POLSKA AKADEMIA NAUK
KOMITET ELEKTRONIKI I TELEKOMUNIKACJI

KWARTALNIK ELEKTRONIKI I TELEKOMUNIKACJI

ELECTRONICS AND TELECOMMUNICATIONS QUARTERLY

TOM 48 — ZESZYT 3/4

WYDAWNICTWO NAUKOWE PWN
WARSZAWA 2002

ADRES REDAKCJI
00-665 Warszawa, ul. Nowowiejska 15/19 Politechnika, pok. 470
Instytut Telekomunikacji, Gmach im. prof. JANUSZA GROSZKOWSKIEGO

Dyżury Redakcji: środy i piątki, godz. 14–16
tel/fax (022) 660 77 37

Telefony domowe: Redaktora Naczelnego: 812 17 65
Zast. Red. Naczelnego: 826 83 41
Sekretarza Odpowiedzialnego: 633 92 52

WYDAWNICTWO NAUKOWE PWN SA
Warszawa, ul. Miodowa 10

Ark. wyd. 11,00 Ark. druk. 8,75	Podpisano do druku w lipcu 2002 r.
Papier offset. kl. III 80 g. B-1	Druk ukończono w lipcu 2002 r.

SKŁAD I ŁAMANIE: PHOTOTEXT, Warszawa, ul. Chmielna 120
DRUK I OPRAWA: Zakład Poligraficzny Oja Druk, ul. Sporna 2F, Warszawa

Szanowni Autorzy

„Kwartalnik Elektroniki i Telekomunikacji" — Electronics and Telecommunications Quartery jest kontynuatorem tradycji powstałego 48 lat temu kwartalnika p.t. „Rozprawy Elektrotechniczne".

Kwartalnik jest czasopismem Komitetu Elektroniki i Telekomunikacji Polskiej Akademii Nauk. Wydawany jest przez Wydawnictwo Naukowe PWN SA w Warszawie. Kwartalnik jest czasopismem naukowym, na którego łamach są publikowane artykuły i komunikaty prezentujące wyniki oryginalnych prac teoretycznych i doświadczalnych, a także przeglądowych. Związane są one z szeroko rozumianymi dziedzinami współczesnej elektroniki, telekomunikacji, mikroelektroniki, optoelektroniki, radiotechniki i elektroniki medycznej.

Autorami publikacji są wybitni naukowcy, znani specjaliści o wieloletnim doświadczeniu, a także młodzi badacze — głównie doktoranci.

Artykuły charakteryzują się oryginalnym ujęciem zagadnienia, interesującymi wynikami badań, krytyczną oceną teorii lub metod, omówieniem aktualnego stanu, lub postępu danej gałęzi techniki oraz omówieniem perspektyw rozwojowych. Sposób pisania matematycznej części artykułów zgodny jest z wytycznymi IEC (International Electronics Commision) oraz ISO (International Organization of Standarization).

Wszystkie publikowane w Kwartalniku artykuły są recenzowane przez znanych krajowych specjalitów, co zapewnia że publikacje te są uznawane jako autorski dorobek naukowy. Opublikowanie w kwartalniku wyników prac naukowych zrealizowanych w ramach „GRANTów" Komitetu Badań Naukowych spełnia więc jeden z wymogów stawianych tym pracom.

Czasopismo dociera do wszystkich zajmujących się elektroniką i telekomunikacją krajowych ośrodków naukowych oraz technicznych, a także szeregu instytucji zagranicznych. Jest ponadto prenumerowane przez liczne grono specjalistów i biblioteki.

Każdy Autor otrzymuje bezpłatnie 20 egzemplarzy nadbitek swojego artykułu, co ułatwia przesłanie go do indywidualnych wybranych przez Autora osób i instytucji w kraju lub za granicą. Ułatwia to dodatkowo fakt, że w Kwartalniku są publikowane artykuły w języku angielskim.

Nadesłane do redakcji artykuły są publikowane w terminie około pół roku, w przypadku sprawnej współpracy Autora z Redakcją. Wytyczne dla Autorów dotyczące formy publikacji są zamieszczone w zeszytach Kwartalnika, można je także otrzymać w siedzibie Redakcji.

Artykuły można dostarczać osobiście, lub pocztą pod adresem zamieszczanym na stronie redakcyjnej w każdym zeszycie.

Redakcja

A. Ciarko
S. Kozieł
M. Gajer:
trans
P. Papiń
K. Wiatr,
dach
K. Wiatr,
rzys
Informac
Roczny s

A. Ciark
S. Kozie
M. Gaje
P. Papli
K. Wiat
K. Wiat
pro
Informat
Year co

SPIS TREŚCI

CONTENTS

KWARTALNIK ELEKTRONIKI I TELEKOMUNIKACJI, 2002, 48, z. 3–4, ss. 487–498

Aproximate representation for the Brillouin precursor in a Lorentz medium

ADAM CIARKOWSKI

Institute of Fundamental Technological Research
Polish Academy of Sciences
e-mail: aciark@ppt.gov.pl

Otrzymano 2001.12.12
Autoryzowano 2002.01.04

Propagation of the Brillouin precursor in a Lorentz dispersive medium is considered. The precursor is excited by a sine modulated signal, with the envelope described by a hyperbolic tangent function. A new approximate simple solution to the saddle point equation is found which determines space-time local characteristics of the precursor. With the help of this solution a uniform asymptotic representation for the precursor is constructed. Numerical examples are shown, employing Brillouin's choice of parameters describing the Lorentz medium.

Keywords: Lorentz medium, dispersive propagation, Brillouin precursor, uniform asymptotic expansions

1. INTRODUCTION

If very fast, rapidly oscillating pulse signals propagate in a medium, the assumption of identical medium reaction to various signal frequency components can no longer be justified. Due to dispersion, various frequency components of the signal propagate with different velocities and are differently dumped. As a result, the form of the signal undergoes significant modification as the signal propagates in the medium. This phenomenon is characteristic of modern electronic devices.

In many applications the dispersion phenomenon can be conveniently described in terms of the Lorentz model of the medium. Fundamental works on electromagnetic propagation in the Lorentz media are due to Sommerfeld [1] and Brillouin [2, 3]. These authors have shown that if a Dirac delta, or Heaviside unit step frequency modulated signal is excited in a Lorentz medium, the resultant disturbance propagating in the

medium splits into three components: small, first precursor whose front propagates with the velocity of light, followed by a larger, slowly oscillating second precursor, and finally the main signal.

Because of very rapid oscillations of the signals involved, the analysis of their propagation with the use of numerical methods becomes troublesome. Instead, asymptotic methods appeared to be appropriate tools in that case. Asymptotic approach has been adapted yet in Brillouin's work [3]. However, due to limitations of then available methods, now referred to as non-uniform ones, the author could not investigate some interesting aspects of dispersive propagation. Significant step forward is due to Chester at al. [4] and Bleistein and Handelsman [5], who elaborated uniform asymptotic techniques applicable in the analysis. With the use of those techniques the description of precursors dynamics in the medium could be done more precisely. Basing on uniform asymptotic methods, thorough investigation of signal propagation in dispersive media has been carried out by Kelbert and Sazonov [6] and Oughstun and Sherman [7].

One important issue in the analysis of precursor dynamic behavior in a Lorentz medium is proper determination of the locations of saddle points, relevant to integral description of the precursor. Those locations are described by the saddle point equation, which cannot be solved in a closed form. In the literature different approximate solutions of the equation were given. They serve to approximate the locations of near and distant saddle points. The latter points correspond to the dynamics of the first precursor while the former ones describe the behavior of the second precursor. In this paper we offer still another approximation to the solution of the saddle point equation for the near saddle points. With the use of this approximation, a very simple uniform asymptotic representation for the second precursor is obtained, corresponding to a finite rise-time initial signal.

