym z ich ostatnio opublikowanych artykułów.

Słowa kluczowe: analiza zniekształceń harmonicznych, układy i systemy analogowe pracujące w zakresie tzw. małych nieliniowości, sprzężenie zwrotne, wpływ sprzężenia zwrotnego na składowe harmoniczne wyższych rzędów, szereg Volterry

UNIWERSYTET TECHNOLOGICZNO-PRZYRODNICZY
IM. JANA I JĘDRZEJA ŚNIADECKICH W BYDGOSZCZY
ZESZYTY NAUKOWE NR 253
TELEKOMUNIKACJA I ELEKTRONIKA 12 (2009), 21-42

SECOND AND HIGHER ORDER WHITENING IMAGE IN RECONSTRUCTION

Nirmal K. Bose[1], Umamahesh Srinivas[1], R. Lee Culver[2]

[1] Department of Electrical Engineering, The Pennsylvania State University
University Park, PA 16802, USA e-mail: bkn@engr.psu.deu

[2] Applied Research Laboratory, The Pennsylvania State University
PO Box 30, State College Pa 16804, USA

Abstract

In natural vision, the information in the natural environment tends to be acquired with minimum redundancy for subsequent handling akin to the technical problems and solutions encountered during the direct or indirect acquisition of compressed or decorrelated (whitened) multidimensional data, their transmission and processing. Considerable challenges are faced when the restrictions of Gaussianity in the signal and noise distributions, and linearity in processing are lifted. We improve recent results in vision research on the concept of higher order whitening of higher order statistics (HOS) based data. This is achieved by modifying the design of the filter and its inverse so that the processed image is of better quality than what is possible by direct implementation of a related scheme. Besides the modified higher order whitener and its complete analysis, the restriction of rotational symmetry in the power spectrum is eliminated in the two-dimensional derivation.

Keywords: whitening, higher-order whitening, power law for images.

1 Motivation

Whitening, in the traditional sense as originally used in filtering, prediction and smoothing of stationary random processes, orthogonalizes the data samples. In communications, such orthogonalization allows the detection algorithm to operate on each output sample independently. In [1], SNR maximization is the main scheme for improving the bit error probability. The difference between the receiver proposed there and conventional ones lies in the presence of a noise-whitening filter. A whitening filter prior to demodulation and smoothing improves the amplitude estimate. The concept of whitening is also widely used in models for signal processing tasks in the retina [2] to achieve a redundancy-reduced representation of

the original input signal. This reduction of second-order correlation is desirable for subsequent processing. The increasing thrust towards higher order statistics (HOS) based signal, data, and information processing embracing non-Gaussianity, non-stationarity and non-linear processing, necessitates removal of redundancies by higher order whitening.

The statistics of the noise affecting real channels significantly deviate from those corresponding to the Gaussian model. Non-Gaussian disturbances are commonly encountered in indoor environments, such as offices, hospitals, and factories as well as in underwater communications applications. If the disturbance is Gaussian and the fading amplitude is Rayleigh-distributed, then the structure of the optimum receiver is a bank of estimator-correlators, possibly preceded by a linear, whitening filter for handling the noise correlation. Receivers designed under the Gaussian noise assumption exhibit dramatic performance degradations in the presence of non-Gaussian impulsive noise. Therefore, attention has been directed (see [3] and [4]) toward the development of non-Gaussian noise models and the design of optimized detection structures that are able to operate in such hostile environments. Buzzi, Conti and Lops [5] modeled the noise as the product of two independent processes, namely, a complex-valued Gaussian process and a real nonnegative one, to account for the fluctuations of the noise source. The noise correlation was accounted for by a whitening linear filter, as for the case of Gaussian noise. Their structure consists of two parallel blocks, namely, an estimator of the short-term PSD plus a bank of estimators correlators, keyed to the estimated PSD of the disturbance. In the non-white noise case, the spectral shape of the noise needs to be estimated and "divided out" of the spectrum. That is, a "prewhitening filter" needs to be constructed and applied to the data so that the noise is whitened. Then the previous case can be applied.

2 Introduction to Second and Higher Order Whitening

Whitening, basically, decorrelates a signal or image by removing statistical redundancies exhibited in the second order statistics of, say, a zero-mean wide-sense stationary random process. A natural image or a sequence of images has considerable spatial as well as temporal (in the case of image sequences) correlation whose removal calls for whitening by linear filtering. In a non-stationary image, higher order correlations that remain require nonlinear methods for their removal based on procedures, referred to as higher order whitening, applied to statistical image models. Crucial in such modeling is the notion of scale-invariance, which refers to an image description that remains fixed with change of image-scale. Scale-invariance is a form of self-similarity, stochastically, a property manifested in a fractal, approximated by objects ubiquitous in nature, like coastlines, clouds, and snow flakes, among many other occurrences in nature.

Wide-sense stationary (WSS) assumption is basic to Wiener filtering and smooth-

ing theory while the popularity of the Gaussian distribution is due to various nice properties it embodies. Images and sequences of images may neither be stationary nor Gaussian and may exhibit highly non-Gaussian statistical behavior. Whitening of Gaussian signals is possible with principal component analysis (PCA) and, in this case, decorrelation is equivalent to statistical independence. Subsequently, independent component analysis (ICA) was developed for handling non-Gaussian sources especially in the problem of blind source separation. ICA methods, however, do not yield statistically independent outputs in case of model inaccuracy and the assumption on the model is also confined, for the most part, to a linear mixture of independent sources whose number also has to be estimated [6].

In the case of a non-Gaussian source vector, when decorrelation does not imply statistical independence, ICA separates a signal (with not more than one Gaussian component) into additive subcomponents subject to the assumption of mutual statistical independence of the non-Gaussian source signals. Most ICA methods are not able to extract the actual number (or order) of source signals, nor the signs or the scales of the sources. Furthermore, ICA is restricted to source vectors $\mathbf{y} = P\mathbf{x}$, where P is a rectangular matrix multiplying the vector $\mathbf{x}$ whose components are assumed to be statistically independent. A decomposition applicable to any matrix, square or rectangular, is the singular value decomposition (SVD),

$$P = USV^T,$$

where U and V are orthogonal matrices and S is the diagonal matrix of singular values. In the case when the singular values are all positive (in general, they are nonnegative) and S is square, $S^{-1/2}$ exists and $S^{-1/2}U^T$ is a whitening transformation (like in PCA) and V is found by minimizing the negentropy.

Typical algorithms for ICA use centering, whitening and dimensionality reduction as preprocessing steps in order to simplify and reduce the complexity of the problem for the actual iterative algorithm. Whitening ensures that all dimensions are treated equally a priori before the algorithm is run. Algorithms for ICA include infomax, FastICA and JADE, but there are many others also. In [7] the SWM (support width measure) contrast is used to solve the ICA problem involving correlated images.This approach is motivated by the unsatisfying results of JADE and FastICA for the same problem, when no other preprocessing than whitening like filtering is used. Its main advantages are its theoretical contrast convexity for bounded sources, its geometrical interpretation and its simplicity. The Probabilistic Independent Component Analysis model is aimed at solving the problem of overfitting in classical ICA. Support vector machines (SVMs) are a set of related supervised learning methods used for classification and regression. They belong to a family of generalized linear classifiers. They can also be considered a special case of Tikhonov regularization. A special property of SVMs is that they simultaneously minimize the empirical classification error and maximize the geometric margin; hence they are also known as maximum margin classifiers.

3 Power Law and Statistical Modeling of Image Databases

Knowledge of the statistical properties of natural images is important in many diverse fields of research. For example, physiologists can test their theories of evolutionary optimization of vision against the measured properties of the environment. At the other end, computer scientists can use the measured statistics to create convincing synthetic environments for the film industry. The class of natural images is receiving attention in image databases used for vision modeling research. These images are sufficiently homogeneous so that a statistical representation model may be constructed for use in compression, denoising, deblurring and also, in the future, for superresolution. Representations using Fourier bases, Gabor bases, wavelet bases and their various ramifications have been extensively used. However, instead of fixed bases, serious attention is needed for generating data driven bases for manipulation and transformation of the data for various tasks (like cognition in the human sensory system). Relatively meagre attention has been directed to research with this type of massive volumetric data. At present, such research thrust is highly computer software (and hardware) driven but more coupling with analytical machineries is needed.

3.1 1-D Analysis

Balboa et al [8] derived results for the power spectra in 1-D by assuming a cut to consist of M independent regions of constant intensity such that the intensity profile of the j-th region is:

$$I_j(x) = \begin{cases} \tilde{I}_j, & x \in [x_j, x_{j+1}] \\ 0, & \text{otherwise.} \end{cases}$$

Here, the $\tilde{I}_j$'s are positive constants and the x_j's are chosen suitably to divide the cut into constant intensity regions. The 1-D cut $I(x)$ can now be represented as

$$I(x) = \sum_{j=1}^{M} I_j(x),$$

whose Fourier transform is

$$\hat{I}(\omega) = \frac{j}{\omega} \sum_{j=1}^{M} \tilde{I}_j (e^{-j\omega x_{j+1}} - e^{-j\omega x_j}),$$

where $\omega = 2\pi f$ and f is the spatial frequency. The power spectrum of $I(x)$ is

$$|\hat{I}(\omega)|^2 = \frac{1}{\omega^2} \left| \sum_{j=1}^{M} \tilde{I}_j (e^{-j\omega x_{j+1}} - e^{-j\omega x_j}) \right|^2.$$

The observation made in [8] is that as spatial frequency increases, the summation term in the last equation varies but the envelope falls as $1/\omega^2$. Further, at very

low spatial frequencies corresponding to $\omega < 1/(x_{J+1} - x_J)$, where $(x_{j+1} - x_j)$ is maximum at $j = J$, a Taylor series expansion (around $x = x_j$) leads to the approximation

$$|\hat{I}(\omega)|^2 \simeq \left| \sum_{j=1}^{M} \tilde{I}_j(x_{j+1} - x_j)(e^{-j\omega x_j}) \right|^2,$$

which reveals that the envelope is approximately constant at very low frequencies.

3.2 2-D Analysis

The reasoning presented above is generalized to a 2-D image $I(x, y)$. Again, the image is treated as continuous rather than as a set of discrete pixels. The 2-D Fourier transform of $I(x, y)$ is

$$\hat{I}(\omega_x, \omega_y) - \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} I(x, y) e^{-j\omega_x x} e^{-j\omega_y y} dx\, dy,$$

where ω_x and ω_y are the bivariate independent angular frequency components. Actually, the integrand is non-zero only over the finite support of the image in 2-D space.

The Fourier transform along the ω_x-axis at $\omega_y = 0$ is

$$\hat{I}(\omega_x, 0) = \int_{-\infty}^{\infty} \left(\int_{-\infty}^{\infty} I(x, y) e^{-j\omega_x x} dx \right) dy.$$

The integral within parentheses is evaluated for an arbitrary but fixed y for equivalence to a 1-D cut analysis. Now let $I(x, y) = \sum_{j=1}^{M(y)} I_j(x, y)$, where

$$I_j(x, y) = \begin{cases} \tilde{I}_j(y), & x \in [x_j, x_{j+1}] \\ 0, & \text{otherwise.} \end{cases}$$

Note that $\tilde{I}_j(y)$ is a constant for the specified range of x at a particular value of y. It however may vary with y and this dependence on y is expressed as $\tilde{I}_j(y)$. Also, the number of constant-intensity regions varies with y and this explains the dependence of M on y in the summation. Proceeding parallel to the 1-D case discussed above leads to the modified results

$$\hat{I}(\omega_x, 0) = \frac{j}{\omega_x} \int_{-\infty}^{\infty} \sum_{j=1}^{M(y)} \tilde{I}_j(y)(e^{-j\omega x_{j+1}} - e^{-j\omega x_j}) dy, \quad (1)$$

$$|\hat{I}(\omega_x, 0)|^2 = \frac{1}{{\omega_x}^2} \left| \int_{-\infty}^{\infty} \sum_{j=1}^{M(y)} \tilde{I}_j(y)(e^{-j\omega x_{j+1}} - e^{-j\omega x_j}) dy \right|^2. \quad (2)$$

The x_{j+1} and x_j implicitly correspond to (x_{j+1}, y) and (x_j, y), respectively. Clearly, the envelope of the power spectrum falls as $1/{\omega_x}^2$ even though the integral term

may fluctuate. A similar analysis for the case $\omega_x = 0$ leads to the result that the power spectrum falls as $1/{\omega_y}^2$.

The more general case of 2-D Fourier transform is now considered. Extending the idea of constant-intensity regions to two dimensions, the image is divided into rectangular patches, each having pixels of the same intensity values. The image is divided into MN rectangles using $M+1$ vertical lines $x = x_j, j = 1, 2, \ldots, M+1$, and $N+1$ horizontal lines $y = y_k, k = 1, 2, \ldots, N+1$. The rectangle bounded by the four lines $x = x_j$, $x = x_{j+1}$, $y = y_k$ and $y = y_{k+1}$ is assumed to have constant intensity $\tilde{I}_{jk}$.

3.3 Division of the image into rectangular patches

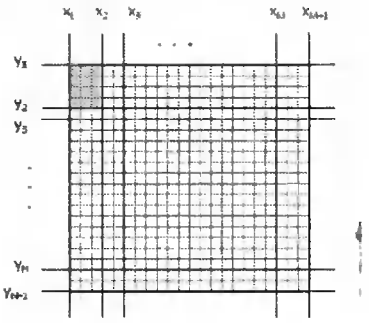


Figure 1: Figure showing the division of the original image into constant-intensity rectangular patches. The shaded region shows one such rectangular patch.

The procedure to divide the image is as follows. Starting from the left vertical boundary of the image (i.e., $x = x_1$), the image is scanned from left to right to check if each column is identical to the column immediately preceding it. The line $x = x_2$ is drawn between adjacent columns L_1 and L_2 where at least one pixel of L_2 differs from the corresponding pixel on the same row in the preceding column L_1. Proceeding thus, all the vertical lines $x = x_2$ to $x = x_M$ are located. Similarly, starting with the upper horizontal image boundary ($y = y_1$), the image is scanned from top to bottom and each horizontal line $y = y_k$ is located between rows where at least one pixel differs from the corresponding pixel on the same column in the preceding row. Figure 1 schematically shows the division of the original image.

Some observations regarding the image division are discussed next. This method does not divide the image into the largest possible regions of constant intensity. The particular method discussed above divides the image based on a uniform grid which simplifies mathematical calculations, and it is likely that more than one region may have identical intensity values. For a white noise image, the rectangles will be of size 1 pixel, since the pixel intensity values are completely independent

of other pixels. Balboa et al [8] derived the result for the 1-D case based on the assumption that images consist of individual constant-intensity patches. This assumption is not valid in the type of 2-D generalization considered here.

3.4 2-D Fourier analysis: general case

Let the intensity profile of the image be described by:

$$I(x,y) = \sum_{k=1}^{N}\sum_{j=1}^{M} I_{jk}(x,y),$$

where

$$I_{jk}(x,y) = \begin{cases} \tilde{I}_{jk}, & x \in [x_j, x_{j+1}] \text{ and } y \in [y_k, y_{k+1}] \\ 0, & \text{otherwise.} \end{cases}$$

For a specified 2-tuple (j,k), $I_{jk}(x,y)$ denotes a constant intensity rectangular patch and $\tilde{I}_{jk}$ is the value of this constant. Let $\hat{I}_{jk}(\omega_x,\omega_y)$ be the Fourier transform of the patch $I_{jk}(x,y)$. Then, the Fourier transform of $I(x,y)$ is given by:

$$\begin{aligned}
\hat{I}(\omega_x,\omega_y) &= \sum_{k=1}^{N}\sum_{j-1}^{M} \hat{I}_{jk}(\omega_x,\omega_y) \\
&= \sum_{k=1}^{N}\sum_{j=1}^{M} \int_{y_k}^{y_{k+1}} \int_{x_j}^{x_{j+1}} I_{jk}(x,y) e^{-j\omega_x x} e^{-j\omega_y y} dx\, dy \\
&= \sum_{k=1}^{N}\sum_{j=1}^{M} \left\{ \tilde{I}_{jk} \int_{x_j}^{x_{j+1}} e^{-j\omega_x x} dx \int_{y_k}^{y_{k+1}} e^{-j\omega_y y} dy \right\} \\
&= \frac{-1}{\omega_x \omega_y} \sum_{k=1}^{N}\sum_{j-1}^{M} \tilde{I}_{jk} (e^{-j\omega_x x_{j+1}} - e^{-j\omega_x x_j})(e^{-j\omega_y y_{k+1}} - e^{-j\omega_y y_k}).
\end{aligned}$$

Note that $\tilde{I}_{jk}$ is a function of j and k and cannot be moved outside the summation in the last equation. The power spectrum is

$$|\hat{I}(\omega_x,\omega_y)|^2 = \frac{1}{{\omega_x}^2 {\omega_y}^2} \left| \sum_{k=1}^{N}\sum_{j=1}^{M} \tilde{I}_{jk} (e^{-j\omega_x x_{j+1}} - e^{-j\omega_x x_j})(e^{-j\omega_y y_{k+1}} - e^{-j\omega_y y_k}) \right|^2. \tag{3}$$

The term inside the summation on the RHS varies with j and k, but the envelope of the power spectrum falls as $1/({\omega_x}^2 {\omega_y}^2)$. At very low spatial frequencies, $\omega_x < 1/(x_{J+1} - x_J)$, where $(x_{j+1} - x_j)$ is maximum for $j = J$, and $\omega_y < 1/(y_{K-1} - y_K)$, where $(y_{k+1} - y_k)$ is maximum for $k = K$. Then, $e^{-j\omega_x x_{j+1}} \simeq e^{-j\omega_x x_j} - j\omega_x (x_{j+1} - x_j) e^{-j\omega_x x_j}$, and $e^{-j\omega_y y_{k+1}} \simeq e^{-j\omega_y y_k} - j\omega_y (y_{k+1} - y_k) e^{-j\omega_y y_k}$, by Taylor series

expansion (around each 2-tuple ($x = x_j$, $y = y_k$)). So,

$$|\hat{I}(\omega_x, \omega_y)|^2 \simeq \frac{1}{{\omega_x}^2{\omega_y}^2}\left|\sum_{k=1}^{N}\sum_{j=1}^{M}\tilde{I}_{jk}(-j\omega_x)(x_{j+1}-x_j)e^{-j\omega_x x_j}(-j\omega_y)(y_{k+1}-y_k)e\right.$$

$$= \left|\sum_{k=1}^{N}\sum_{j=1}^{M}\tilde{I}_{jk}(x_{j+1}-x_j)e^{-j\omega_x x_j}(y_{k+1}-y_k)e^{-j\omega_y y_k}\right|^2.$$

This shows that at very low spatial frequencies, the envelope of the power spectrum is approximately constant with frequency.

4 Observations

To summarize the results, the 2-D power spectra of images fall off as $1/({\omega_x}^2{\omega_y}^2)$, where ω_x and ω_y are the components in the wavenumber (angular frequency) domain. Along the ω_x-axis, the spectra fall off as $1/{\omega_x}^2$, and as $1/{\omega_y}^2$ along the ω_y-axis. At very low spatial frequencies, the envelope of the spectrum is approximately constant. In [9], the authors provide justification for why the spectrum is flat at very low frequencies.

4.1 Gradients of illumination

The case of gradients of illumination discussed in Balboa [8] for 1-D can be extended to 2-D. The simplest model for expressing gradients (variable-separable case) is:

$$I_{jk}(x,y) = \begin{cases} \tilde{I}_{jk} + \gamma xy, & x \in [x_j, x_{j+1}] \text{ and } y \in [y_k, y_{k+1}] \\ 0, & \text{otherwise} \end{cases}$$

where γ is a constant. Since $I(x,y) = \sum_{k=1}^{N}\sum_{j=1}^{M} I_{jk}(x,y)$, the Fourier transform of $I(x,y)$ is obtained as:

$$\hat{I}(\omega_x, \omega_y) = \sum_{k=1}^{N}\sum_{j=1}^{M}\hat{I}_{jk}(\omega_x, \omega_y),$$

where

$$\hat{I}_{jk}(\omega_x, \omega_y) = \int_{y_k}^{y_{k+1}}\int_{x_j}^{x_{j-1}}(\tilde{I}_{jk} + \gamma xy)e^{-j\omega_x x}e^{-j\omega_y y}dx\,dy$$

$$= \tilde{I}_{jk}\int_{x_j}^{x_{j+1}} e^{-j\omega_x x}dx\int_{y_k}^{y_{k+1}} e^{-j\omega_y y}dy + \gamma\int_{x_j}^{x_{j+1}} xe^{-j\omega_x x}dx\int_{y_k}^{y_{k+1}} y$$

$$= \frac{-\tilde{I}_{jk}}{\omega_x\omega_y}(e^{-j\omega_x x_{j+1}} - e^{-j\omega_x x_j})(e^{-j\omega_y y_{k+1}} - e^{-j\omega_y y_k}) + I_1,$$

and I_1 can be simplified using integration by parts as

$$I_1 = \gamma \left\{ \frac{j}{\omega_x}(x_{j+1}e^{-j\omega_x x_{j+1}} - x_j e^{-j\omega_x x_j}) + \frac{1}{\omega_x^2}(e^{-j\omega_x x_{j-1}} - e^{-j\omega_x x_j}) \right\}$$
$$\left\{ \frac{j}{\omega_y}(y_{k+1}e^{-j\omega_y y_{k+1}} - y_k e^{-j\omega_y y_k}) + \frac{1}{\omega_y^2}(e^{-j\omega_y y_{k+1}} - e^{-j\omega_y k_k}) \right\}.$$

Thus, $\hat{I}_{jk}(\omega_x, \omega_y)$ contains a term proportional to $1/(\omega_x^2\omega_y^2)$ and the power spectrum given by $|\hat{I}(\omega_x, \omega_y)|^2 = |\sum_{k=1}^{N}\sum_{j=1}^{M}\hat{I}_{jk}(\omega_x, \omega_y)|^2$ is proportional to $1/(\omega_x^4\omega_y^4)$, which implies a steeper descent than $1/(\omega_x^2\omega_y^2)$. Thus, illumination gradients lead to a steeper descent of the power spectrum in 2-D. Other models to describe the effects of illumination are possible and only the simplest case has been dealt with here.

4.2 Statistical independence

According to Ruderman [10], *statistically independent* regions have random size, location and intensity. The segmentation performed by Ruderman is based on dividing images into largest possible regions of constant intensity and the regions are of various irregular shapes. Consequently, adjacent independent regions in the image are unlikely to have correlation in pixel intensities. In contrast, the image segmentation into rectangular patches explained in this report is based on a uniform rectangular grid of lines. It is possible to have adjacent regions with the same intensity since both regions may in fact belong to the same object. Hence, the rectangular regions may not be statistically independent.

One more point to be noted is that since the power spectrum $|\hat{I}(\omega_x, \omega_y)|^2$ is proportional to $1/(\omega_x^2\omega_y^2)$, points in the wavenumber (angular frequency) domain satisfying $\omega_x\omega_y = c$ where c is some constant, show identical spectrum fall-off behavior. These points lie on a rectangular hyperbola. In comparison, for the rotation-averaged case, the power spectrum $\hat{P}(\omega) \propto 1/\omega^2$ and $\omega = \sqrt{\omega_x^2 + \omega_y^2}$ and the points showing identical fall-off behavior lie on a circle.

5 Higher Order Whitening and De-whitening Filters

This section discusses higher order whitening, in particular the filter described by Gluckman [11]. A second-order whitening filter is first designed to remove correlations between image pixel intensities, as specified by Olshausen [12]. It is established that images have considerable regularity in their second order spatial correlations as measured by the autocorrelation or power spectrum [13]. It has been observed that the orientation-averaged power spectrum $P(f)$ varies with spatial angular frequency f as:

$$P(f) \propto 1/f^2, \tag{4}$$

where $f = \sqrt{f_1^2 + f_2^2}$ and the 2-D frequency response is defined in the (f_1, f_2)-plane. So, it is logical to design a whitening filter whose magnitude of frequency response varies as f so that the whitened image has a 'flat' power spectrum. A crucial assumption in this procedure is that of circular symmetry of the inherently 2-D frequency response in the (f_1, f_2)-plane required for filtering 2-D image data.

After second order whitening, lines and edges, which correspond to higher order correlations, are visible in the whitened image. These correlations can be removed using higher order whitening. In the method discussed in [11], the (second order) whitened image $W(\mathbf{x})$ may be expressed as:

$$W(\mathbf{x}) = g(\mathbf{x}) * I(\mathbf{x}), \tag{5}$$

where $\mathbf{x} \equiv (x_1, x_2)$, $I(\mathbf{x})$ is the original image and $g(\mathbf{x})$ is the second order whitening filter unit impulse response with the property that the magnitude of its Fourier transform, $|\hat{g}(f)|$, is proportional to f. $W(\mathbf{x})$ may assume both positive and negative values. The statistics of second order whitened images are unstable (in the sense of large dynamic range) due to the high kurtosis of pixel intensities and so Gluckman suggests usage of the log magnitude whitened image $L(\mathbf{x}) = log|W(\mathbf{x})|$ for further whitening. He further conjectured, based only on simulations, that $L(\mathbf{x})$ also obeys a power law similar to that in Equation (4) and hence filtering it with the second order whitening filter g produces higher order whitening. The higher order whitened image $H(\mathbf{x})$ is obtained as

$$H(\mathbf{x}) = sgn(W(\mathbf{x}))\ exp\{g(\mathbf{x}) * log|W(\mathbf{x})|\}. \tag{6}$$

This equation characterizes a form of homomorphic filtering. The *sgn* term represents the *signum* function. $H(\mathbf{x})$ in (6) assumes both positive and negative values due to the presence of $sgn(W(\mathbf{x}))$. Figure 2 shows the higher order whitening scheme followed in [11].

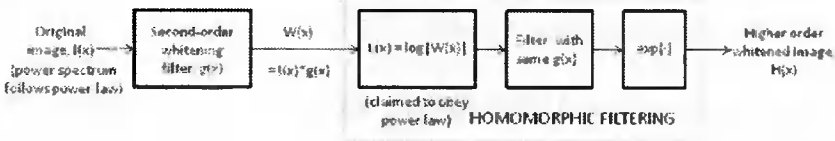


Figure 2: Schematic showing the steps in the higher-order whitening procedure suggested by Gluckman.

A brief analysis of the above higher order filter equation follows. The original image can be recovered from the higher order whitened image by a straightforward inverting procedure described by Equation 10 in reference [11]. The identities $H(\mathbf{x}) = |H(\mathbf{x})|sgn(H(\mathbf{x}))$ and $sgn(H(\mathbf{x}))^2 = 1$ are easily seen to hold. Applying the *signum* function to both sides of Eq. (6), one gets $sgn(H(\mathbf{x})) = sgn(W(\mathbf{x}))$.

For notational brevity, the explicit functional dependence on the variable x is dropped. Multiplying both sides of Eq. (6) by $sgn(H)$, one gets

$$\begin{aligned} Hsgn(H) &= sgn(H)sgn(W)\ exp\{g * log|W|\} \\ &= exp\{g * log|W|\}. \end{aligned}$$

But the left hand side of the preceding equation is equal to $|H|$. So, denoting the inverse of the whitening filter unit impulse response g by g^{-1}, it follows that

$$\begin{aligned} |W| &= exp\{g^{-1} * log|H|\} \\ \text{i.e., } W &= sgn(W)\ exp\{g^{-1} * log|H|\}. \end{aligned}$$

Therefore, convolving both sides of Eq. (5) with g^{-1}, substituting the preceding expression for W and noting that $sgn(W) = sgn(H)$, it follows that

$$I(x) = g^{-1} * [sgn(H)\ exp\{g^{-1} * log|H|\}]. \tag{7}$$

Eq. (7) is identical to Eq. 10 in [11]. To get the modified filter proposed here, drop the sgn term in Eq. (6) and then replace the higher order whitening filter H by

$$H_1 = exp\{g * log|W|\}. \tag{8}$$

Noting that H_1 assumes only positive values (since $exp\{g * log|W|\} > 0$), it follows that

$$g * log|W| = log(H_1) = log|H_1|.$$

Following arguments similar to the ones above and inserting the identity $sgn(H_1) \equiv 1$, the counterpart of Eq. (7) becomes

$$I = g^{-1} * [sgn(H_1)\ exp\{g^{-1} * log|H_1|\}]. \tag{9}$$

Equations (7) and (9) are equivalent except that $sgn(H_1) \equiv 1$ in Eq. (9) (because in Eq. (8), $H_1 > 0$ always). It will be found later from simulations that this replacement of H by H_1 improves the overall quality of the reconstructed images after higher order whitening.

6 Simulation results

6.1 Second order whitening based on 2-D power law

Results of simulations to verify the 2-D power law proposed in Equation (3) are presented in this section. A second order whitening filter G is designed in the frequency domain such that:

$$\hat{G}(\omega_x, \omega_y) = \begin{cases} \omega_x\omega_y, & \omega_x \neq 0 \text{ and } \omega_y \neq 0 \\ \omega_x, & \omega_y = 0 \\ \omega_y, & \omega_x = 0. \end{cases}$$

From Eqs. (2) and (3), it can be seen that filtering an image with the above whitening filter leads to an approximately flat spectrum over a significant range of

frequencies. The image power spectrum shows that most of the energy is concentrated in the low spatial frequencies and the spectrum falls off with higher frequencies. The power spectrum of the filter proposed above increases in magnitude with spatial frequencies, resulting in a whitened image with amplified high-frequency noise. To avoid this problem, the whitening filter may be combined with a 2-D lowpass filter and then applied to images. The modified whitening filter, based on a filter proposed by Liao et al in [14], is given by:

$$\hat{G}_1(\omega_x, \omega_y) = \hat{G}(\omega_x, \omega_y) \exp\left\{ -k \left(\frac{\sqrt{\omega_x^2 + \omega_y^2}}{\omega_c} \right)^n \right\}, \tag{10}$$

where k, ω_c and n are parameters which can be suitably varied to achieve a flat spectrum ($k = 1$ in [14]). The exponential term acts as a lowpass filter, and the cut-off frequency is determined by ω_c.

For the set of images considered, the parameter value $\omega_c = 0.05N$, where the input image size is $N \times N$. By suitably choosing the parameters k and n in Equation (10), flat spectra over larger range of frequencies can be obtained. The whitening was performed on different sets of images: natural, mixed and thermal. The natural and mixed scenery visible images have been sourced from the van Hateren database [15], while the thermal images have been sourced from Morris et al [16].

The whitened image obtained in each case reveals that most of the correlations between pixels in the image have been removed. As a means of comparison, the rotation-averaged 1-D power spectra are plotted. Some of the image spectra have distinct streaks along the ω_x and ω_y axes. These streaks may not be sufficiently flattened in the process of whitening. Schaaf and Hateren [17] suggest that one reason for the appearance of such streaks could be that, on average, there are more horizontally and vertically oriented structures in natural images. Langer [18] explains why scaling may fail in individual images, and how the appearance of the streaks is determined by the shape and size of the image window.

The whitening filter is separable and is easily invertible. Accordingly, a 2-D de-whitening filter $\hat{G}^{-1}(\omega_x, \omega_y)$ is designed in the frequency domain as follows:

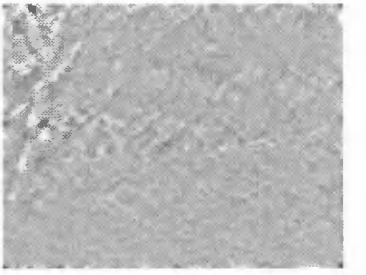
$$\hat{G}^{-1}(\omega_x, \omega_y) = \begin{cases} 1/(\omega_x \omega_y), & \omega_x \neq 0 \text{ and } \omega_y \neq 0 \\ 1/\omega_x, & \omega_y = 0 \\ 1/\omega_y, & \omega_x = 0. \end{cases} \tag{11}$$

The modified whitening filter in Eq. (10) has an exponential term which also can be incorporated into the de-whitening filter as:

$$\hat{G}_1^{-1}(\omega_x, \omega_y) = \hat{G}^{-1}(\omega_x, \omega_y) \exp\left\{ k \left(\frac{\sqrt{\omega_x^2 + \omega_y^2}}{\omega_c} \right)^n \right\}. \tag{12}$$

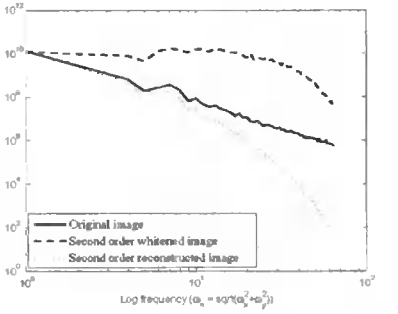
6.1.1 Natural image - second order whitening and de-whitening using 2-D filter

(a) Original image.