2. FORMULATION OF THE PROBLEM

Consider one dimensional electromagnetic propagation in a Lorentz medium. The medium is characterized by the frequency-dependent complex index of refraction

$$n(\omega) = \left(1 - \frac{b^2}{\omega^2 - \omega_0^2 + 2i\delta\omega}\right)^{\frac{1}{2}}, \tag{1}$$

where b is so called plasma frequency of the medium, δ is a damping constant and ω_0 is a characteristic frequency.

It is assumed that in the plane $z = 0$ an electromagnetic signal is turned on at the moment $t = 0$. For $t > 0$ it oscillates with a fixed frequency ω_c and its envelope is described by a hyperbolic tangent function. Denote x-component of the electric field by $A(0, t)$. Then $A(0, t)$ has the form

$$A(0,t) = \begin{cases} 0 & t < 0 \\ \tanh \beta t \sin \omega_c t & t \geq 0 \end{cases}, \tag{2}$$

The parameter β determines how fast the signal grows before it attains its limiting amplitude equal unity.

This electromagnetic disturbance excites a signal $A(z, t)$ inside the medium. In what follows we will be interested in the field propagating in the half-space $z>0$. The problem under investigation then can be classified as a mixed, initial-boundary value problem for Maxwell equations with its solution vanishing at $z \to \infty$.

The exact solution for this specific form of the initial signal $A(0, t)$ is described by the contour integral [8]

$$A(z,t) = \int_{ia-\infty}^{ia+\infty} g(\omega) e^{\frac{z}{c}\varphi(\omega,\theta)} d\omega \tag{3}$$

defined in the complex frequency plane ω. Here,

$$g(\omega) = \frac{1}{4\pi}\left[\frac{\mathrm{i}}{\beta}\mathcal{B}\left(-\frac{i(\omega-\omega_c)}{2\beta}\right)+\frac{1}{\omega-\omega_c}-\frac{\mathrm{i}}{\beta}\mathcal{B}\left(-\frac{i(\omega+\omega_c)}{2\beta}\right)-\frac{1}{\omega+\omega_c}\right], \tag{4}$$

the complex phase $\varphi(\omega, \theta)$ function is given by

$$\varphi(\omega,\theta) = i\frac{c}{z}\left[\hat{k}(\omega)z-\omega t\right] = i\omega[n(\omega)-\theta), \tag{5}$$

and $\mathcal{B}$ is the beta function [9] defined by the psi function as

$$\mathcal{B}(u) = \frac{1}{2}\left[\psi\left(\frac{u+1}{2}\right)-\psi\left(\frac{u}{2}\right)\right]. \tag{6}$$

This function is related to the envelope of $A(0, t)$ via Fourier transformation. The dimensionless parameter

$$\theta = \frac{ct}{z} \tag{7}$$

defines a space-time point (z, t) in the field. The integration path in (3) is the line $\omega = \omega' + ia$, where a is a constant greater than the abscissa of absolute convergence for the function in square brackets in (4), and ω' ranges from negative to positive infinity.

A suitable approach revealing the physical structure of the signal defined by (3) is that based on asymptotic considerations. Sommerfeld and Brillouin have shown [3] that in mature dispersive regime (i.e. for large values of z), in addition to the main signal two accompanying signals are formed in the medium. One of them propagates with a high velocity and is very rapidly oscillating. It can be attributed to a pair of distant saddle points of $\varphi(\omega, \theta)$, located in the complex frequency plane symmetrically with respect to the imaginary axis. The location of these points is governed by the saddle point equation

$$n(\omega)+\omega n'(\omega)-\theta = 0. \tag{8}$$

As θ increases from unity to infinity those points vary in such a way that their distance from origin decreases from infinity (where they form a saddle point of infinite order) to $\sqrt{\omega_0^2+b^2+\delta^2}$. This signal is referred to as primary, or Sommerfeld precursor. The dynamics of the primary precursor resulting from the excitation (2) has been recently described in [10].

Two other saddle points, being also subject to (8), vary in the domain $|\omega| < \sqrt{\omega_0^2+\delta^2}$. They are called near saddle points. As θ increases from 1 to a value we denote by θ_1, the near saddle points approach each other along the imaginary axis, and they coalesce to form a second order saddle point at $\theta = \theta_1$. As θ tends to infinity, they depart from the axis and symmetrically approach the points $\omega = \pm\omega_0 - i\delta$ in the right and the left complex ω half-plane, respectively. These saddle points correspond to the secondary (or Brillouin) precursor. It is larger in size than the primary one. Below $\theta = \theta_1$ it is changing monotonically, while above this value it slowly oscillates.

If $n(\omega)$ is eliminated from the saddle point equation (8), the equation can be written down in the form of a polynomial of eighth degree in ω equal to zero [7]. Therefore, it does not seem to have a solution in a closed form. Brillouin and then Oughstun and Sherman found different approximate solutions to it, which are discussed in [7], Chapt. 6. Yet another approximation is due to Kelbert and Sazonov [6]. In the next section we provide a new approximate solution to (8), which is simpler than those noted above, and is still quite good in an interval of θ, where major part of Brillouin precursor energy is concentrated. With the help of that approximation, in Sec. 4 we find simple representation for the precursor dynamics in the Lorentz medium.

3. APPROXIMATE SOLUTION TO THE SADDLE POINT EQUATION

In order to obtain a new approximate solution to (8) we expand $n(\omega)$ in powers of ω in the vicinity of $\omega = -i\delta$. This results in

$$n(\omega) \approx \sqrt{1+p} + \frac{p^2(i\delta+\omega)^2}{2b^2\sqrt{1+p}}, \tag{9}$$

where $p = b^2/(\omega^2-\delta^2)$. By using this in (8) and then solving for ω we find the following simple approximation for the location of the near saddle points

$$\omega_\pm \approx -\frac{2i\delta}{3} \pm \frac{p_a(\theta)}{3p}. \tag{10}$$

Here,

$$p_a(\theta) = \sqrt{6b^2[\sqrt{1+p}\,\theta-(1+p)]-p^2\delta^2}, \tag{11}$$

where $\arg p_a(\theta) = \pi/2$ if $\theta < \theta_1$ and $\arg p_a(\theta) = 0$ if $\theta > \theta_1$. It is seen from (10) and (11) that for small values of θ both saddle points are purely imaginary. If $p_a(\theta) = 0$, which occurs at

$$\theta_1 \approx \frac{6b^2(1+p)+p^2\delta^2}{6b^2\sqrt{1+p}}, \tag{12}$$

then both saddle points coalesce into one saddle point $\omega = \omega_s$ of the second order, which is approximately given by

$$\omega_s \approx -\frac{2i\delta}{3}. \tag{13}$$

The last result is the same as in Brillouin's approximation. Finally, when θ increases above θ_1, this saddle point separates into two simple saddle points that move off of the imaginary axis in the right and in the left ω half-plane, respectively, symmetrically with respect to the axis. This evolution of the saddle points reflects the evolution deduced from exact solutions to the saddle point equation (8). Note that the incoming and outgoing directions of SDP through $\omega = \omega_s$ are $5\pi/6$ and $\pi/6$, respectively.

At the extreme values of θ, i.e. as $\theta \to 1^+$ and $\theta \to \infty$ the approximation (10) differs from the exact result. This is a direct consequence of the way the refraction coefficient $n(\omega)$ is approximated. However the departures from exact behavior of the saddle points do not have significant consequences on the analysis of the precursor dynamics, as the precursor is strongly attenuated in these cases. It is also to be noted that simplicity of the approximation obtained here is a consequence of a specific choice of the value of $\omega = -i\delta$, about which $n(\omega)$ is expanded. Attempts to use different values of ω led to approximations much more complicated in form than that obtained in this section.

4. UNIFORM ASYMPTOTIC REPRESENTATION OF THE SECOND PRECURSOR

Systematic procedure of constructing uniform asymptotic expansions for integrals with two neighbouring simple saddle points was given by Chester et. al. [4], and then incorporated into a general scheme of finding uniform asymptotic expansions of integrals by Bleistein and Handelsman [5]. It is also presented in [12].

The first step in the procedure is to change the integration variable t in (3) to a new variable s, such that the map $s(\omega)$ in some disk D containing the saddle points $\omega_\pm$ (but not any other saddle points) is conformal and at the same time the exponent takes the simplest

$$\varphi(\omega, \theta) = \varrho + \gamma^2 s - \frac{s^3}{3} \equiv \tau(s, \theta), \tag{14}$$

polynomial form. Notice that $\tau(s, \theta)$ has two simple saddle points $s = \pm\gamma$ that can coalesce to one saddle point $s = 0$ of the order 2, corresponding to $\omega = \omega_s$. We infer from

$$\dot{\omega} = \frac{\gamma^2 - s^2}{\varphi'_\omega(\omega, \theta)}, \tag{15}$$

that for $s(\omega)$ to be conformal, $s = \pm\gamma$ should correspond to $\omega = \omega_\pm$. Then,

$$\dot{\omega}(\pm\gamma) = \sqrt{\frac{\mp 2\gamma}{\varphi''(\omega_\mp)}}, \tag{16}$$

where $\varphi''(\omega_\mp)$ is a short notation for $\varphi''_{\omega\omega}(\omega_\mp, \theta)$.
With the help of (10) we find

$$\varphi(\omega_\pm) \approx \frac{1}{3}\left(2\delta \mp \frac{ip_a(\theta)}{p}\right)\left[\sqrt{1+p} - \theta - \frac{(p\delta \pm ip_a(\theta))^2}{18b^2\sqrt{1+p}}\right]. \tag{17}$$

By using the correspondence $\omega_\pm \leftrightarrow \pm\gamma$ in (14) the following expressions for γ^3 and ϱ are obtained

$$\frac{4\gamma^3}{3} = \varphi(\omega_+) - \varphi(\omega_-) \approx \frac{2ip_a^3(\theta)}{27b^2p\sqrt{1+p}}, \tag{18}$$

$$\varrho = \frac{1}{2}[\varphi(\omega_+) + \varphi(\omega_-)] \approx \frac{\delta[-p^2\delta^2 + 18b^2(1+p-\sqrt{1+p}\,\theta)]}{27b^2\sqrt{1+p}}. \tag{19}$$

The equation (18) for γ has three complex roots. Only one root corresponds to the regular branch of the transformation (14), that leads to the conformal map $s(\omega)$. To find the proper value of γ we first note that at $s = 0$,

$$\dot{\omega}(0) = \left[\frac{-2}{\varphi''(\omega_s)}\right]^{1/3}. \tag{20}$$

It is readily seen that here $\arg\dot{\omega}$ can take one of the three values: $\pi/6$, $5\pi/6$ or $-\pi/2$. We choose the value $\arg\dot{\omega}(0) = -\pi/2$. Because of continuity, the arguments of $\arg\dot{\omega}(\pm\gamma)$ tend to the same value as $\gamma \to 0$.
It then follows that

$$\dot{\omega}(\pm\gamma) \approx -i\frac{2^{1/2}b^{2/3}(1+p)^{1/6}}{3^{1/3}p^{2/3}}. \tag{21}$$

Since $\arg\varphi''(\omega_-) = 0$ if $\theta < \theta_1$ and $-\pi/2$ if $\theta > \theta_1$, then by (16),

$$\gamma = -\frac{ip_a(\theta)}{(3b)^{2/3}(2p)^{1/3}(1+p)^{1/6}}, \tag{22}$$

which implies that $\arg\gamma = 0$ if $\theta < \theta_1$ and $-\pi/2$ if $\theta > \theta_1$.

In the new variable of integration the integral (3) can be written down in the form

$$A(z,t) = \int_{C_1 \cap \tilde{D}} G(s,\theta) e^{\lambda \tau(s,\theta)} ds + \varepsilon, \tag{23}$$

where $\lambda = z/c$, and

$$G(s,\theta) = g[\omega(s)]\dot{\omega}(s). \tag{24}$$

The contour C_1 is an infinite arc in the left s complex half-plane, symmetrical with respect to the real axis, running upwards and having rays $\exp(-i2\pi/3)$ and $\exp(i2\pi/3)$ as its asymptotes. The domain $\tilde{D}$ is the image of D under (14) and ε is a term exponentially smaller than $A(z,t)$ itself.

We now represent $G(s,\theta)$ in the canonical form

$$G(s,\theta) = c_0 + c_1 s + (s^2 - \gamma^2) H_0(s,\theta). \tag{25}$$

Provided the function $H_0(s,\theta)$ is regular, the last term in (25) vanishes at the saddle points $s = \pm\gamma$, and its contribution to the asymptotic expansion is smaller than that from the first two terms. Indeed, it can be shown that integration of the last term by parts leads to integral of similar form as (23) multiplied by λ^{-1}.

To determine c_0 and c_1 we substitute $s = \pm\gamma$ in (25) and thus find

$$c_0 = \frac{G_6(\gamma,\theta) + G_6(-\gamma,\theta)}{2}, \tag{26}$$

$$c_1 = \frac{G_6(\gamma,\theta) - G_6(-\gamma,\theta)}{2\gamma}. \tag{27}$$

From (26) and (27) and with the use of (21) we obtain

$$c_0 = -i \frac{b^{2/3}(1+p)^{1/6}}{3^{1/3}(2p)^{2/3}} (g[\omega_-(\theta)] + g[\omega_+(\theta)]), \tag{28}$$

$$c_1 = \frac{3^{1/3} b^{4/3}(1+p)^{1/3}}{(2p)^{1/3} p_a(\theta)} (g[\omega_-(\theta)] - g[\omega_+(\theta)]). \tag{29}$$

We also note that the canonical integral $\int_{C_1} e^{\lambda \tau(s,\theta)} d\theta$ is related to the Airy function [5]

$$\int_{C_1} e^{-\lambda(s^3/3 - \gamma^2 s)} ds = 2\pi i \lambda^{-1/3} Ai(\lambda^{2/3}\gamma^2). \tag{30}$$

By using (25), (28) and (29) in (23), and extending the integration contour in the resulting integrals to C_1, we finally find that the leading term of the asymptotic expansion of $A(z,t)$ as $\lambda \to \infty$ is given by

$$A(z,t) \sim 2\pi i e^{\lambda\varrho(\theta)} \left\{ \frac{c_0(\theta)}{\lambda^{1/3}} Ai[\lambda^{-2/3}\gamma^2(\theta)] + \frac{c_1(\theta)}{\lambda^{2/3}} Ai'[(\lambda^{-2/3}\gamma^2(\theta)] \right\}. \tag{31}$$

The expansion is described in terms of the Airy function and its derivative. It holds for any $\gamma(\theta)$, including $\gamma = 0$. The last case corresponds to coalescing of the two simple saddle points $s = \pm\gamma$ into one saddle point of the second order. In other words the expansion is uniform in γ, and hence in θ. (Note that the asymptotic representation of the precursor obtained with the classical, saddle point method blows up at $\theta = \theta_1$, i.e. when the simple near saddle points coalesce into one saddle point of the second order.) It is seen that for $\gamma = 0$, i.e. for $\theta = \theta_1$, the algebraic order of $A(z,t)$ in λ is $\lambda^{-1/3}$. This behavior is characteristic of an integral with a saddle point of the second order.