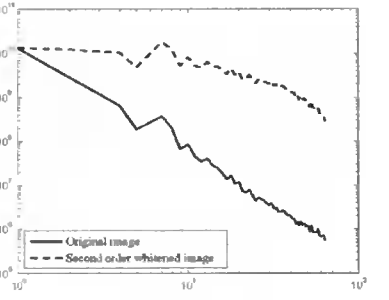
(b) Whitened image using 2-D filter.

(c) Reconstructed image using 2-D de-whitening filter.



(d) Rotation-averaged power spectra.

(e) Whitened image - Gluckman's filter.



(f) Rotation-averaged power spectra - Gluckman.

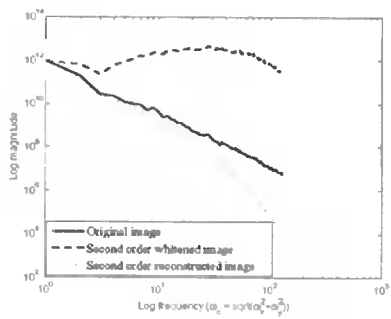
Figure 3: Natural image - second order whitening and de-whitening. 2-D filter parameter values used: $k = 0.3, \eta = 1.2$. The 2-D filter used in (b) decorrelates the image better than Gluckman's filter in (e).

6.1.2 Mixed scenery visible image - second order whitening and de-whitening using 2-D filter

(a) Original image.

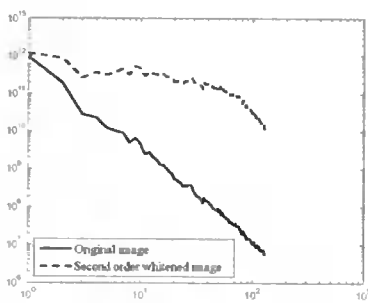
(b) Whitened image using 2-D filter.

(c) Reconstructed image using 2-D de-whitening filter.



(d) Rotation-averaged power spectra.

(e) Whitened image - Gluckman's filter.

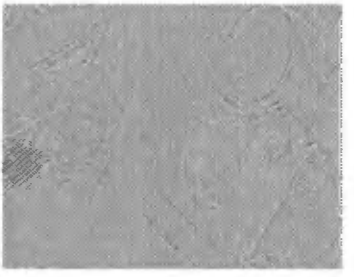


(f) Rotation-averaged power spectra - Gluckman.

Figure 4: Mixed scenery image - second order whitening and de-whitening. 2-D filter parameter values used: $k = 0.4, \eta = 0.9$. The 2-D filter used in (b) decorrelates the image better than Gluckman's filter in (e).

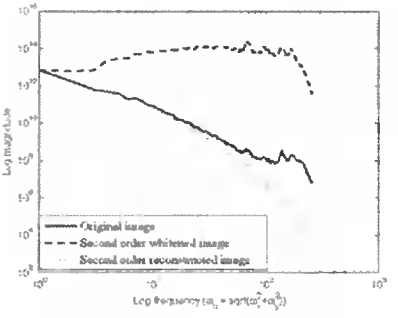
6.1.3 Textured visible image - second order whitening and de-whitening using 2-D filter

(a) Original image.



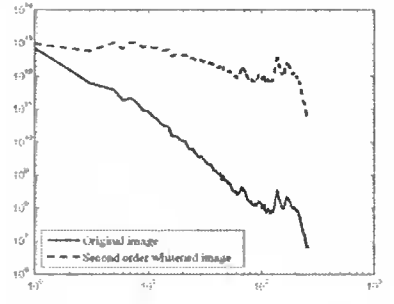
(b) Whitened image using 2-D filter.

(c) Reconstructed image using 2-D de-whitening filter.



(d) Rotation-averaged power spectra.

(e) Whitened image - Gluckman's filter.



(f) Rotation-averaged power spectra - Gluckman.

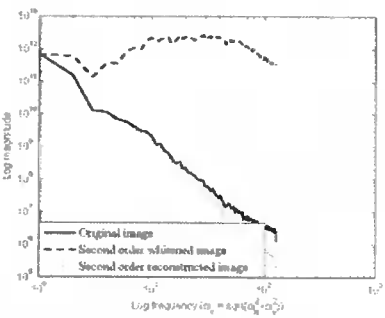
Figure 5: Mixed scenery image (highly-textured) - second order whitening and de-whitening. 2-D filter parameter values used: $k = 0.3, \eta = 1.2$. The 2-D filter used in (b) decorrelates the image better than Gluckman's filter in (e).

6.1.4 Thermal image - second order whitening and de-whitening using 2-D filter

(a) Original image.

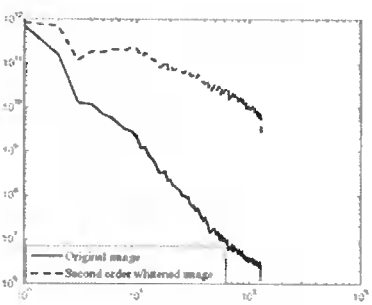
(b) Whitened image using 2-D filter.

(c) Reconstructed image using 2-D de-whitening filter.



(d) Rotation-averaged power spectra.

(e) Whitened image - Gluckman's filter.



(f) Rotation-averaged power spectra - Gluckman.

Figure 6: Thermal image - second order whitening and de-whitening. 2-D filter parameter values used: $k = 0.1, \eta = 1.0$. The 2-D filter used in (b) decorrelates the image better than Gluckman's filter in (e).

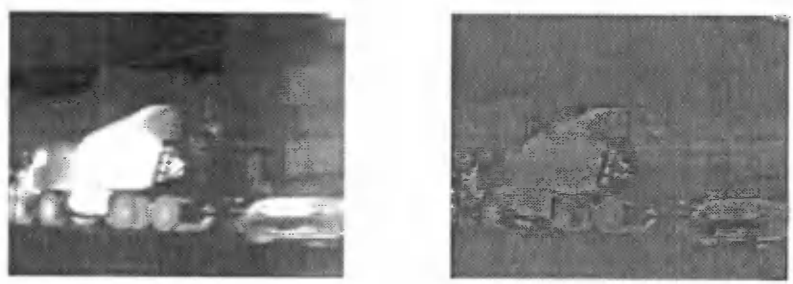
Simulation results from a variety of test image data - natural, mixed scenery, textured, thermal - show that a satisfactory reconstruction of the original image can be obtained by filtering with G^{-1} from Eq. (11) itself. This obviates the need for the additional three parameters k, η and ω_c at the de-whitening side. Exact reconstruction is achieved by using the de-whitening filter specified in Eq. (12).

6.2 Higher order whitening

Higher order whitening is performed on a sample thermal image sourced from Morris et al [16]. The procedure followed is described in Equations (5) and (6). The second order and higher order whitened images are displayed. The reconstructed original image is shown for the two cases: with the signum function in the higher order filter as proposed by Gluckman, and without the signum function as proposed in this paper. Results are shown in Figures 7-9.

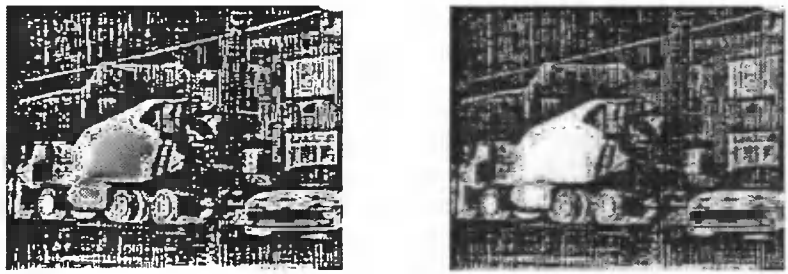
Next, the higher order whitening and de-whitening using Gluckman's procedure is repeated using the 2-D whitening filter proposed in this paper. The higher order whitened image and the log power spectra are shown in Figures 10 and 11, respectively.

6.2.1 Thermal image - higher order whitening and de-whitening using Gluckman's procedure



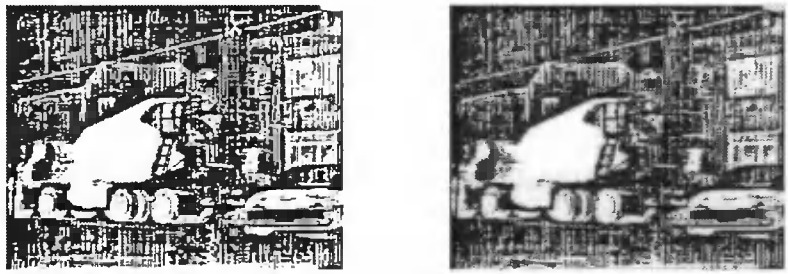
(a) (b)

Figure 7: (a) Original thermal image, (b) Second order whitened image using Gluckman's orientation-averaged whitening procedure.



(a) (b)

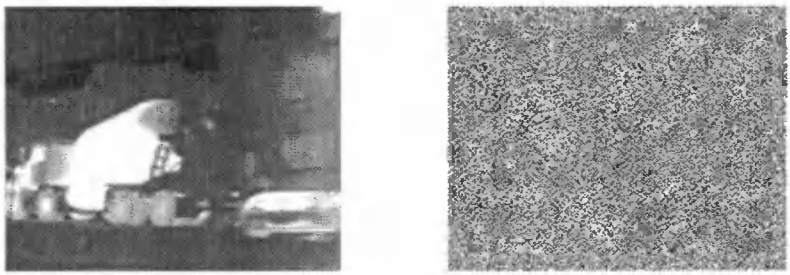
Figure 8: (a) Higher order whitened image using Gluckman's higher order whitening procedure. (b) Reconstructed original image (signum term used in higher order whitening, as proposed by Gluckman).



(a) (b)

Figure 9: (a) Higher order whitened image using Gluckman's higher order whitening procedure, (b) Reconstructed original image (signum term not used in higher order whitening, as proposed in this paper).

6.2.2 Thermal image - higher order whitening and de-whitening combining 2-D filter and Gluckman's procedure



(a) (b)

Figure 10: (a) Original thermal image, (b) Higher order whitened image (decorrelated) using Gluckman's procedure and a 2-D second order whitening filter.

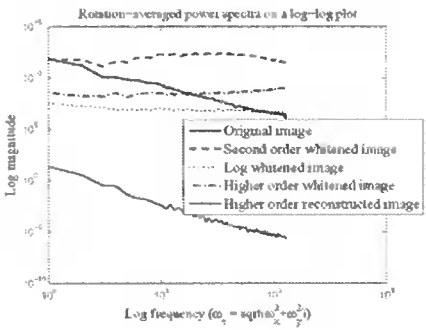


Figure 11: Power spectra plotted as a function of the rotation-averaged frequency for comparison.

7 Conclusions

This paper is concerned with the problems and solutions encountered during the acquisition of compressed multidimensional data, their transmission and processing when the assumptions of Gaussianity in the signal and noise distributions and linearity in processing need not hold simultaneously. While attempting such a generalization one feels the need for more powerful analytical tools with which to develop technical devices to tackle the increasing complexities of the models. For example the fundamental concept of whitening for second order statistics based

signal or data forms the nucleus for generalization to higher order whitening of higher order statistics (HOS) based data, where both the higher order whitening filter and its inverse have nonrational nonlinear input/output descriptions. Their implementations are done by software for the simulated examples in this paper. A modification of the second and higher order whitening filters in [11] is proposed in this paper. It is seen from simulation on infrared images that the visual quality of the reconstructed image is better using the proposed modifications. For example, in Figure 8, black specks are seen in the white regions (corresponding to hotter areas) of the reconstructed image, while the specks are missing in the reconstructed image in Figure 9. This suggests that the *sgn* factor is not only redundant but also may be harmful in higher order filtering. Also, Gluckman's conjecture that the log magnitude second order whitened image follows the power law in Equation (4) is not necessarily true, as shown by the power spectrum plot in Figure 11, in which the log whitened image spectrum is nearly flat.

The concept of whitening, in turn, has strong links to choice of bases for compressed signal representation, their coding (sparse or otherwise) and reconstruction by constrained optimization using mixed norms. These links are currently under detailed investigation with respect to properties like wavelength diversity as briefly reported here and elsewhere [19] with respect to visible and thermal wavelength images. The results here contribute to the goal for unification of different but related concepts (like whitening and compressed signals) and further development of mathematical (simultaneous matrix diagonalization, simultaneous matrix singular value decomposition, simultaneous matrix polar decomposition) as well as probabilistic descriptions (Gaussian as well as non-Gaussian distributions) for detection, estimation, sampling and processing of signals.

Acknowledgement

The research reported here was conducted under the sponsorship of the National Science Foundation Grant CCF-0429481 and the Office of Naval Research, Undersea Signal Processing, Program Code 321US. Constructive comments by the reviewers are gratefully acknowledged and incorporated in the revised manuscript prior to publication.

Bibliography

[1] A. M. Monk, M. Davis, L. B. Milstein, and C. W. Helstrom, "A noise whitening approach to multiple access noise rejection part I: theory and background," *IEEE Journal on Selected Areas in Communications*, vol. 12, no. 5, June 1994.

[2] J. J. Atick and A. N. Redlich, "What does the retina know about natural scenes?" *Neural Computations*, vol. 4, pp. 196–210, 1992.

[3] S. Buzzi, E. Conte, A. D. Maio, and M. Lops, "Optimum diversity detection over fading dispersive channels with non-Gaussian noise," *IEEE Trans. Signal Process.*, vol. 49, no. 4, pp. 767–776, April 2001.

[4] H.-F. Chen, X.-R. Cao, H.-T. Fang, and J. Zhu, "Nonlinear adaptive blind whitening for MIMO channels," *IEEE Trans. Signal Process.*, vol. 53, no. 8, pp. 2635–2647, August 2005.

[5] S. Buzzi, E. Conte, and M. Lops, "Optimum detection over Rayleigh-fading, dispersive channels, with non-Gaussian noise," *IEEE Trans. Comm.*, vol. 45, no. 9, pp. 1061–1069, September 1997.

[6] A. Hyvärinen, J. Karhunen, and E. Oja, *Independent Component Analysis.* Wiley Interscience, 2001.

[7] F. Vrins and M. Verleysen, "SWM: A class of convex contrasts for source separation," in *Proc. IEEE Int. Conf. Acoust., Speech. and Signal Process.*, Philadelphia, US, 2005.

[8] R. M. Balboa, C. W. Tyler, and N. M. Grzywacz, "Occlusions contribute to scaling in natural images," *Vision Research*, vol. 41, pp. 955–964, 2001.

[9] N. M. Grzywacz, R. M. Balboa, and C. W. Tyler, "A reply to a letter to the editor by Ruderman," *Vision Research*, vol. 42, pp. 2803–2805, 2002.

[10] D. L. Ruderman, "Origins of scaling in natural images," *Vision Research*, vol. 37, no. 23, pp. 3385–3398, 1997.

[11] J. Gluckman, "Higher order whitening of natural images," *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, no. 12, pp. 354–360, June 2005.

[12] B. Olshausen. [Online]. Available: http://redwood.berkeley.edu/bruno/npb261b/lab2/lab2.html

[13] D. J. Field, "Relations between the statistics of natural images and the response profiles of cortical cells," *Journal of the Optical Society of America A*, pp. 2379–2394, 1987.

[14] L.-Z. Liao, S.-W. Luo, and M. Tian, "Whitenedfaces recognition with PCA and ICA," *IEEE Signal Processing Letters*, vol. 14, no. 12, pp. 1008–1011, December 2007.

[15] J. H. van Hateren and A. van der Schaaf, "Independent component filters of natural images compared with simple cells in primary visual cortex," *Proc. of the Royal Statistical Society of London (B)*, vol. 265, pp. 359–366, 1998.

[16] N. J. W. Morris. S. Avidan, W. Matusik, and H. Pfister, "Statistics of infrared images," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, June 2007.

[17] A. van der Schaaf and J. H. van Hateren, "Modelling the power spectra of natural images: statistics and information," *Vision Research*, vol. 36, pp. 2759–2770, 1996.

[18] M. S. Langer, "Large-scale failures of $f^{-\alpha}$ scaling in natural image spectra," *Journal of the Optical Society of America A*, vol. 17, no. 1, pp. 28–33, 2000.

[19] N. K. Bose, U. Srinivas, and R. L. Culver, "Wavelength diversity based infrared super-resolution and condition-based maintenance," *Insight*, vol. 50, no. 8, pp. 423–428, August 2008.