For γ well separated from zero the Airy function and its derivative can be replaced by their asymptotic expansions

$$Ai(x) \sim e^{-\frac{2x^{3/2}}{3}} \left[\frac{1}{2\sqrt{\pi}x^{1/4}} + O(x^{-3/4}) \right], \qquad Ai'(x) \sim e^{-\frac{2x^{3/2}}{3}} \left[-\frac{x^{1/4}}{2\sqrt{\pi}} + O(x^{-3/4}) \right] \tag{32}$$

as $x \to \infty$, and

$$\begin{aligned} Ai(x) &\sim \frac{1}{\sqrt{\pi}(-x)^{1/4}} \left\{ \sin\left[\frac{2}{3}(-x)^{3/2} + \frac{\pi}{4} \right] + O(x^{-3/2}) \right\}, \\ Ai'(x) &\sim -\frac{(-x)^{1/4}}{\sqrt{\pi}} \left\{ \cos\left[\frac{2}{3}(-x)^{3/2} + \frac{\pi}{4} \right] + O(x^{-3/2}) \right\}, \end{aligned} \tag{33}$$

as $x \to -\infty$. By using these expansions in (23) we see that the algebraic order of $A(z,t)$ in λ is now $\lambda^{-1/2}$. This reflects the fact that in this case the contribution to the expansion is due to separate simple saddle points. The uniform expansion (31) provides smooth transition between these two cases.

5. NUMERICAL EXAMPLES

Consider the Lorentz dispersive medium described by the Brillouin parameters

$$b = \sqrt{20.0} \times 10^{16} s^{-1}, \qquad \omega_0 = 4.0 \times 10^{16} s^{-1}, \qquad \delta = 0.28 \times 10^{16} s^{-1} \tag{34}$$

and choose

$$\omega_c = 2.5 \times 10^{16} s^{-1}, \qquad \beta = 2.0 \times 10^{16} s^{-1}. \tag{35}$$

The dynamic evolution of the second precursor resulting from the hyperbolic tangent excitation and based on our approximation is shown in Fig. 1 and Fig. 2. They depict local field values A as found from (31) against the space-time variable θ. In Fig. 2, λ is

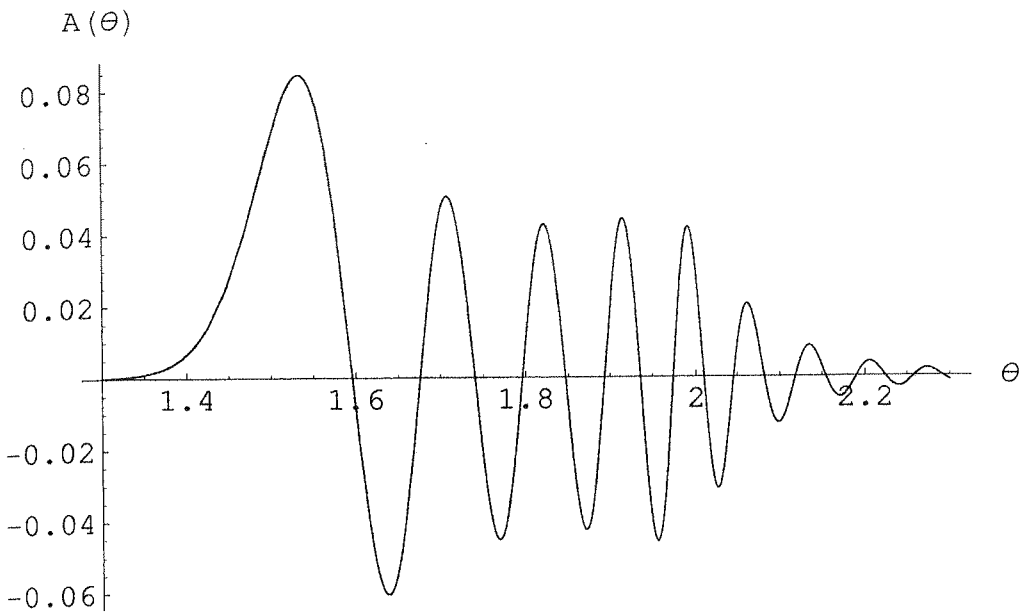


Fig. 1. The dynamic evolution of the second precursor in the medium described Brillouin's parameters. The saddle point locations are approximated by Eq. 10. It is assumed that $\omega_c = 2.5\times10^{16}\ s^{-1}$, $\beta = 2.0\times10^{16}\ s^{-1}$ and $\lambda = 3.0\times10^{-15}\ s$

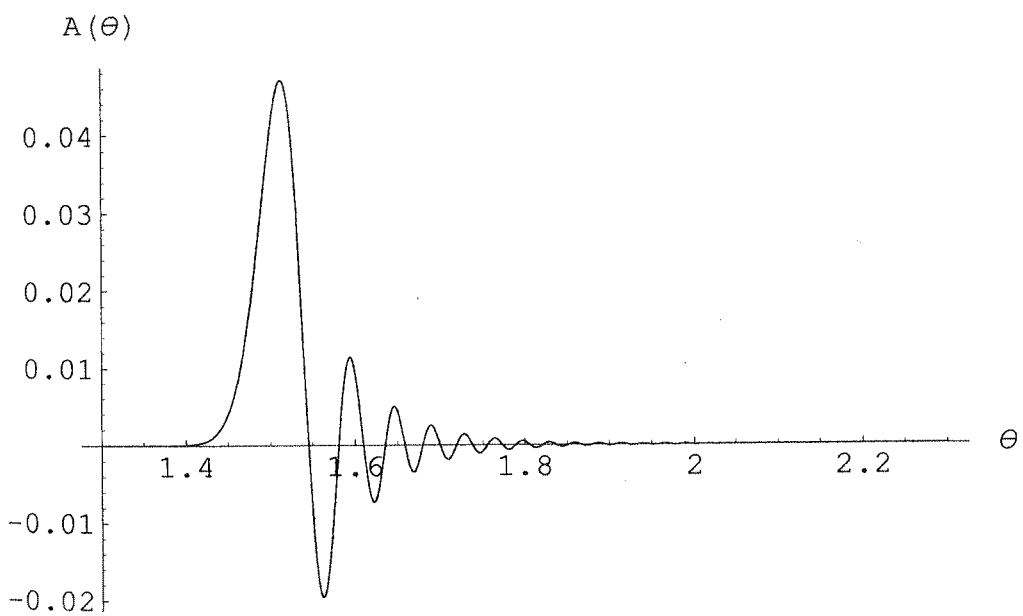


Fig. 2. The dynamic evolution of the second precursor in the medium described by Brillouin's parameters. The saddle point locations are approximated by Eq. 10. It is assumed that $\omega_c = 2.5\times10^{16}\ s^{-1}$, $\beta = 2.0\times10^{16}\ s^{-1}$ and $\lambda = 1.0\times10^{-15}\ s$

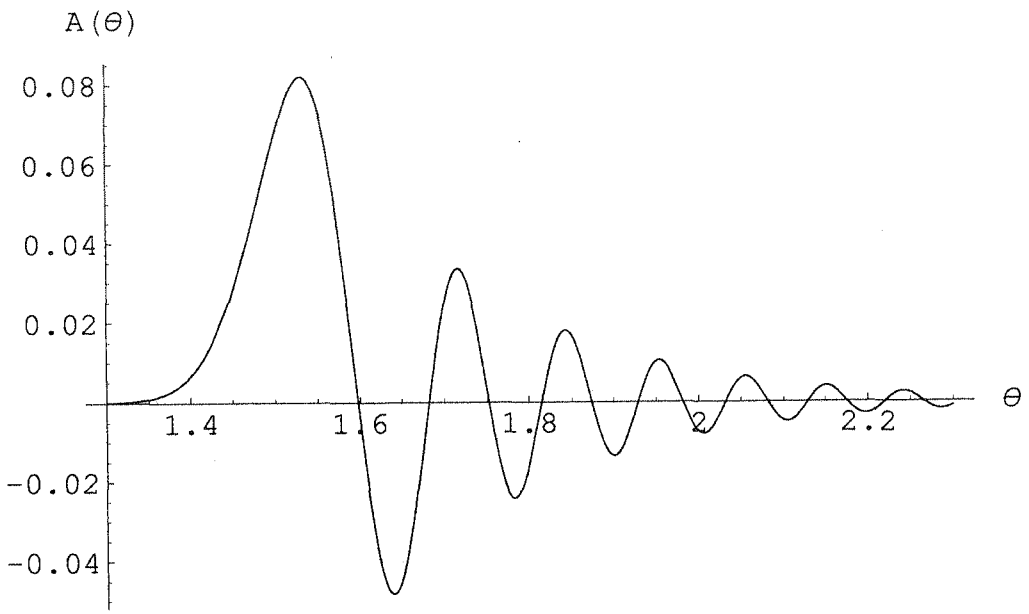


Fig. 3. The dynamic evolution of the second precursor in the medium described by Brillouin's parameters. The saddle point locations are exact. It is assumed that $\omega_e = 2.5\times10^{16}\ s^{-1}$, $\beta = 2.0\times10^{16}\ s^{-1}$ and $\lambda = 3.0\times10^{-15}\ s$

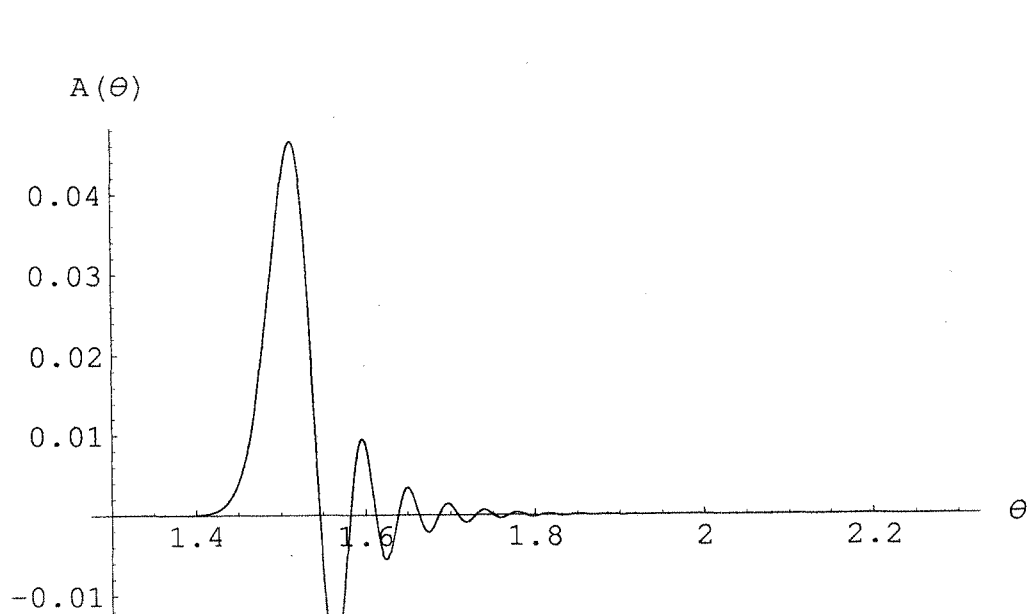


Fig. 4. The dynamic evolution of the second precursor in the medium described by Brillouin's parameters. The saddle point locations are exact. It is assumed that $\omega_e = 2.5\times10^{16}\ s^{-1}$, $\beta = 2.0\times10^{16}\ s^{-1}$ and $\lambda = 1.0\times10^{-15}\ s$

about three times larger than in Fig. 1. Note that as $\theta<\theta_1$, the arguments of the Airy function and its derivative are positive, and hence the precursor is monotonically growing in this space-time interval. If $\theta>\theta_1$, the arguments of these functions are negative, and consequently the precursor becomes oscillatory.

The effect of dumping of the precursor with increasing θ is clearly seen from the plots. The results here obtained can be compared against the results based on numerically exact solution to the saddle point equation, see Fig. 3 and Fig. 4.

Fig. 3 corresponds to Fig. 1, and Fig. 4 corresponds to Fig. 2. Differences between compared plots are observed at higher values of θ, where the approximation (9) of the refraction coefficient is less accurate.

Acknowledgment

The research presented in this work was partially supported by the State Committee for Scientific Research under grant 8 T11D 020 18.

6. CONCLUSIONS

In this paper we consider propagation of the Brillouin precursor in a dispersive Lorentz medium. This precursor, also referred to as the second precursor, follows the (first) Sommerfeld precursor, and precedes the main signal arrival at a particular space point. Description of its dynamics requires the knowledge of the near saddle points locations in a complex frequency plane. No exact, analytic formula describing the locations is available. Instead, different approximate representations for the locations have been obtained in the literature. In this paper we offer still another representation, based on the refraction coefficient power expansion in the vicinity of $\omega = -i\delta$. With this approximation we obtain a very simple, compact form representation for the Brillouin precursor, derived with the use of a uniform asymptotic technique. Unlike the representation found using the classical saddle point method, the representation obtained here is continuous for any value of the space-time coordinate θ, and properly describes the variation of the algebraic orders of the precursor in the parameter λ with changing θ. It also gives quite satisfactory numerical approximation of the precursor form in the Lorentz medium.

7. REFERENCES

1. A. Sommerfeld: *Über die Fortpflanzung des Lichtes in disperdierenden Medien.* Ann. Phys., Lepzig, 1914, vol. 44, pp. 177–202.
2. L. Brillouin: *Über die Fortpflanzung des Lichtes in disperdierenden Medien.* Ann. Phys., Lepzig, 1914, vol. 44, pp. 203–240.
3. L. Brillouin: *Wave Propagation and Group Velocity.* New York, Academic, 1960.
4. C. Chester, B. Friedman, F. Ursell: *An extension of the method of steepest descents.* Proc. of Cambridge Phil. Soc., 1957, vol. 53, pp. 599–611.