METODA REDUKCJI NADMIAROWOŚCI W STRUMIENIU DANYCH SEKWENCJI OBRAZÓW

Streszczenie

Sekwencje obrazów zawierają w sposób naturalny dużo strukturalnych powiązań (korelacji), co wiąże się ze znaczną nadmiarowością zawartych w nich informacji. Dla szybkiego przetwarzania i transmisji danych cyfrowych o takich sekwencjach konieczna jest redukcja w nich tych zbytecznych nadmiarowości. W artykule została zaproponowana i dogłębnie przebadana metoda służąca temu celowi. Jest ona, jak pokazano, efektywna. Od strony teoretycznej bazuje na zaproponowanych ulepszeniach, w wielu aspektach, technik dotychczas stosowanych.

Słowa kluczowe: metody redukcji nadmiarowości w strumieniu danych sekwencji obrazów

UNIWERSYTET TECHNOLOGICZNO-PRZYRODNICZY
IM. JANA I JĘDRZEJA ŚNIADECKICH W BYDGOSZCZY
ZESZYTY NAUKOWE NR 253
TELEKOMUNIKACJA I ELEKTRONIKA 12 (2009), 43-56

USING FPGA AND JAVA IN RAPID PROTOTYPING OF A REAL-TIME H.264/AVC DECODER

Marek Parfieniuk[1], Alexey Petrovsky[2], Andrew Stankevich[2],
Michail Kachinsky[2], Alexander Petrovsky[1]

[1] Department of Real-Time Systems
Faculty of Compute Science
Bialystok Technical University
ul. Wiejska 45a, 15-351 Bialystok, Poland

[2] NTLab: New Technologies Laboratory
Minsk, Belarus
www.ntlab-soc.com

Summary: This paper reports on an attempt to implement a real-time hardware H.264 video decoder. The initial results of the project are presented: a customized RISC core and some digital modules, both of which have been implemented in Xilinx FPGA. The former has to serve as a host processor that supervises the latter, which speed up the essential decoding subtasks. The system is designed and tested using a software decoder and diagnostic tools, which are implemented in Java using the object-oriented paradigm. Our experiences allow us to recommend the combination of FPGA and Java technologies as a good basis for rapid prototyping of advanced DSP algorithms.

Keywords: H.264 decoder, Plasma RISC, Xilinx FPGA, Java

1. INTRODUCTION

H.264/AVC (Advanced Video Coding: MPEG-4 Part 10) is the state-of-art standard video codec, which is recommended by both ITU-T and ISO/IEC [3, 6, 8, 15]. Designed to be flexible and network-friendly, it is expected to dominate the market of multimedia devices and services in the near future. The most notable areas where H.264 is used are the Digital Video Broadcasting (DVB), Bluray Disc, 3GPP mobile communication, and video streaming over the Internet.

The main advantages of H.264 over its predecessors, especially over MPEG-2 Video (H.262), are adaptability to various applications and better bandwidth usage. They have been achieved at the price of higher computational complexity and greater memory requirements. Although the main principles of hybrid video coding are still used, the subalgorithms: transform, prediction, motion compensation, as well as entropy coding have been revised and given new options, which improve effectiveness but decrease efficiency. Significant modifications of both bitstream format and decoding

process limit reusing of existing software and hardware, so that new infrastructure needs to be created. Since 2003, when the first version of the standard had been published, many H.264-related products have been issued, like encoding engines, decoder chips, software players, and bitstream analyzers. However, there are still reasons for developing new solutions, because the existing ones often do not follow standard's nuances or recent extensions, or have deficiencies related to speed or power consumption.

As working in this field is interesting from both engineering and commercial points of view, the authors have undertaken the task of developing a real-time hardware H.264 decoder. Its novel architecture has to be not only computationally efficient but also flexible from the design point of view. Namely, its modularization and reconfigurability should guarantee that:

i) future extensions of the standard can easily be incorporated into the system without entirely redesigning it,
ii) speed can be traded off for resource consumption,
iii) customized and optimized digital circuits can be synthesized which satisfy particular application requirements without wasting resources.

An additional decision was to widely use free-of-charge and open-source development tools in order to keep investments small and to be able to customize the toolset in accordance with needs.

Because so far our team mainly specialized in speech processing, see e.g. [1] or [11], in order to gain experience, we have split work between two directions. One is to develop from scratch a reliable object-oriented model of the decoder and to implement it in software. The second one is to design hardware modules using the resulting code as a foundation for FPGA development and using the developed software to generate data for functional verification. In order to achieve high productivity, the Java platform has been used in object-oriented development. In addition to preventing errors that are common in C programming, it was expected to facilitate building a consistent development toolset adjusted to our needs.

The paper presents the initial results of the project, which are related to both software and hardware, and justifies main design decision the authors made. After characterizing H.264 briefly, we describe our software decoder core, applications based on it, and the corresponding hardware architecture. The Plasma-NTLab processor is then presented, which has been developed for the purpose of supervising the prototype platform, including the decoding pipeline. Finally, FPGA designs of some modules that speed up the decoding algorithm are shown. In particular, the transform unit is compared with a known solution, in order to show that our methodology brings benefits.

2. H.264 VIDEO CODEC

The general scheme of the H.264 codec is shown in Fig. 1. Like the older standards, it is a hybrid algorithm that uses transform coding as well as motion-compensated predictive coding to remove both spatial and temporal redundancy of video signals. Better flexibility and compression efficiency have been achieved by only improving subalgorithms [8, 19]. Both fine-grained partitioning of macroblocks into smaller units and quarter-pixel accuracy allow motion estimation/compensation to be more effective. Estimation precision is further improved by employing an in-loop de-

blocking filter which removes the blocking artifact before using a frame for prediction. Temporal redundancy can be better reduced by allowing multiple (up to 16) reference frames to be used and by making bidirectional prediction possible, in which future frames can be referenced in addition to past ones. Removing spatial dependencies among pixels via a decorrelating transform is enhanced by multi-mode intra prediction of a block using adjacent fragments of the same frame. Moreover, transform size can be switched between 4×4 and 8×8 in order to best fit macroblock contents. Finally, more effective methods of entropy coding have been developed: Context-based Adaptive Binary Arithmetic Coding (CABAC) and Context-Adaptive Variable Length Coding (CAVLC) [15]. Specific needs of studio and wireless applications have been satisfied by incorporating Fidelity Range Extensions (FREXt) [8] and Scalable Video Coding (SVC) [18], respectively, into the standard.

Encoders, usually equipped with a lot of switches, allow for customizing output bitstreams in order to best suit a particular usage of the codec. This has been rationalized by defining several H.264 profiles, which correspond to various trade-offs among quality, bitrate, and computational requirements. Owing to these advanced techniques and high flexibility, H.264 offers even two times better compression efficiency than MPEG-2 Video and is much better suited to contemporary applications. However, decoding might require even four times more operations, even though in the new standard, the Discrete Cosine Transform (DCT) has been replaced with efficient multiplierless approximations.

Because of its complexity and little connection to previous standards, implementing H.264 is not trivial, especially if a small and energy-efficient device is required to operate in real time. Thus, there is a great interest in novel solutions in this field.

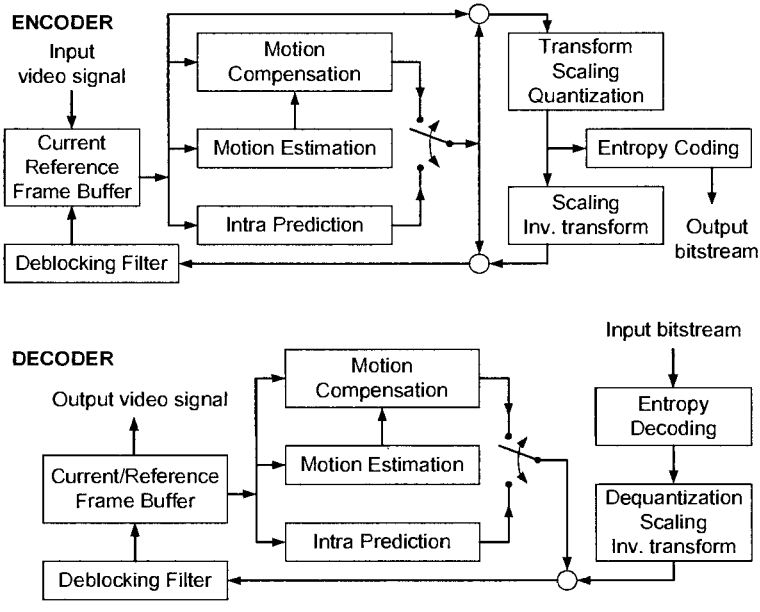


Fig. 1. The general scheme of the H.264 video codec
Rys. 1. Ogólny schemat kodeka wideo standardu H.264

3. SOFTWARE PROTOTYPE

Our review of the existing support for implementing H.264 has shown that a lot of information is accessible but they are distributed among books, papers, web pages, see [3-8, 12, 13, 15-19]. Moreover, they usually are not conveyed readily, so that there are no reliable implementation patterns and ready-to-use high-quality source code. Especially, the standard document and the reference software are very unclear and very difficult to understand even for experienced developers. Thus we decided to design from-the-scratch a software H.264 decoder that does not necessarily work with real-time performance but forms a nice, well-documented foundation for developing and testing hardware modules.

Such objectives have motivated us to employ the Java platform instead of the C or C++ languages, which are commonly used for implementing DSP algorithms [9]. The former is undoubtedly slower but greatly increases productivity and code quality [10]. Its strict type-control prevents many errors that can easily be made when using C, some other bugs become trivial to detect, and finally, the language helps programmers with object-oriented design. Moreover, there is no need for combining different open-source tools/libraries or for relaying on platform-dependent commercial products. The standard Java packages provide all that is necessary for creating advanced GUI- and network-based applications, which work on both Windows and Linux. A rich set of OS-independent development tools can be downloaded as a single bundle, including the sophisticated RAD development environment, NetBeans, and JavaDoc, a simple means for generating well-organized documentation from code comments.

The simplified UML class diagram of our object-oriented model of the H.264 decoder is shown in Fig. 2. It comprises about 60 classes, which represent data and subprocesses related to decoding. They have been designed in such a way that it is easy to identify objects and methods with hardware modules, registers, or state changes. For most of classes, it is possible to strictly determine the number of instances. Knowing the latter allows objects to be preallocated as static fields and to exist continuously during program execution. This significantly reduces computational load related to memory management and garbage collection. It seems that using this technique is crucial for developing a Java-based H.264 decoder that works in real time. Another conclusion, which does not directly result from the standard document, is that most of operations can be performed without explicit integer multiplications. The latter can widely be replaced with binary shifts, possibly supplemented with additions. As to data types, 16 bits (including sign) seem sufficient to store variables related to decoding, but in some cases, auxiliary results need 32 bits. Internal variables of decoding pipelines do not occupy much memory. Quantization tables and sample buffers for transform and prediction purposes take up the most space yet it seems possible to incorporate them into a chip. The main problem is in storing reference frames for inter-prediction, which requires large out-of-chip memory. Some of known decoders require encoders to limit the number of reference frames depending on both video resolution and accessible storage space, and we will probably employ this approach in our chip.

Even though the software decoder is currently developed only in order to help engineers with implementing a hardware analogue, it can become a stand-alone project. Our results suggest that for low resolution videos, real-time performance can be achieved on current PCs even without rewriting the code in the C language.

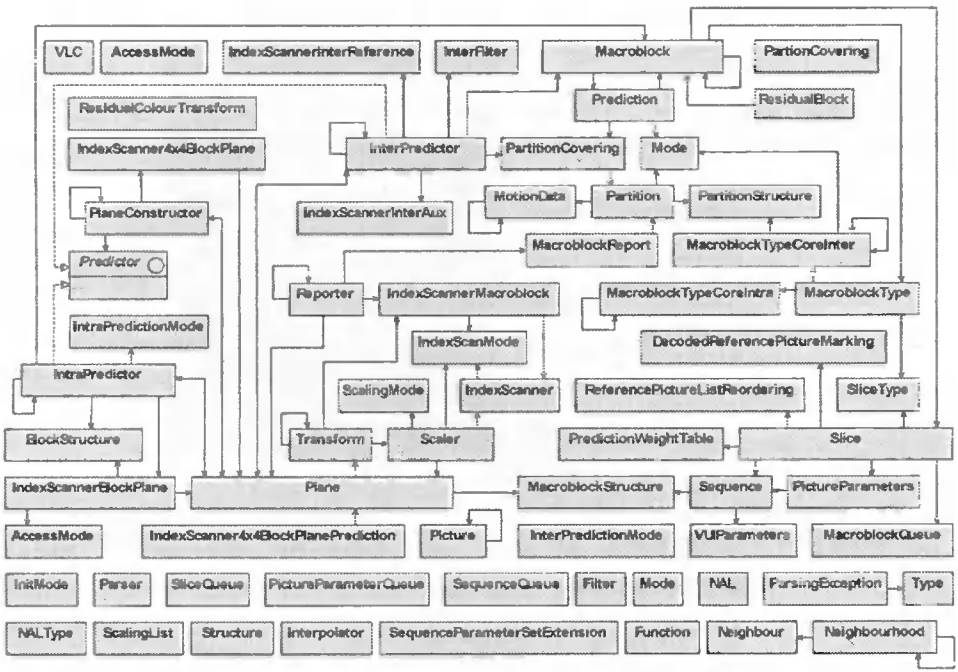


Fig. 2. Simplified UML class diagram of the Java-based H.264 decoder
Rys. 2. Uproszczony diagram UML klas dekodera H.264 zaimplementowanego w Javie

4. DIAGNOSTIC TOOLS

The software decoder is the foundation of our platform-independent diagnostic tool. Being written in Java, the program works in any operating system equipped with the JVM, especially on Linux. It allows for interactively testing the decoder against errors and for preparing data for functional verification of digital modules. This is possible via two main functionalities, which are GUI-controlled using the windows of Fig. 3. Firstly, H.264 bitstreams can be analyzed and restructured, in order to identify and extract input data that cause the decoder to fail. Secondly, the correctness of the decoding of a single frame can be examined both visually and by following the dataflow step by step. The latter is based on a quite advanced reporting mechanism, which collects data in a synthetic form, so that they can be both displayed on screen and exported to verification tools. The mechanism has been designed in a way that allows it to be easily incorporated into the decoder, without refactoring and messing up the essential code.

Similar commercial tools are accessible, e.g. H264Visa [2], but they are quite expensive, work only on Windows, and it is impossible to customize them as desired. Especially, access to the internals of the decoding pipeline is limited, and the data inspected via GUI cannot be efficiently translated to a form suitable for verification. Both drawbacks are addressed in our application. Also, filters are to be developed that allow interesting information to be quickly extracted. Another functionality under development is automatic detection and extraction of erroneously decoded frames of long bit-

streams. Nevertheless, in most cases, interactive testing the program supports is sufficient.

Side-effects of our work are several applications that demonstrate H.264 decoding subalgorithms. For example, Fig. 4 shows the main window of our tool that allows users to interactively study different modes of accessing image samples.