5. N. Bleistein, R. A. Handelsman: *Asymptotic Expansions of Integrals.* Holt, Rinehart and Winston, 1975, Ch. 9.
6. M. Kelbert, I. Sazonov: *Pulses and Other Wave Processes in Fluids.* Kluwer, 1996.
7. K. E. Oughstun, G. C. Sherman: *Electromagnetic Pulse Propagation in Casual Dielectrics.* vol. 16, Berlin, Springer, 1997.
8. A. Ciarkowski: *Asymptotic analysis of propagation of a signal with finite rise-time in a dispersive, lossy medium.* Arch. Mech., 1997, vol 49, pp. 877–892.
9. I. M. Rhyzhik, I. S. Gradshteyn: *Tables of Integrals, Sums, Series and Products.* 3-rd ed., National Publishers of the Technical Literature, Moscow, 1951, Sec. 6.39.
10. A. Ciarkowski: *Improved representation for the first precursor in the Lorentz medium.* Eng. Trans., 2000, vol. 48, pp. 43–59.
11. A. Ciarkowski: *Frequency dependence on space-time for electromagnetic propagation in dispersive medium.* Arch. Mech., 1999, vol. 51, pp. 33–46.
12. L. B. Felsen, N. Marcuvitz: *Radiation and Scattering of Waves.* Prentice Hall, 1973, Ch. 4.

A. CIARKOWSKI

PRZYBLIŻONA REPREZENTACJA PREKURSORA BRILLOUINA W OŚRODKU LORENTZA

Streszczenie

Rozpatrzono zagadnienie propagacji prekursora Brillouina w ośrodku dyspersyjnym Lorentza. Założono, że prekursor jest pobudzony sygnałem sinusoidalnym o obwiedni opisanej tangensem hiperbolicznym. Znaleziono nowe, przybliżone, proste rozwiązanie równania punktu siodłowego, opisujące lokalne własności czaso-przestrzenne prekursora. Korzystając ze znalezionego rozwiązania skonstruowano jednostajne wyrażenie asymptotyczne dla tego prekursora. Przedstawiono ilustrację numeryczną otrzymanych wyników, z zastosowaniem parametrów ośrodka zaproponowanych przez Brillouina.

Słowa kluczowe: ośrodek Lorentza, propagacja w ośrodku dyspersyjnym, prekursor Brillouina, jednostajne rozwinięcia asymptotyczne

KWARTALNIK ELEKTRONIKI I TELEKOMUNIKACJI, 2002, 48, z. 3–4, ss. 499–521

General Description of State-Space Continuous-Time G_m-C Filters

SŁAWOMIR KOZIEŁ, STANISŁAW SZCZEPAŃSKI

Faculty of Electronics, Telecommunications and Informatics,
Technical University of Gdańsk, ul. G. Narutowicza 11/12, 80-952 Gdańsk, Poland
e-mail: koziel@ue.eti.pg.gda.pl, stanisla@pg.gda.pl

Otrzymano 2001.11.13
Autoryzowano 2002.01.14

A general approach to analysis continuous-time analog G_m-C filters and equalizers, based on matrix description, is presented. A general transfer function formula for any G_m-C filter structure are given. The considerations accompanying this approach lead to useful relationships between the passive network of the filter and its transfer function. Based on these considerations, a new definition of continuous-time state-space G_m-C filters is given. Moreover, connections between state matrices for voltage- and current-mode state-space G_m-C filters are formulated and two canonical transformations of state-space filters into direct state-space ones, i.e. those having grounded capacitors only, are defined. Another two classes of G_m-C filters, i.e. reducible and quasi-state-space structures are also considered.

Keywords: state-space filters, G_m-C filters, matrix description, state-space representation.

1. INTRODUCTION

In recent years there has been a growing interest in continuous-time (CT) filters using transconductance amplifiers and capacitors (G_m-C). A number of attractive designs of such filters that are fully compatible with current CMOS technology have been reported in the literature [1]–[21]. The low-complexity excellent high-frequency performance and easy tunability of G_m-C filters make them suitable solutions for many applications such as hard disc drives, video filters, wireless communications, computer systems and instrumentation systems [11], [27]–[31].

In this paper, a novel and general approach to continuous-time G_m-C filters and equalizers is developed. Special attention is focused on an interesting sub-class of G_m-C filters, so-called state-space filters. This class of filters is usually identified with filters

that have only grounded capacitors [6], [25]. Such topologies are very attractive for integrated circuit (IC) realizations. In this work a novel definition of state-space filters is proposed, which comprises much wider class of G_m-C structures. According to this definition, the filter structure is called a state-space one if and only if there exist the state matrices for this structure. Of course, filter structures that have grounded capacitors satisfy this condition, however, there are many filters containing floating capacitors that are also state-space ones according to the new definition. Due to this, it is possible to apply a very useful technique of state-space description to much broader class of structures. It is important, since contemporary CMOS technologies make it possible to implement floating capacitors without any problems.

The paper is organized as follows. In Section 2, a general structure of G_m-C filter is presented. We derive a matrix description of general G_m-C filter based on a separate treatment of passive and active network of the circuit. It follows that the transfer function of any G_m-C filter can be easily calculated by means of the presented formulas. Section 3 is devoted to studying relationships between the transfer function of G_m-C filter and its passive network. In this section we also describe practical criteria of estimating the order of the filter's transmittance that can be used in computer aided filter synthesis. In Section 4 we deal with state-space G_m-C filters. A new definition of state-space G_m-C filters, based on invertibility properties of matrix corresponding to the passive network of the filter, is introduced. Relationships between state matrices of voltage- and current-mode G_m-C filters are derived. Moreover, we discuss two canonical transformations which enable us to convert any state-space filter into structures containing only grounded capacitors. In Section 5 we introduce two interesting classes of G_m-C structures, i.e. reducible and quasi-state-space ones. The paper concludes with Section 6.

2. GENERAL STRUCTURE OF G_m-C FILTER

Consider a general structure of a voltage-mode G_m-C filter shown in Fig. 1. The current-mode counterpart can be obtained by inverting all transconductors and interchanging input and output of the filter [23], [32]. The structure in Fig. 1 contains n internal nodes denoted as x_i, $i = 1,...,n$, n input transconductors G_{mbi}, an output summer consisting of transconductors c_iG_m and $-G_m$ as well as a feedforward transconductor dG_m. All transconductors form *active network*, while input capacitors C_{bi}, $i = 1,...,n$ and capacitors C_{ij}, $1 \le i \le j \le n$ form *passive network*. It is easily seen that any G_m-C filter is a particular case of the general structure in Fig. 1. Note also that n is not necessarily equal to the order of the filter transmittance. In the following, we will investigate the connection between the number n of internal nodes, the order of the filter and its particular structure.

Now we would like to derive an analytical description of the considered structure. In particular, we will calculate the transmittance $H(s)$ of the filter and show that the voltage-to-voltage transmittance $H_v(s)$ of the voltage-mode filter is the same as the current-to-current transmittance $H_c(s)$ of the current-mode one. In the following con-

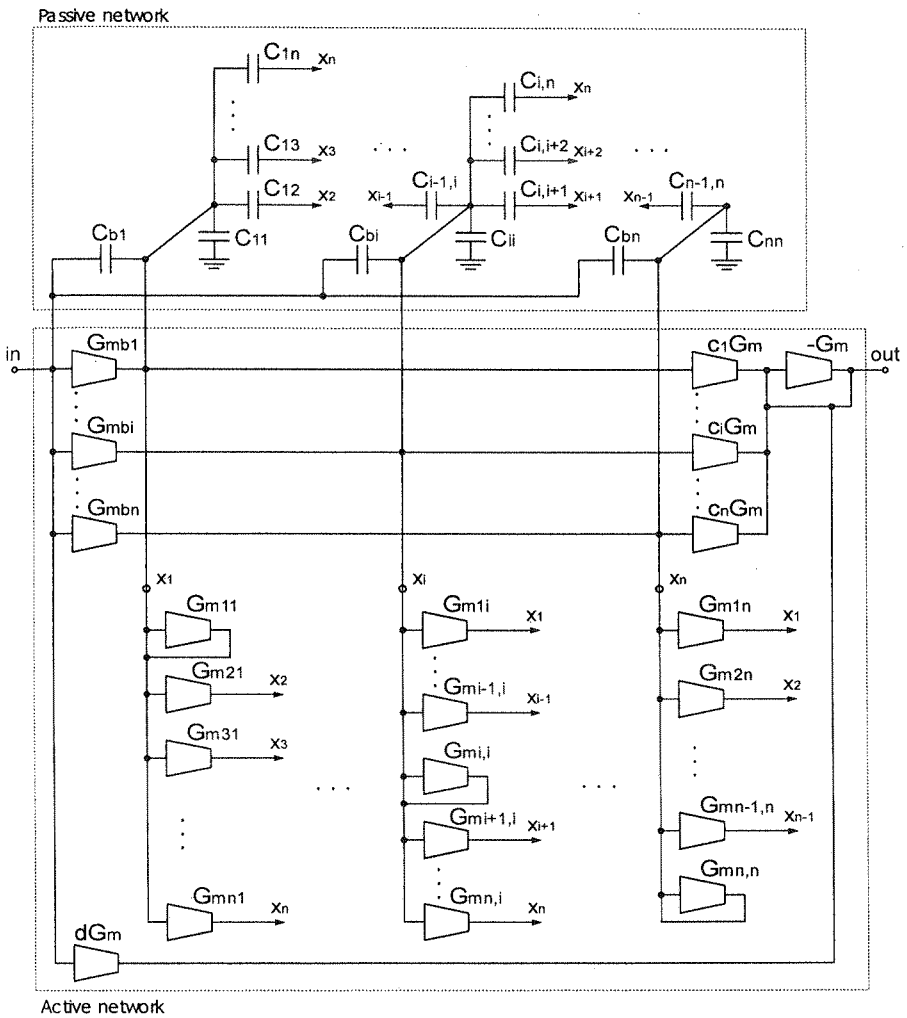


Fig. 1. General structure of voltage-mode G_m-C filter

siderations we will denote the voltage at the i-th node x_i also by x_i, which will not lead to confusion. A general structure of the voltage-mode G_m-C filter in Fig. 1 can be described by the following matrix equations:

$$s\begin{bmatrix} \sum_{j=1}^{n} C_{1j} & -C_{12} & \cdots & -C_{1n} \\ -C_{12} & \sum_{j=1}^{n} C_{2j} & \cdots & -C_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ -C_{1n} & -C_{2n} & \cdots & \sum_{j=1}^{n} C_{nj} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} u_i G_{mb1} + (u_i - x_1) s C_{b1} + \sum_{j=1}^{n} G_{m1j} x_j \\ u_i G_{mb2} + (u_i - x_2) s C_{b2} + \sum_{j=1}^{n} G_{m2j} x_j \\ \vdots \\ u_i G_{mbn} + (u_i - x_n) s C_{bn} + \sum_{j=1}^{n} G_{mnj} x_j \end{bmatrix} \quad (1)$$

and

$$u_0 = [c_1 \dots c_n] \cdot \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} + d \cdot u_i \tag{2}$$

where u_i, u_o are the input and output voltages, respectively. To simplify the notation, in (1) we used symbols C_{kl} $(k>l)$ that denote the same elements as C_{lk}. Note that the vector on the right-hand side of (1) can be written in the form:

$$\begin{bmatrix} G_{m11} & G_{m12} & \cdots & G_{m1n} \\ G_{m21} & G_{m22} & \cdots & G_{m2n} \\ \vdots & \vdots & \ddots & \vdots \\ G_{mn1} & G_{mn2} & \cdots & G_{mnn} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} - s \begin{bmatrix} C_{b1} & 0 & \cdots & 0 \\ 0 & C_{b2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & C_{bn} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} G_{mb1} + sC_{b1} \\ G_{mb2} + sC_{b2} \\ \vdots \\ G_{mbn} + sC_{bn} \end{bmatrix} u_i \tag{3}$$

Introducing the following notation:

$$\boldsymbol{T}_C = \begin{bmatrix} C_{b1} + \sum_{j=1}^{n} C_{1j} & -C_{12} & \cdots & -C_{1n} \\ -C_{12} & C_{b2} + \sum_{j=1}^{n} C_{2j} & \cdots & -C_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ -C_{1n} & -C_{2n} & \cdots & C_{bn} + \sum_{j=1}^{n} C_{nj} \end{bmatrix}, \tag{4a}$$

$$\boldsymbol{G} = \begin{bmatrix} G_{m11} & G_{m12} & \cdots & G_{m1n} \\ G_{m21} & G_{m22} & \cdots & G_{m2n} \\ \vdots & \vdots & \ddots & \vdots \\ G_{mn1} & G_{mn2} & \cdots & G_{mnn} \end{bmatrix}, \quad \boldsymbol{X} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}, \quad \tilde{\boldsymbol{C}}_v = [c_1 \dots c_n], \tag{4b}$$

$$\tilde{\boldsymbol{C}}_C = [G_{mb1} + sC_{b1} \cdots G_{mbn} + sC_{bn}], \quad \boldsymbol{D} = d \tag{4c}$$

(note that $\boldsymbol{T}_C$ is symmetrical matrix, i.e. $\boldsymbol{T}_C^T = \boldsymbol{T}_C$), one can rewrite (1) and (2) in the form of:

$$\begin{aligned} s\boldsymbol{T}_C\boldsymbol{X} &= \boldsymbol{G}\boldsymbol{X} + \tilde{\boldsymbol{C}}_c^T u_i \\ u_o &= \tilde{\boldsymbol{C}}_v\boldsymbol{X} + \boldsymbol{D}u_i \end{aligned} \tag{5}$$

In a similar way one can write appropriate equations for the current-mode filter. Using notation (4) we get:

$$\begin{aligned} s\boldsymbol{T}_C\boldsymbol{X} &= \boldsymbol{G}^T\boldsymbol{X}+\tilde{\boldsymbol{C}}_v^T i_i \\ i_o &= \tilde{\boldsymbol{C}}_c\boldsymbol{X}+\boldsymbol{D}i_i \end{aligned} \tag{6}$$

where the entries x_i of the vector $\boldsymbol{X}$ denote, as before, the intrinsic node voltages, and i_i, i_o are the input and output current, respectively.