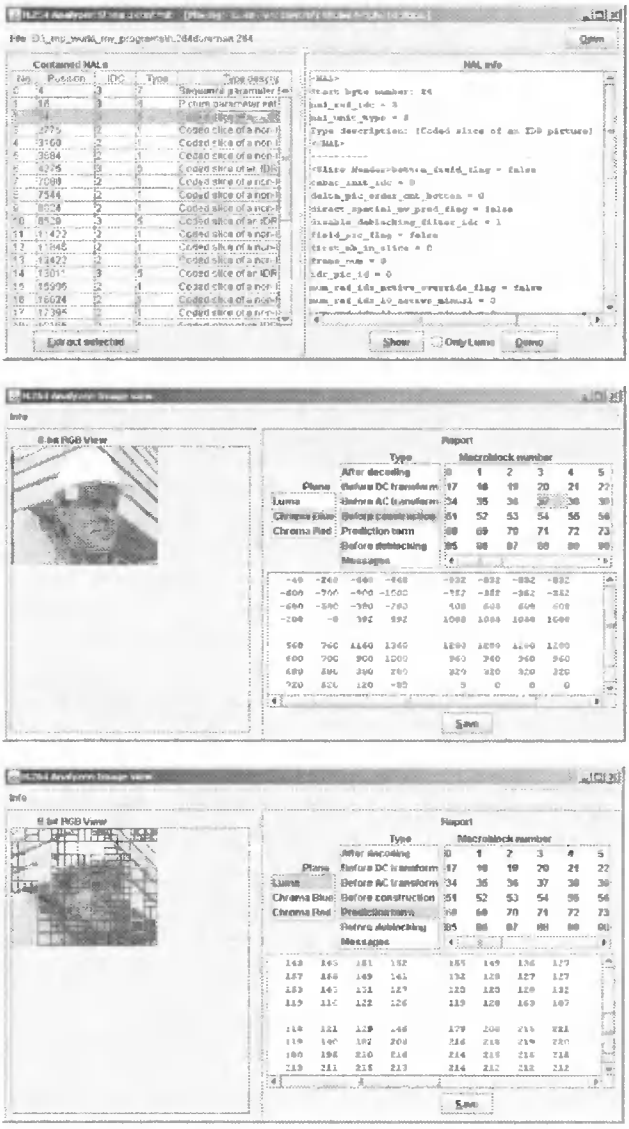


Fig. 3. Diagnostic tool: (a) stream analysis window and frame analysis window for (b) intra- and (c) inter-coding

Rys. 3. Narzędzie diagnostyczne: (a) okno analizy strumienia i okno analizy klatki kodowanej w trybie (b) „intra" i (c) „inter"

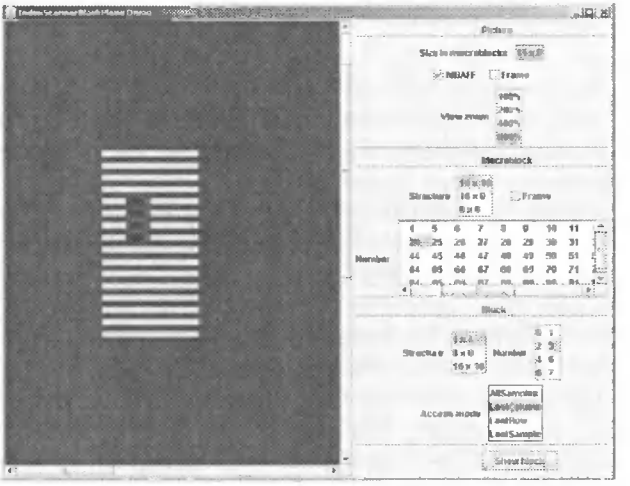


Fig. 4. Tool for demonstrating PAFF and MBAFF modes of image sample access
Rys. 4. Narzędzie do demonstracji trybów PAFF i MBAFF dostępu do próbek obrazu

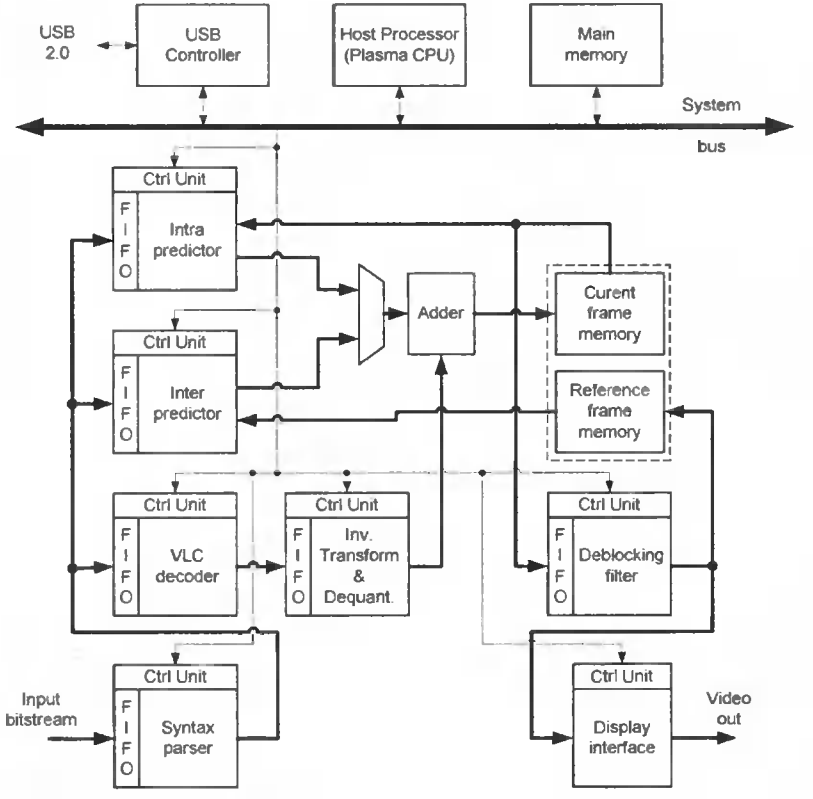


Fig. 5. Hardware decoder architecture
Rys. 5. Architektura dekodera sprzętowego

5. DECODER ARCHITECTURE

Based on the software, we have designed the hardware architecture of Fig. 5. One its part is a host CPU with a USB controller and general-purpose memory. The second one is a H.264 decoding pipeline with dedicated memory for video frames. Except the memories, all circuits have been implemented on a single FPGA chip. After reaching a mature state, the design is expected to be translated to the ASIC technology.

The Virtex-4 ML 401 Evaluation Platform was used in our experiments. It is powered by the Xilinx XC4VLX25 FPGA device and equipped with industry-standard peripherals, interfaces, and connectors like DB15 VGA and USB. The main clock source is a 100 MHz oscillator. The memory resources comprise 64 MB DDR SDRAM, 1 MB ZBT SRAM, 32 MB Compact Flash, 8 MB Flash, 4 kb IIC EEPROM, and 32 Mb Platform Flash. They are connected to the FPGA chip via 32-bit data buses. The main DDR SDRAM runs up to 266 MHz data rate.

Such a configuration is expected to be able to decode videos of resolutions from 320×240 to 720×576 at the rate of 30 frames per second. Both Baseline and Main H.264 profiles have to be supported.

The decoder has been made programmatically reconfigurable. Most of its modules have microprogrammable control units. The host processor is responsible for loading up-to-date microprograms before starting a decoding job. In our architecture, decoding modules are cascaded so that they form a pipeline, in addition to being connected to the system bus. The latter is used only to initialize and roughly control block states. Data related to decoding are passed from module to module via dedicated FIFO-buffered connections between them, which are also responsible for interblock synchronization. This is expected to greatly improve concurrency. Firstly, it helps with avoiding bottlenecks caused by sharing one bus by many devices. Secondly, a connection can be made wide enough to pass an entire 4×4 or 8×8 block of samples at once, which allows them to be processed in parallel. This is especially the case of transform and prediction. The current prototype has only one pipeline which is switched between luminance and chrominance processing. A future approach we consider is to have three separate pipelines, which allows decoding to be totally parallelized but requires a lot of FPGA resources.

6. PLASMA-NTLAB PROCESSOR

The host processor we use is a customized version of the Plasma CPU [14]. The latter is a simple RISC processor accessible as a VHDL project (about 4000 lines of source code + documentation), and thus can be customized and used as a part of advanced SOPC solutions. Moreover, it is free for commercial use, even though its features are sufficient for a wide range of applications, especially for DSP-related ones.

Fig. 6 shows the block diagram of the processor. The 32-bit address bus allows the core to handle large memory. Excessive accesses to the latter can be avoided by wise use of 32 32-bit general-purpose registers. Two additional special-purpose registers for storing the results of both integer multiplication and division allow the ALU to fully support fixed-point arithmetic. At the VHDL level, it is possible to select between big- and little-endian byte-ordering. A number of peripherals are also accessible: UART,

Interrupt Controller, Interrupt Timer, SRAM Controller, Flash Controller, DDR SDRAM Controller, and Ethernet MAC. The instruction set is compatible with the MIPS I architecture, e.g. with the MIPS R2000 CPU. From another point of view, it is equivalent to the user-mode subset of the MIPS32 instructions, except nonaligned data access and exceptions. The two-stage command pipeline can be extended to three stages.

The Plasma core is not very resource consuming. If Xilinx Spartan-3 XC3S1000 is the target platform, it takes up 1604 slices (20% of chip area) and can operate at the maximum clock of 32–33 MHz. For Xilinx Virtex-4 XC4VLX25, it takes up 1588 slices (14% chip area), whereas the maximum clock is 64–67 MHz.

The original Plasma has been customized in order to match the memory organization and I/O interfaces the development board provides. Especially, a USB controller has been added, which allows for communication between the system and a PC workstation. The testing environment runs on the latter, which allows for sending a video stream to the prototype decoder and for verifying the output. This required the system to be extended in such a way that the CPU can control and monitor stages of the decoding pipeline. The resulting microprocessor architecture has been called the Plasma-NTLab CPU.

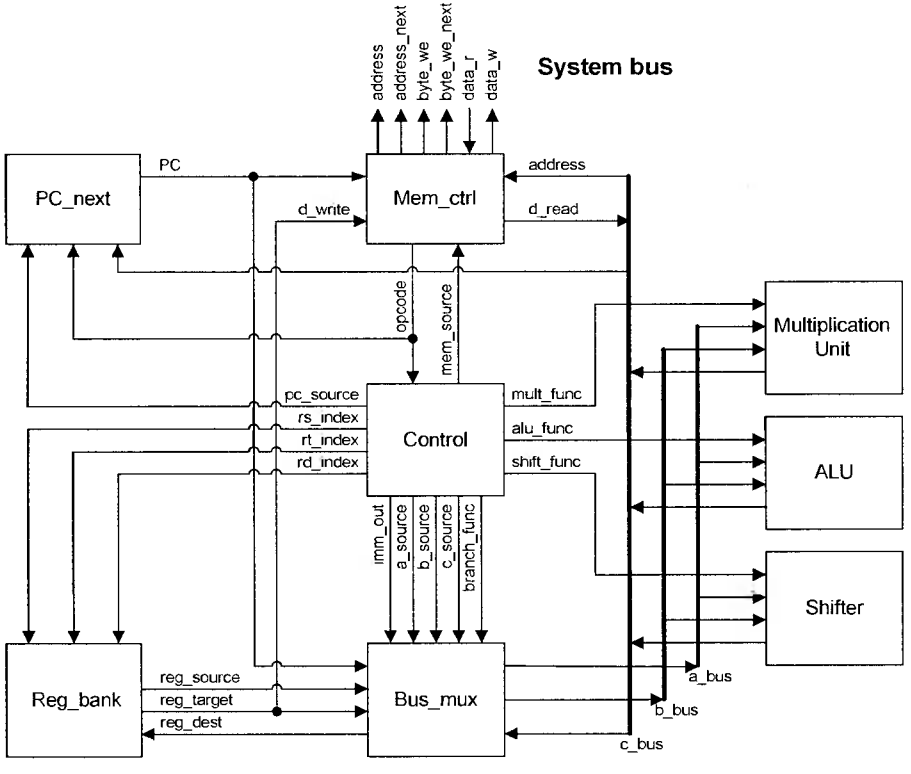


Fig. 6. The block diagram of the Plasma microprocessor
Rys. 6. Diagram blokowy mikroprocesora Plasma

7. FPGA DESIGN OF DECODING MODULES

For those parts of the software decoder for which code had been frozen after thorough tests, the corresponding digital circuits have been developed in FPGA. These are the NALU (Network Abstraction Layer Unit) detector, bitstream parser, including the VLC (Variable Length Coding) decoder, and residual transform unit. Their schemes are shown in Figs. 7, 8, and 9, respectively.

The first module is responsible for determining NALU boundaries in a input bitstream and for extracting the contents units carry. The contents form a higher-level bitstream, which is analyzed by the parameter parser in order to decode syntax elements. The related operations require only iterating binary shifts and comparisons, which seems simple, but takes many cycles of a general-purpose CPU. In order to relieve the latter, only a fraction of the FPGA chip area needs to be sacrificed. Namely, the NALU detector of Fig. 7 uses only 70 of 21504 Slice Flip-Flops and 174 of 21504 LUTs that XC4VLX25 contains.

A much more complex part of the decoder is the transform unit of Fig. 9, which computes approximations of 2-dimensional DCTs. It needs quite large memory buffers for storing two 16 × 16 arrays of integers: one of input data and one of auxiliary/output values. The 4 × 4 version of our transform unit can be compared with that of [5], where detailed information of a H.264 implementation in FPGA is given. Two variants of the circuit are considered therein, which we have also realized in our architecture. In the first, calculations on vectors are performed element by element, in order to conserve chip area. In the second, an entire inner product of 4-element vectors is computed at once, which requires replicated arithmetic blocks to work in parallel. The synthesis results of Tables 1 and 2 show that our preliminary designs are comparable to those by others, or even slightly better in terms of chip area utilization. This proves that our software prototype well accomplishes its task. It allows hardware engineers to quickly understand what is expected and to construct digital circuits of good quality, i.e. high performance is achieved at low resource utilization. The inter- and intra-prediction units are under development. Finishing them will allow a first version of the decoder to be assembled.

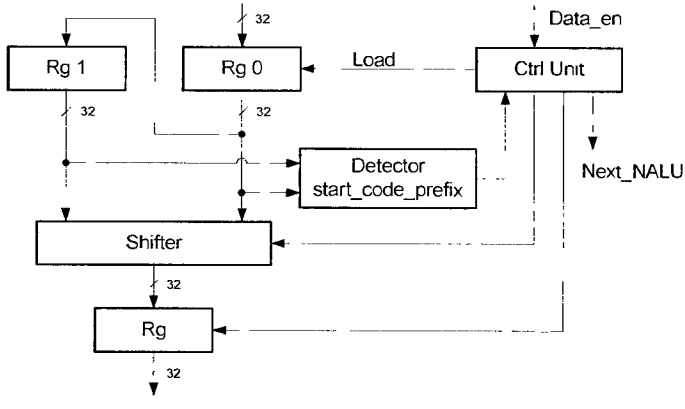


Fig. 7. NALU detector
Rys. 7. Detektor jednostek NALU w wejściowym strumieniu binarnym

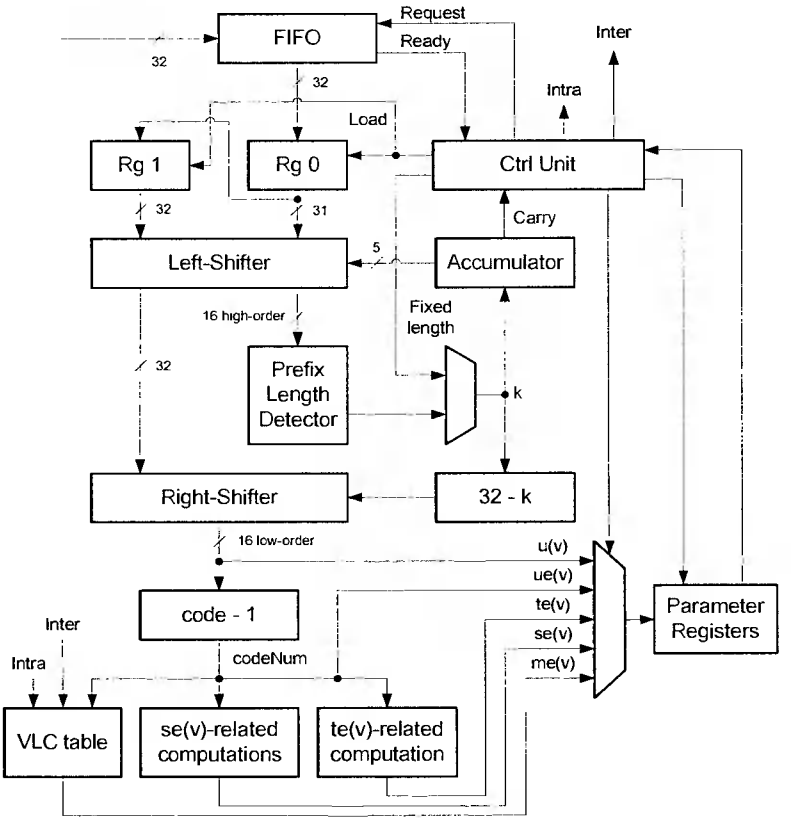


Fig. 8. Parser unit
Rys. 8. Parser

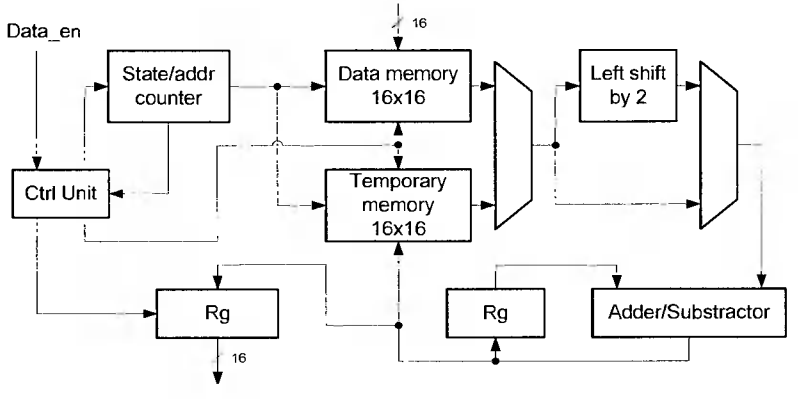


Fig. 9. Transform unit
Rys. 9. Moduł transformacji rezyduum

Table 1. Evaluation of single-stage residual transform
Tabela 1. Ocena jednostopniowej implementacji transformacji rezyduum

Parameter	Authors' realization	[5]
FPGA chip	XC4VLX25-12	XC2VP7
Wordlength	16	16
Slice Flip-Flop	37	65
LUT	111	*
Slice	93	103
Max. clock [MHz]	200	150

Table 2. Evaluation of parallel residual transform
Tabela 2. Ocena równoległej implementacji transformacji rezyduum

Parameter	Authors' realization	[5]
FPGA chip	XC4VLX25-12	XC2VP7
Wordlength	16	16
Slice Flip-Flop	-	257
LUT	1008	*
Slice	512	644
Max. critical delay [ns]	9.7	9.3

8. CONCLUSION

Using Java routines as a basis for FPGA development turns out to be a good methodology for implementing such an advanced DSP algorithm as the H.264 decoder. Clear and well-documented code has allowed hardware specialists both to design a flexible modularized architecture of the real-time system and to rapidly prototype digital circuits that speed up decoding subtasks. The advantage of high productivity is accompanied by good quality of FPGA designs, which can compete with the solutions by others, in terms of throughput and resource utilization. Additionally, without the necessity of looking for other tools, the Java platform has allowed us to develop in parallel a multi-platform GUI-based diagnostic application for test and verification purposes. On the other hand, the simple Plasma RISC core, which is publicly available as a VHDL source code, has served as a foundation for developing a customized host processor for controlling the hardware decoding pipeline.

BIBLIOGRAPHY

[1] A. Borowicz, M. Parfieniuk, A. Petrovsky, 2006: An application of the warped discrete Fourier transform in the perceptual speech enhancement. Speech Comm., vol. 48, pp. 1024-1036.
[2] H264Visa (online). Available: http://www.h264visa.com
[3] ITU-T and ISO/IEC, 2003: ITU-T Rec. H.264 Advanced video coding for generic audiovisual services / ISO/IEC 14496-10 MPEG-4 AVC. Geneva (online). Available: http://www.itu.int/rec/T-REC-H.264/
[4] ITU-T and ISO/IEC, 2001: ITU-T Rec. H.264.2 Reference software for H.264 advanced video coding / ISO/IEC 14496-5 MPEG-4 Reference software. Geneva.

[5] R. Kordasiewicz, S. Shirani, 2006: On hardware implementations of DCT and quantization blocks for H.264/AVC. J. VLSI Signal Process., vol. 47, pp. 189-199.
[6] D. Marpe, T. Wiegand, G. J. Sullivan, 2006: The H.264/MPEG4 Advanced Video Coding standard and its applications. IEEE Commun. Mag., pp. 134-143.
[7] C.S. Kannangara, 2006: Complexity management of H.264/AVC video compression. PhD Thesis, The Robert Gordon University.
[8] S.-k. Kwon, A. Tamhankar, K.R. Rao, 2006: Overview of H.264/MPEG-4 part 10. J. Vis. Commun. Image R., vol. 17, pp. 186-216.
[9] J. Labrosse et al., 2008: Embedded software: know it all. Newnes, Oxford.
[10] C.D. Locke, P.C. Dibble, 2003: Java technology comes to real-time applications. Proc. IEEE, vol. 91, no. 7, pp. 1105-1113.
[11] M. Livshitz, M. Parfieniuk, A. Petrovsky, 2005: Wideband CELP coder with multiband excitation and multilevel vector quantization based on reconfigurable codebook, Digital Signal Process. no. 2, pp. 20-35, Moscow, Russia (in Russian).
[12] Mailing list for x246 developers (online). Available: http://mailman.videolan.org/listinfo/x264-devel
[13] Mp4-tech mailing list (online). Available: http://lists.mpegif.org/mailman/list-info/mp4-tech
[14] Plasma CPU (online). Available: http://www.opencores.org/projects/mips
[15] E. G. Richardson, 2003: H.264 and MPEG-4 Video Compression. Wiley, New York.
[16] The FFmpeg libavcodec library (online). Available: http://ffmpeg.org
[17] The H.264/AVC reference software (JM) (online). Available: http://iphome.hhi.de/suehring/tml/
[18] H. Schwarz, D. Marpe, T. Wiegand, 2007: Overview of the scalable video coding extension of H.264/AVC. IEEE Trans. Circuits Syst. Video Technol., vol. 17, no. 9, pp. 1103-1120.
[19] T. Wiegand et al., 2003: Overview of the H.264/AVC video coding standard. IEEE Trans. Circuits Syst. Video Technol, vol. 13, no. 7, pp. 560-576.

This work was supported by Bialystok Technical University under the grants W/WI/8/2008 and W/WI/6/2009.

UŻYCIE FPGA I JAVA DO SZYBKIEGO PROTOTYPOWANIA DEKODERA H.264/AVC DZIAŁAJĄCEGO W CZASIE RZECZYWISTYM

Streszczenie

W pracy przedstawiono raport z próby implementacji działającego w czasie rzeczywistym sprzętowego dekodera wideo standardu H.264. Zaprezentowano wstępne wyniki projektu: jądro RISC i wybrane moduły cyfrowe zaimplementowane z użyciem Xilinx FPGA. Jądro ma służyć jako nadrzędny procesor sterujący pozostałymi obwodami dekodera, które przyśpieszają podstawowe etapy dekodowania. System jest projektowany i testowany w oparciu o dekoder programowy i narzędzia diagnostyczne, które są implementowane obiektowo w Javie. Uzyskane rezultaty pozwalają autorom rekomendować połączenie FPGA i Java jako dobrą podstawę do szybkiego prototypowania zaawansowanych algorytmów DSP.

Słowa kluczowe: dekoder H.264, Plasma RISC, Xilinx FPGA, Java

UNIWERSYTET TECHNOLOGICZNO-PRZYRODNICZY
IM. JANA I JĘDRZEJA ŚNIADECKICH W BYDGOSZCZY
ZESZYTY NAUKOWE NR 253
TELEKOMUNIKACJA I ELEKTRONIKA 12 (2009), 57-71

PARALLEL 4X4 TRANSFORM ON BIT – SERIAL SHARED MEMORY ARCHITECTURE FOR H.264/AVC

Grzegorz Rubin

Department of Computer Science
Bialystok University of Technology
Bialystok, Poland
gregor@wi.pb.edu.pl

Summary. The aim of this paper is to present an implementation and simulation of parallel 4x4 transform on bit-serial shared memory architecture for H.264/AVC. Compared with the existing parallel implementations, the proposed architecture reduces interconnection resources of physical elements of FPGA device. The results of simulation show that the transform can be realized in real–time on bit–serial arithmetic. The paper concludes with a summary.

Keywords: FPGA, shared memory, video coding

1. INTRODUCTION

H.264/AVC is the latest standard of video coding for applications used in mobile devices. There are few known ways of its implementations [1, 2, 3]. Reduction of power consumption or increasing an image quality and real-time processing are the main goals for portable multimedia devices. Therefore, H.264/AVC can be used for video coding in wireless video applications if such requirements will be fulfilled.

Long-range bit-parallel data links provide high data rates at the cost of large chip area, routing difficulty, noise and power [4]. Additionally, such links are often utilized only a small portion of the time, but dissipate leakage power at all times. Parallel link performance is bounded by available clock rate and by clock skew, delay uncertainty due to process variations, cross-talk noise, and layout geometries. There is an alternative to bit-parallel interconnects, mitigating the issues of area and leakage power, when bit-serial few wires are used. However, to provide the same throughput as an N-bit parallel interconnect, the serial link must operate N times faster. For bit-serial implementation, H.264/AVC has relatively higher complexity than other video standards, which might result in increase of power consumption and difficulty of real-time processing.

In this paper, a new parallel 4x4 transform architecture based on bit-serial shared memory architecture is proposed to improve processing rate for H.264/AVC and reduce power consumption. This paper is organized as follows. H.264 transform algorithms are described in section 2. The new architecture and scheduling method by special toolbox

are presented in section 3. Simulation and implementation results are shown in section 4. Finally, conclusion is given in section 5.

2. 2-D INTEGER TRANSFORM IN H.264

The H.264/AVC standard is known as: ISOMPEG4 Part 10, ITU-T H.264, and the Advanced Video Coding (AVC)[5]. This paper discusses the Residual Transform (RT) block, although the standard specifies a complete decoder. Both, the encoder and the decoder use this block, and perform a transform on a macro block level. The transform block in H.264/AVC is one of the key components and there are several aspects of its design that are considered[1]. First, the transform is orthogonal, separable, low gain, and has a strong decorrelating performance. Second, the transforms are performed in integer arithmetic, because floating point arithmetic is harder to implement in hardware. Third, the transform block is designed to reduce the memory access and computation overhead. There are three different transforms used in H.264/AVC, one for 4x4 DC luma coefficients, another for 2x2 chroma DC coefficients and a third for all other 4x4 residual data [3]. It has been shown through complexity analysis [4] that the 4x4 residual data transform takes up majority of computation, therefore it will be used as an example in this paper. However, the proposed architecture can be used to implement other transforms. The ability of adapting proposed designs to incorporate other calculations will be discussed. The authors of H.264/AVC start with a well known two dimensional Discrete Cosine Transform (DCT). This transform can be represented by:

$$Y = AXA^t = \begin{bmatrix} a & a & a & a \\ b & c & -c & -b \\ a & -a & -a & a \\ c & -b & b & -c \end{bmatrix} [X] \begin{bmatrix} a & b & a & c \\ a & c & -a & -b \\ a & -c & -a & b \\ a & -b & a & -c \end{bmatrix} \tag{1}$$

where:

$$a = \frac{1}{2}, b = \sqrt{\frac{1}{2}}\cos\left(\frac{\pi}{8}\right), c = \sqrt{\frac{1}{2}}\cos\left(\frac{3\pi}{8}\right)$$

Then matrix equation (1) can be factorized into equivalent form:

$$Y = \left(CXC^T\right) \otimes E = \left(\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & d & -d & -1 \\ 1 & -1 & -1 & 1 \\ d & -1 & 1 & -d \end{bmatrix} [X] \begin{bmatrix} 1 & 1 & 1 & d \\ 1 & d & -1 & -1 \\ 1 & -d & -1 & 1 \\ 1 & -1 & 1 & -d \end{bmatrix} \right) \otimes \begin{bmatrix} a^2 & ab & a^2 & ab \\ ab & b^2 & ab & b^2 \\ a^2 & ab & a^2 & ab \\ ab & b^2 & ab & b^2 \end{bmatrix} \tag{2}$$

Where E is matrix scaling factors and the symbol $\otimes$ indicates that each element of CXC^T is multiplied by the scaling factor in the same position in matrix E. Values a and b are the same as listed in (1) and d=c/b.

More simply implementation of the transform can be done, when we modify constants like this:

$$a=\frac{1}{2}, b=\sqrt{\frac{2}{5}}, d=\frac{1}{2}$$

Therefore final forward transform becomes:

$$Y=\left(CXC^{T}\right)\otimes E=\left(\begin{bmatrix}1 & 1 & 1 & 1\\ 2 & 1 & -1 & 2\\ 1 & -1 & -1 & 1\\ 1 & -2 & 2 & -1\end{bmatrix}[X]\begin{bmatrix}1 & 2 & 1 & 1\\ 1 & 1 & -1 & -2\\ 1 & -1 & -1 & 2\\ 1 & -2 & 1 & -1\end{bmatrix}\right)\otimes\begin{bmatrix}a^2 & \frac{ab}{2} & a^2 & \frac{ab}{2}\\ \frac{ab}{2} & \frac{b^2}{4} & \frac{ab}{2} & \frac{b^2}{4}\\ a^2 & \frac{ab}{2} & a^2 & \frac{ab}{2}\\ \frac{ab}{2} & \frac{b^2}{4} & \frac{ab}{2} & \frac{b^2}{4}\end{bmatrix} \quad (3)$$

CXC^T part is the core of transform and can be carried out with integer arithmetic using only additions, subtractions and shifts. The last operation $\otimes E$ requires multiplication, which can be done into the quantization process.

Several hardware design methods for the implementation of the 2-D integer transform have been developed in recent years. In this paper an architecture which performs one dimensional transform on a column of input data by a matrix multiplication architecture was used [2] as shown in Fig. 2. Figure 1 and formula (4) show 1-D transform and that operation needs to be performed four times along the vertical dimension and four times along the horizontal dimension on X. Each of these eight 1-D column/row transforms requires four adders and four subtractors. In the paper [3] with this novel architecture it is possible to perform the whole DCT in one cycle, where the cycle duration is close to a carry propagation through a 64–bit adder. The large amount of hardware used is the drawback of this design, which totals 32 16–bit adders and 32 16–bit subtractors. This amount of hardware can be reduced by introducing various pipeline splits. An addition of a register stage in the middle of the combinational. This papers based on given in figure 2 transform what is detailed in[3], but hardware implementation uses shared memory architecture approach. Moreover, proposed approach uses bit-serial arithmetic and synchronous calculations.

$$\begin{aligned}
x_0' &= (x_0+x_3)+(x_1+x_2),\\
x_1' &= (x_0-x_3)\cdot 2+(x_1-x_2),\\
x_2' &= (x_0+x_3)-(x_1+x_2),\\
x_3' &= (x_0-x_3)+(x_1-x_2)\cdot 2,
\end{aligned} \quad (4)$$

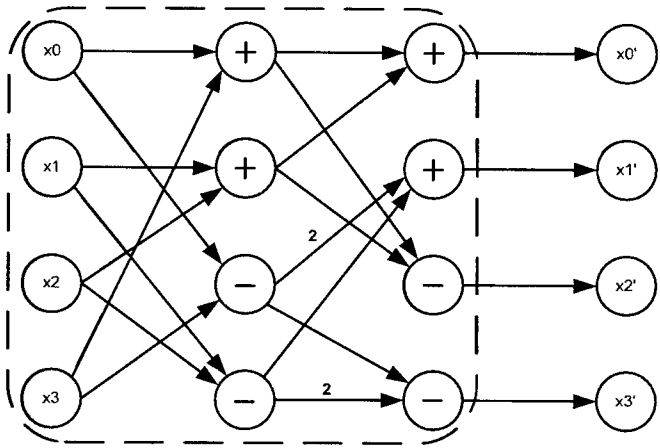


Fig. 1. 1-D transform
Rys. 1. Przekształcenie 1-wymiarowe

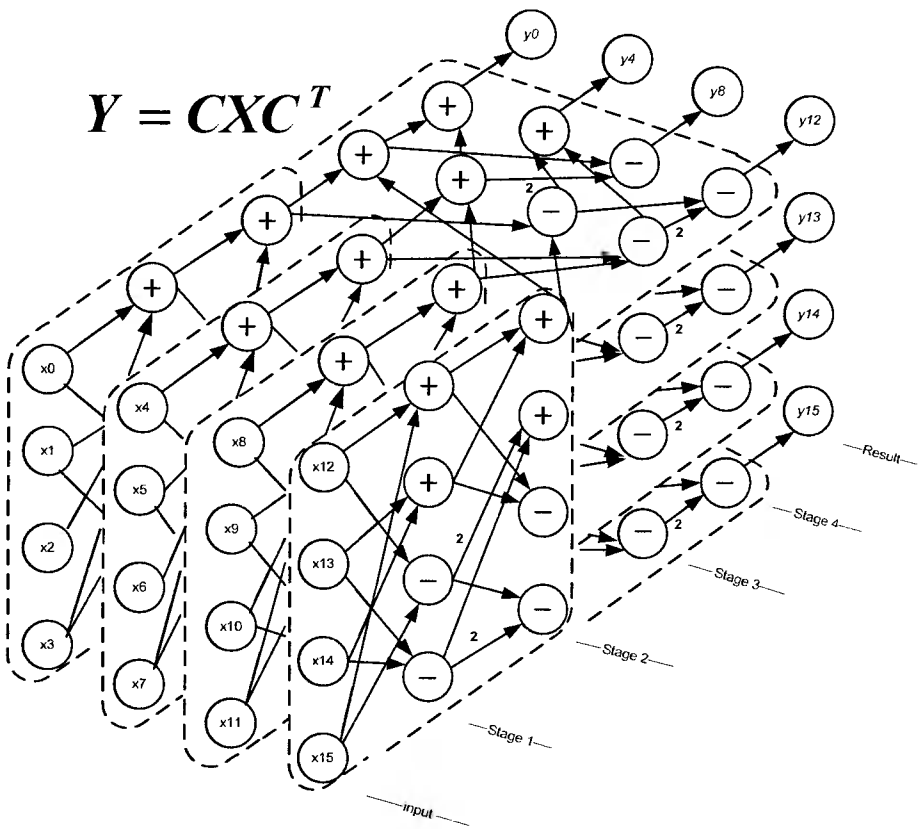


Fig. 2. Parallel 2-D transform
Rys. 2. Równoległe przekształcenie 2-wymiarowe

3. PROPOSED BIT-SERIAL ARCHITECTURE

In the following subsections two usages of hardware architectures of the transform block are presented. The first set of designs is concentrated on unit of bit-serial shared memory architecture, and the second set of designs concentrate on parallel usage of bit–serial blocks. In addition, the description of the two approaches could be useful for designing other area and timing optimized designs.

3.1. SHARED MEMORY ARCHITECTURE

Shared-memory architecture approach (SMA) is detailed in [6]. Figure 3 shows proposition of that architecture. The idea is very simple. In order to simultaneously provide the PEs (Processing Elements) with input data, the shared memory is partitioned into blocks. PEs usually perform simple memoryless mapping of the input values to a single output values. Using a rotating access scheme, each processor gets access to the memories once per N (N – number of PE's) cycles. During this time, processor either writes or reads data from memory. All processors have the same duration time slot to access to the memories and access conflict is completely avoided. Examples of usage and algorithm implementations can be found in [7]. Presented paper presents some modifications of architecture. Separate input and output registers were replaced by registers with input/output interface. There is possible to save result of serial calculation into free cells of input registers during shifting data into processing elements. The second modification is 16–bit width of data extension.

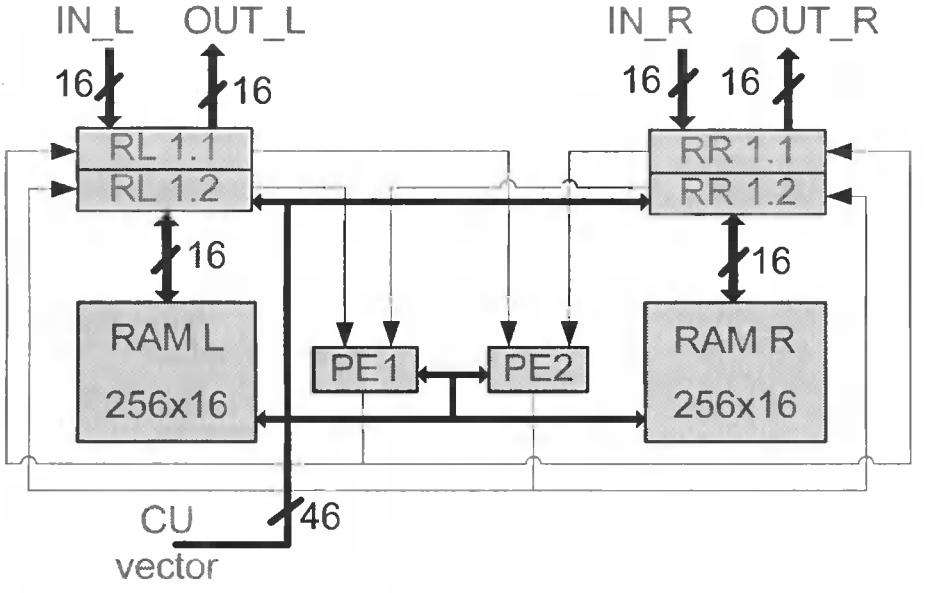


Fig. 3. Shared Memory Architecture unit
Rys. 3. Jednostka Architektury o współdzielonej pamięci

3.2. PARALLEL FORM OF SMA

One SMA unit can calculate the algorithm witch depends on "CU vector" bus state. Implementation of integer residual transform on one SMA requires additions and multiplications. Because of multiplier value on bit-serial arithmetic shifting can be replaced against multiplication. For each block of the 2-D 4x4 integer transform the same calculations has to be done. Therefore, we can repeat the same operations eight times on the same SMA unit or take parallel connection form of few identical SMA blocs with the same dynamic instructions of scheduling (Fig.4). Note that, to save input pins in physical device we must share input lines for data and control vectors. The input data are coming in 16-bit parallel form step by step, but calculations in 16-bit serial form. H.264/AVC supports 8-bit residual pixel data, but it is likely that this will be extended in the future [2]. The time period between two next data depends on bit-serial calculations. Proposed approach of calculations takes 879 steps for one block. During that we can use input pins for data of next 4x4 block, but we can't use input of control vector. Otherwise the control vector is the same for each block, but if we want to use it for parallel calculations, delay by few steps should be applied according to proper input data. One of possibilities based on additional instruction register for every SMA block. Then delayed input of control vector can be applied and the additional set of data calculated.

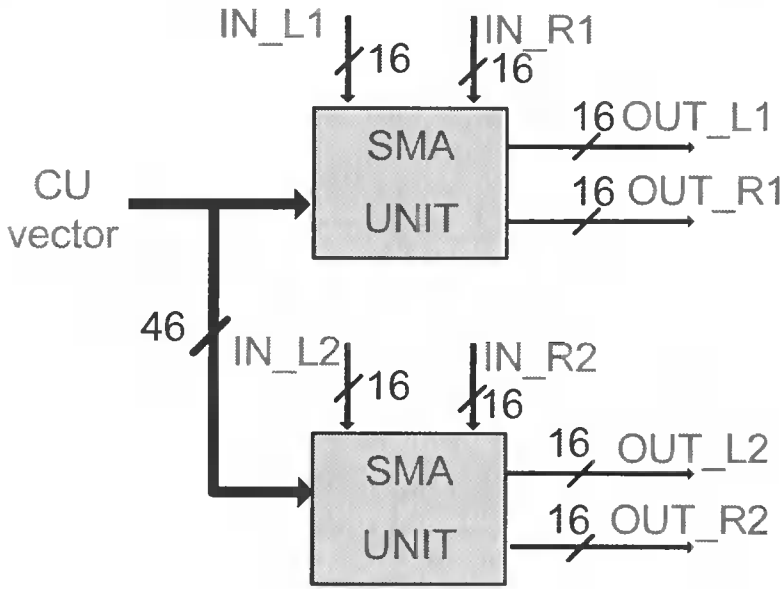


Fig. 4. Two parallel SMA units in the same operating mode
Rys. 4. Dwie równoległe jednostki SMA w jednakowym trybie pracy

We can use set of SMA blocks, but note that, the number of blocks can't exceed resources of physical device and steps of calculation of the first SMA unit. Figure 5 shows proposition of three SMA connected architecture.

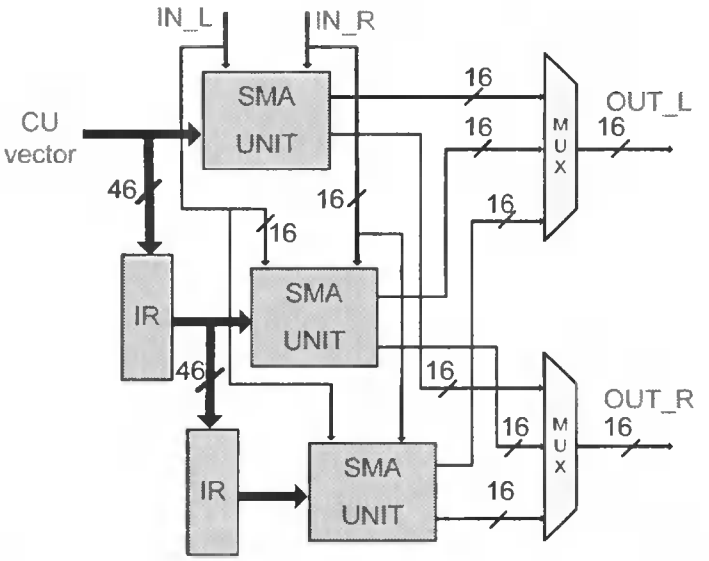


Fig. 5. Three parallel SMA units in the same operating mode with sharing input/ output pins

Rys. 5. Trzy równoległe jednostki SMA w jednakowym trybie pracy ze współdzielonymi pinami wejścia/wyjścia

3.3. SMA TOOLBOX FOR CONTROL UNIT

For fast and efficient scheduling of SMA elements an application was designed. Using that tool the designer can very quickly generate control unit vectors for any algorithm realized on proposed SMA architecture. That's very helpful during simulation process, because of width of 46-bit vector. It's a big probability of setting wrong values into the control vectors witch enables and runs elements of SMA architecture, especially in bit serial arithmetic. Proposed architecture was designed that it's possible to run every part inside the SMA in any time period, therefore the user decide of proper scheduling of algorithm realization. Such architectures are dynamic reconfigurable, so modification of control vector can change scheduling of SMA unit and another algorithm can be realized without any physical changes. The interface of SMA toolbox is presented on figure 6. Upper right corner consist scheme of used architecture for scheduling. Chosing „operation" IN or OUT the left side boxes are enabled or disabled, then DATA IN or DATA OUT options are available to set. There is also possibility to load or save the results of scheduling.

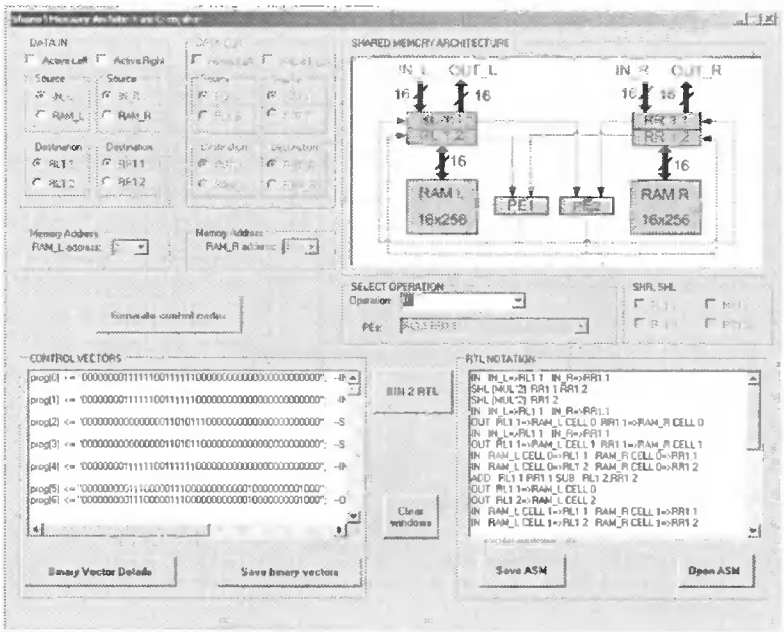


Fig. 6. SMA compiler interface
Rys. 6. Interfejs kompilatora SMA

Proposed SMA compiler is very simple at this time. There are two ways of programming required control vectors – by setting each step using buttons and checkboxes or writing by text in RTL notation. Then just clicking "Generate control vector" in the control vectors window we can see expected set of bits. For example input two set of data, add them together and put it out, RTL notation is as follow:

```
IN    IN_L=>RL1.1    IN_R=>RR1.1
ADD   RL1.1,RR1.1
OUT   RL1.1=>OUT_L   RR1.1=>OUT_R
```

and control vector listed bellow:

```
prog(0)  <= "00000000111111001111110000000000000000000000000";
--IN    IN_L=>RL1.1    IN_R=>RR1.1
prog(1)  <= "10000000000000000000000000000000000000000000000";
--ADD   RL1.1, RR1.1
prog(2)  <= "00100000010000010100000100000000000000000000000";
prog(3)  <= "00100000010000010100000100000000000000000000000";
prog(4)  <= "00100000010000010100000100000000000000000000000";
prog(5)  <= "00100000010000010100000100000000000000000000000";
prog(6)  <= "00100000010000010100000100000000000000000000000";
prog(7)  <= "00100000010000010100000100000000000000000000000";
prog(8)  <= "00100000010000010100000100000000000000000000000";
prog(9)  <= "00100000010000010100000100000000000000000000000";
```

```
prog(10) <= "00100000010000010100000100000000000000000000";
prog(11) <= "00100000010000010100000100000000000000000000";
prog(12) <= "00100000010000010100000100000000000000000000";
prog(13) <= "00100000010000010100000100000000000000000000";
prog(14) <= "00100000010000010100000100000000000000000000";
prog(15) <= "00100000010000010100000100000000000000000000";
prog(16) <= "00100000010000010100000100000000000000000000";
prog(17) <= "00100000010000010100000100000000000000000000";
prog(18) <= "00100000010000010100000100000000000000000000";
prog(19) <= "00000000111111011111110100000000000000000000";
--OUT  RL1.1=>OUT_L  RR1.1=>OUT_R
```

Given example shows that we have strict rules of Processing Elements usage. They can work together at the same time or one do nothing. That limitations are only because of application and SMA architecture can do independent operations, but scheduling must be done manually.

3.4. CONTROL VECTORS FOR 4-POINT 2-D TRANSFORM

For proper work of any algorithm, set of control vectors are required. Using SMA compiler it's possible to generate set of bit vectors for 4-point 2-D transform presented in Figure 2. The SMA unit is able to calculate arithmetic operations such as adding, subtracting, multiply and negation. Additionally it can store some temporary results of any calculation using two separate memory blocks. Therefore SMA unit can by applied for any algorithm using proper control vectors. Because of paper space only the RTL notation for input data and first part of calculations for residual transform is given bellow:

- read input data

```
IN IN_L=>RL1.1 IN_R=>RR1.1
```

- save input data into RAM L and R - address 0

```
OUT RL1.1=>RAM_L CELL 0 RR1.1=>RAM_R CELL 0
```

- read input data

```
IN IN_L=>RL1.1 IN_R=>RR1.1
```

- save input data into RAM L and R - address 1

```
OUT RL1.1=>RAM_L CELL 1 RR1.1=>RAM_R CELL 1
```

- read input data

```
IN IN_L=>RL1.1 IN_R=>RR1.1
```

- save input data into RAM L and R - address 2

```
OUT RL1.1=>RAM_L CELL 2 RR1.1=>RAM_R CELL 2
```

- read input data

```
IN IN_L=>RL1.1 IN_R=>RR1.1
```

- save input data into RAM L and R - address 3

```
OUT RL1.1=>RAM_L CELL 3 RR1.1=>RAM_R CELL 3
```

- read input data

```
IN IN_L=>RL1.1 IN_R=>RR1.1
```

- save input data into RAM L and R - address 4

```
OUT RL1.1=>RAM_L CELL 4 RR1.1=>RAM_R CELL 4
```

- read input data

```
IN IN_L=>RL1.1 IN_R=>RR1.1
```

- save input data into RAM L and R - address 5

```
OUT RL1.1=>RAM_L CELL 5 RR1.1=>RAM_R CELL 5
```

```
- read input data
  IN IN_L=>RL1.1 IN_R=>RR1.1
- save input data into RAM L and R - address 6
  OUT RL1.1=>RAM_L CELL 6 RR1.1=>RAM_R CELL 6
- read input data
  IN IN_L=>RL1.1 IN_R=>RR1.1
- save input data into RAM L and R - address 7
  OUT RL1.1=>RAM_L CELL 7 RR1.1=>RAM_R CELL 7

- the first 1-D transform
  IN RAM_L CELL 0=>RL1.1 RAM_R CELL 0=>RR1.1
  IN RAM_L CELL 0=>RL1.2 RAM_R CELL 0=>RR1.2

  ADD RL1.1,RR1.1 SUB RL1.2,RR1.2
  OUT RL1.1=>RAM_L CELL 8
  OUT RL1.2=>RAM_L CELL 9
  IN RAM_L CELL 1=>RL1.1 RAM_R CELL 1=>RR1.1
  IN RAM_L CELL 1=>RL1.2 RAM_R CELL 1=>RR1.2
  ADD RL1.1,RR1.1 SUB RL1.2,RR1.2
  OUT RR1.1=>RAM_R CELL 8
  OUT RR1.2=>RAM_R CELL 9
  IN RAM_L CELL 8=>RL1.1 RAM_R CELL 8=>RR1.1
  IN RAM_L CELL 8=>RL1.2 RAM_R CELL 8=>RR1.2
  ADD RL1.1,RR1.1 SUB RL1.2,RR1.2
  OUT RL1.1=>RAM_L CELL 10
  OUT RL1.2=>RAM_L CELL 11
  IN RAM_L CELL 9=>RL1.1 RAM_R CELL 9=>RR1.1
  SHL (MUL*2) RL1.1
  IN RAM_L CELL 9=>RL1.2 RAM_R CELL 9=>RR1.2
  SHL (MUL*2) RL1.2
  ADD RL1.1,RR1.1SUB RL1.2,RR1.2
  OUT RL1.1=>RAM_L CELL 12
  OUT RL1.2=>RAM_L CELL 13
```

Presented transformation uses 8 identical programmed operations (1-D residual transform), only memory addresses for temporary data differs. One operations takes 113 synchronous steps for 16-bit serial arithmetic. In this paper proposed approach uses one SMA unit for one 4x4 set of data and calculates coefficient during 879 synchronous steps. As was written previously for the test, two parallel connected SMA units were used in Xilinx ISE simulator. Both have the same control vector and separate input/output pins, so the units work together for two sets of data simultaneous. Proposed SMA units can be connected in cascade or parallel form and number of units is limited only by physical programmable device. Such approach allows for better scheduling of algorithm, for example simulated transformation can be calculated by eight SMA units for one 4x4 block. According to Figure 2, firstly compute 4 of 1-D transformation in parallel form, secondly next 4 computation. Then the result will be after 226 steps. Flexible scheduling allows for many combinations of calculations.

3.5. SCHEDULING USING PETRI NETS

Scheduling of parallel processes can be done based on Petri Nets theory. There are few know examples of that approach [8, 9, 10]. For proposed approach based on shared memory architecture an application was desing. Presented environment allows on graphical designing and simulation of algorithms step by step. Formal analysis, possibility of hierarchical designing and simulation are allowed too. That is useful for error correction by simulation of each part of designed algorithms. As the result of simulation of presented graph on figure 7 corresponding to SMA unit for bit-serial calculations we have control vectors for FPGA device. It works similar to previous toolbox described in section 3.3.

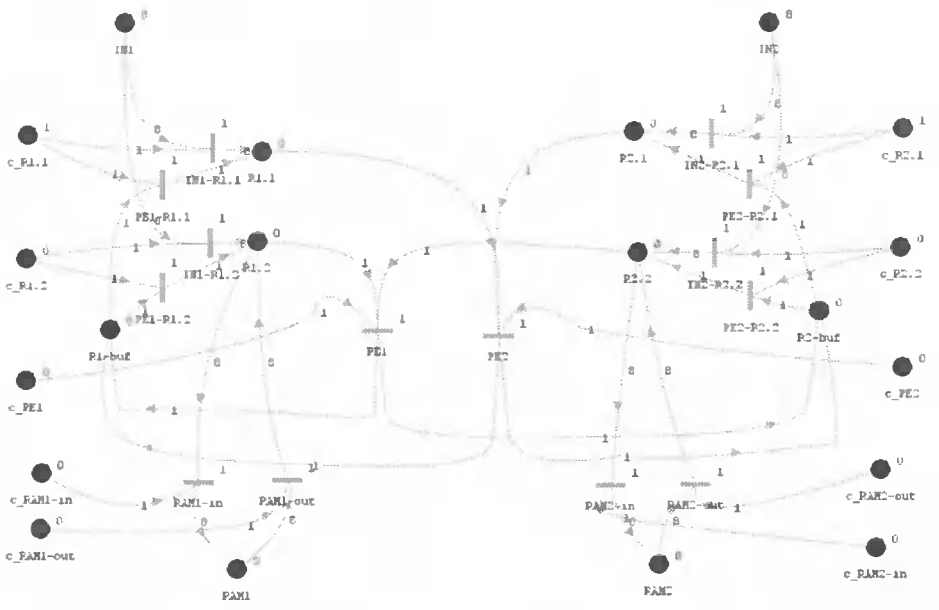


Fig. 7. Petri Net for SMA unit
Rys. 7. Sieć Petriego przedstawiająca jednostkę SMA

Proposed approach allows for deadlock prevention in parallel processes. Every transition corresponds to enable signal in physical device. Figure 8 presents piece of simulation result. Two processing elements works together but PE1 is delayed by two steps. Moreover there is possibility to design hierarchical structure of the net. Every transition and place can be design as the subnet. Simulation of hierarchical structure is also possible. The control vectors of simulation result can be written as a text file, then control vectors can be loaded to Xilinx simulator for FPGA device simulation. Such approach allows for better scheduling because of running every elements of architecture as fast as it's possible, when proper data are ready for processing.

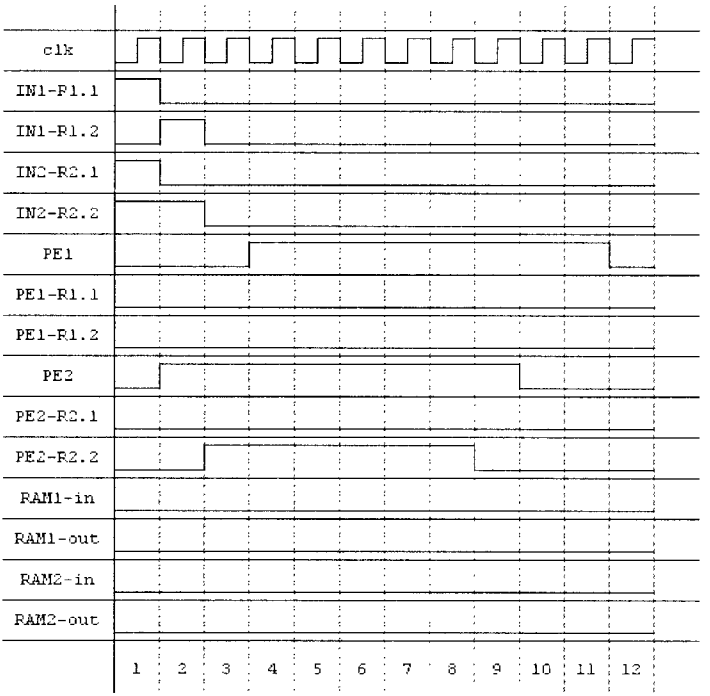


Fig. 8. Piece of simulation of Petri Net for SMA unit

4. IMPLEMENTATION AND SIMULATIONS RESULTS

The transformations were synthesized with Xilinx Project Navigator 10.1 for Virtex II xc2v3000. Top design is schematic (Fig. 9) and realized according with proposition on Fig. 4. Two blocks of data:

$$X_1 = \begin{bmatrix} 1 & 6 & 11 & 16 \\ 2 & 7 & 12 & 17 \\ 5 & 10 & 15 & 20 \\ 3 & 8 & 13 & 18 \end{bmatrix} \qquad X_2 = \begin{bmatrix} 255 & 226 & 225 & 226 \\ 224 & 225 & 225 & 226 \\ 224 & 225 & 226 & 226 \\ 224 & 225 & 225 & 228 \end{bmatrix}$$

were set as input in Xilinx ISE Simulator and the result was purchased:

$$Y_1 = \begin{bmatrix} 164 & -140 & 0 & -20 \\ -28 & 0 & 0 & 0 \\ -12 & 0 & 0 & 0 \\ 16 & 0 & 0 & 0 \end{bmatrix} \qquad Y_1 = \begin{bmatrix} 3605 & -18 & 1 & -9 \\ -1 & 15 & -3 & 0 \\ 3 & 0 & 3 & -5 \\ 2 & 5 & -4 & 5 \end{bmatrix}$$

The "controlUnit" block gets control vectors from the text file which was generated by SMA compiler, reads line by line and puts the 46-bit vector into "C_Signal" of "arch" units. There is only one main clock signal for SMA architecture.

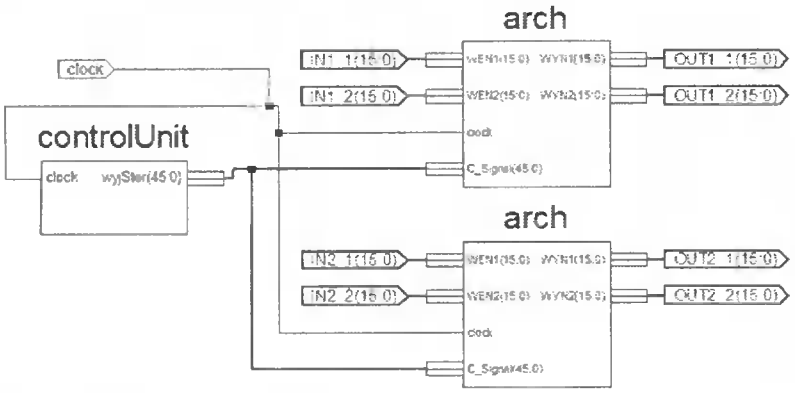


Fig. 9. Parallel form of SMA units
Rys. 9. Połączenie równoległe jednostek SMA

Simulation results shows that SMA units works as were programmed. Figures 10,11 shows start and the end of simulation process.

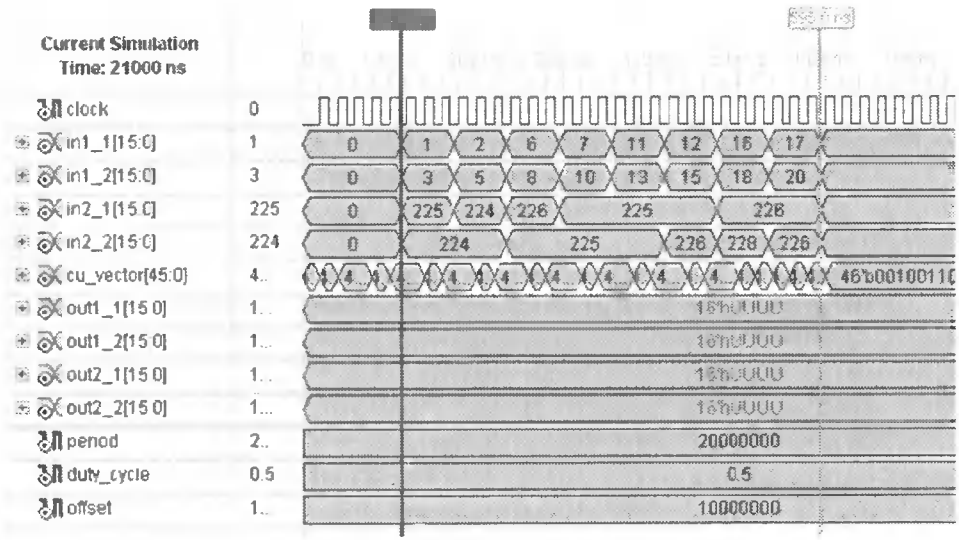


Fig. 10. Begin of simulation process
Rys. 10. Początek procesu symulacji

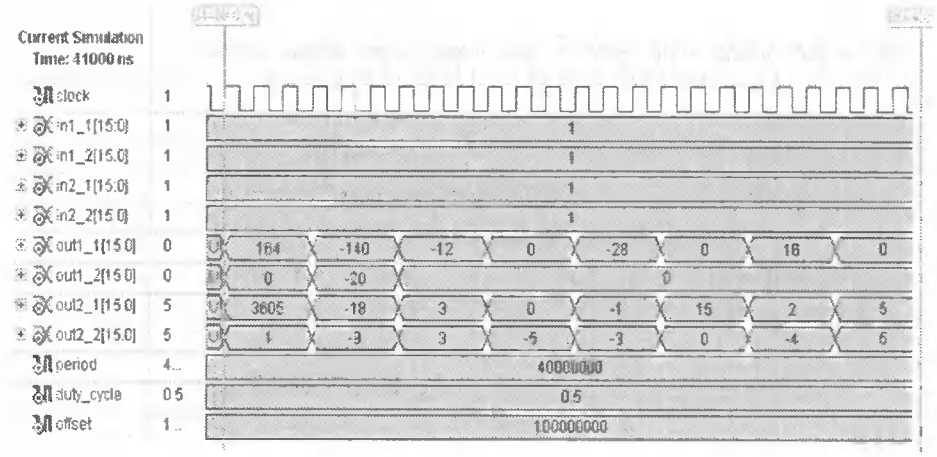


Fig. 11. End part of simulation process
Rys. 11. Końcowy fragment procesu symulacji

Table 1. Device utilization summary
Tabela 1. Podsumowanie wykorzystania zasobów układu

Number of Slices:	1243 out of 14336	8%
Number of Slice Flip Flops:	578 out of 28672	2%
Number of 4 input LUTs:	2192 out of 28672	7%
Number of bonded IOBs:	173 out of 684	25%
Number of 4 BUFGMUXs:	1 out of 16	6%

The same results of simulations using control vectors from SMA toolbox and Petri Nets approach are the same, but the second case allows for clock steps reduction. For better comparison of effectiveness for parallel processing, more tests will be done in the future work.

5. SUMMARY

In this paper, a new parallel 4x4 transform architecture based on bit-serial shared memory architecture was shown to improve processing rate for H.264/AVC and reduce power consumption. Reducing the power consumption is achieved by bit-serial communications as an alternative to bit-parallel interconnects. Simulations results of proposed SMA approach shows that calculations for one 4x4 set of data can be done in 879 steps of one SMA unit. Real time processing of FPGA device working with 100 MHz frequency, with implemented SMA unit can calculate 113705 of 4x4 blocks with 16-bit data. For video frame size 176x144 (3G standard) we have 1584 of 4x4 blocks. One SMA unit can calculate transformation with speed about 71 fps, so for 30 fps we can lower the clock rate to 50MHz. Applying proposed SMA approach for higher resolutions we need more SMA units connected in parallel form. For example 640x480 frames for 30 fps residual transform requires device of 100 MHz clock rate and 5 SMA units connected in parallel form.

BIBLIOGRAPHY

[1] H.S. Malvar, A. Hallapuro, M. Karaczewicz, L. Kerofsky, 2003: Low-Complexity Transform and Quantization in H.264/AVC. IEEE Trans. Circuits Syst Video Technol., vol. 13, no. 7.

[2] E. Hong, E. Jung, H. Fraz, D. Har, 2005: Parallel 4×4 transform architecture based on bit extended arithmetic for H.264/AVC. Proc. Int. Sym. On Circuits and Systems, vol. 1, pp. 95- 98.

[3] R. Kordasiewicz, S. Shirani, 2007: On Hardware Implementations Of DCT and Quantization Blocks for H.264/AVC. Journal of VLSI Signal Processing 47, pp. 189-199.

[4] R. Dobkin, A. Morgenshtein, A. Kolodny, R. Ginosar, 2008: Parallel vs. serial on-chip communication. Proc. of the 2008 International Workshop on System-Level Interconnection Prediction, NewCastle, pp. 43-50.

[5] ITU-T Rec. H.264/ISO/IEC 11496-10, 2002: Advanced video coding. Final Committee Draft, Document JVT-G050, December.

[6] L. Wanhammar, 1999: DSP integrated circuits. Academic Press, USA.

[7] G. Rubin, M. Omieljanowicz, A. Petrovsky, 2007: Reconfigurable FPGA-based hardware accelerator for embedded DSP," MIXDES'2007, Ciechocinek, pp.147-151.

[8] M. Adamski, M. Węgrzyn, 1994: Hierarchically Structured Colored Petri Net Specification and Validation of Concurent Controllers. Proc. in 39th International Scientific Colloquium, IWK'94, Ilmenau, Germany, Band 1, pp. 517-522.

[9] A. Węgrzyn, 2003: The symbolic analysis of binary control units using given methods of Petri nets. Rozprawa doktorska, Politechnika Warszawska (in Polish).

[10] A.A. Петровский, 1988: Techniques and microprocessors tools of fast and wideband processing of real-time processors, Наука и Техника, Минск (in Russian).

RÓWNOLEGŁE PRZEKSZTAŁCENIE 4X4 NA BITOWO-SZEREGOWEJ ARCHITEKTURZE O WSPÓŁDZIELONEJ PAMIĘCI DO ZASTOSOWAŃ H.264/AVC

Streszczenie

Praca zawiera opis implementacji oraz symulacji równoległego przekształcenia 4x4 stosowanego w H.264/AVC, bazując na bitowo-szeregowej architekturze o współdzielonej pamięci. W porównaniu z istniejącymi rozwiązaniami implementacji równoległej, proponowana architektura obliczeniowa redukuje liczbę linii połączeń wewnętrznych fizycznego układu FPGA. Zawiera ona również wyniki symulacji, pokazujące możliwość wykonywania przekształcenia w czasie rzeczywistym, przy zastosowaniu arytmetyki szeregowej.

Słowa kluczowe: FPGA, pamięć współdzielona, kodowanie wideo

ERRATA

In the paper "A short survey of recent representation results for linear system maps" by Irwin W. Sandberg, published in issue 9, two typographical errors occurred. They should be corrected as follows:

1. In footnote 5, the name Sobelov should be replaced with Sobolev.
2. In the third line above (2) "is in some respects" should replace "is some respects".

ISSN 1899-0088