On the basis of (5) and (6) we can calculate the transmittances H_v and H_c which are:

$$H_v(s) = \frac{u_o(s)}{u_i(s)} = \tilde{\boldsymbol{C}}_v(s\boldsymbol{T}_C-\boldsymbol{G})^{-1}\tilde{\boldsymbol{C}}_c^T+\boldsymbol{D} \tag{7a}$$

$$H_c(s) = \frac{i_o(s)}{i_i(s)} = \tilde{\boldsymbol{C}}_c(s\boldsymbol{T}_C-\boldsymbol{G}^T)^{-1}\tilde{\boldsymbol{C}}_v^T+\boldsymbol{D} \tag{7b}$$

It is worth noticing that using (7a) and (7b), one can easily check the reciprocal behavior of voltage- and current-mode filters:

$$\begin{aligned} H_v(s) = \frac{u_o(s)}{u_i(s)} &= \boldsymbol{C}_v(s\boldsymbol{T}_C-\boldsymbol{G})^{-1}\boldsymbol{C}_c^T+\boldsymbol{D} = (\boldsymbol{C}_v(s\boldsymbol{T}-\boldsymbol{G})^{-1}\boldsymbol{C}_c^T+\boldsymbol{D})^T = \\ &= \boldsymbol{C}_c(s\boldsymbol{T}_C-\boldsymbol{G}^T)^{-1}\boldsymbol{C}_v^T+\boldsymbol{D} = \frac{i_o(s)}{i_i(s)} = H_c(s) \end{aligned} \tag{8}$$

where we made use of the facts that $H^T = H$ (since H is a scalar), $(\boldsymbol{A}^{-1})^T = (\boldsymbol{A}^T)^{-1}$ for any invertible matrix $\boldsymbol{A}$ [22], and $\boldsymbol{T}_C$ is symmetrical. Since $H_v(s) = H_c(s)$, we will use general symbol $H(s)$ to denote a filter transmittance. Now, let us denote adjoint matrix of $s\boldsymbol{T}_C-\boldsymbol{G}$ as $\tilde{\boldsymbol{A}}$ where

$$\tilde{\boldsymbol{A}}(s) = adj(s\boldsymbol{T}_C-\boldsymbol{G}) = adj(s\boldsymbol{T}_C-\boldsymbol{G}^T)^T = \begin{bmatrix} \tilde{A}_{11}(s) & \tilde{A}_{12}(s) & \cdots & \tilde{A}_{1n}(s) \\ \tilde{A}_{21}(s) & \tilde{A}_{22}(s) & \cdots & \tilde{A}_{2n}(s) \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{A}_{n1}(s) & \tilde{A}_{n2}(s) & \cdots & \tilde{A}_{nn}(s) \end{bmatrix} \tag{9}$$

This allows us to rewrite H in the form:

$$H(s) = \frac{1}{det(s\boldsymbol{T}_C-\boldsymbol{G})}\sum_{i,j=1}^{n} c_i(G_{mbj}+sC_{bj})\tilde{A}_{ij}(s)+d \tag{10}$$

Note that many filter structures have only one input transconductor (i.e. no input signal distribution), a trivial output summer (i.e. one of the internal nodes is the output of the filter), and no input capacitors. This means that $\tilde{\boldsymbol{C}}_C = [0\cdots 0\ G_{mbk}\ 0\cdots 0]$, $\tilde{\boldsymbol{C}}_v = [0\cdots 0\ 1\ 0\cdots 0]-1$ at l-th position and $C_{bi} = 0$ for $i = 0, 1, \ldots n$. In such case, expression (13) reduces to the form:

$$H(s) = \frac{G_{mbk}\tilde{A}_{lk}(s)}{det(s\boldsymbol{T}_C - \boldsymbol{G})} \tag{11}$$

Similar expression can be written for two slightly more general cases. The first one is the case with no input capacitors, input signal distribution (i.e. $\tilde{\boldsymbol{C}}_c = [G_{mb1} \cdots G_{mbn}]$), and trivial output summer ($\tilde{\boldsymbol{C}}_v = 0 \cdots 0\ 1\ 0 \cdots 0] - 1$ at l-th position). Then, the filter transmittance takes the form:

$$H(s) = \frac{1}{det(s\boldsymbol{T}_C - \boldsymbol{G})} \sum_{i=1}^{n} G_{mbi}\tilde{A}_{li}(s) \tag{12}$$

Another case is when there are no input capacitors, no input signal distribution (i.e. $\tilde{\boldsymbol{C}}_c = [0 \cdots 0\ G_{mbk}\ 0 \cdots 0]$), however, there is non-trivial output summer (e.g. in follow--the-leader-like structures). Then, we have:

$$H(s) = \frac{G_{mbk}}{det(s\boldsymbol{T}_C - \boldsymbol{G})} \sum_{i=1}^{n} c_i\tilde{A}_{ik}(s) \tag{13}$$

On the basis of the above expressions one can easily calculate the transmittance of any particular structure of G_m-C filter. In the next section we deal with the sensitivity of our general G_m-C structure.

We end this section with an example of a particular filter structure considered in the general setting presented above. Fig. 2 shows a structure of third order elliptic G_m-C filter based on a LC ladder simulation.

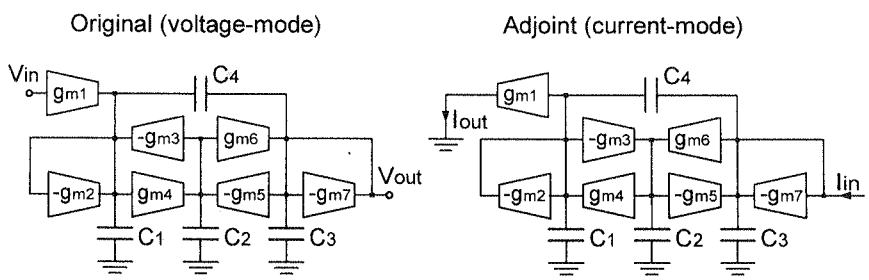


Fig. 2. Third order elliptic G_m-C filter in voltage- and current-mode

Matrices $\boldsymbol{T}_C$, $\boldsymbol{G}$, $\tilde{\boldsymbol{C}}_c$, $\tilde{\boldsymbol{C}}_v$ and $\boldsymbol{D}$ for this circuit are the following (internal nodes are indexed from left to right):

$$\boldsymbol{T}_C = \begin{bmatrix} C_1 + C_4 & 0 & -C_4 \\ 0 & C_2 & 0 \\ -C_4 & 0 & C_3 + C_4 \end{bmatrix}, \quad \boldsymbol{G} = \begin{bmatrix} -g_{m2} & -g_{m3} & 0 \\ g_{m4} & 0 & -g_{m5} \\ 0 & g_{m6} & -g_{m7} \end{bmatrix}, \quad \begin{aligned} \tilde{\boldsymbol{C}}_c &= [g_{m1}\ 0\ 0] \\ \tilde{\boldsymbol{C}}_v &= [0\ 0\ 1] \\ \boldsymbol{D} &= 0 \end{aligned} \tag{14}$$

Transmittance of the filter can be calculated using (11) with $k = 1$ and $l = 3$. Assuming for simplicity that all transconductances are equal, i.e. $g_{mi} = g_m$ for $i = 1,...,7$, we obtain:

$$\det(s\boldsymbol{T}_C - \boldsymbol{G}) =$$
$$= C_2(C_1C_3 + C_1C_4 + C_3C_4)s^3 + C_2(C_1 + C_3 + 2C_4)g_m s^2 + (C_1 + C_2 + C_3)g_m^2 s + 2g_m^2 \quad (15)$$

$$\tilde{A}_{31}(s) = C_2C_4s^2 + g_m^2 \quad (16)$$

Let $C_x^2 = C_1C_3 + C_1C_4 + C_3C_4$. Then, filter transmittance $H(s)$ has the form:

$$H(s) = \left(\frac{C_4 g_m}{C_x^2}\right) \cdot \frac{s^2 + \dfrac{g_m^2}{C_2C_4}}{s^3 + \dfrac{(C_1 + C_3 + 2C_4)g_m}{C_x^2}s^2 + \dfrac{(C_1 + C_2 + C_3)g_m^2}{C_x^2}s + \dfrac{2g_m^3}{C_2C_x^2}} \quad (17)$$

The above example shows that the presented general approach provides as with a very elegant and easy way to calculate transfer function of any G_m-C filter structure.

3. PASSIVE NETWORK OF GENERAL G_m-C FILTER

In this section we investigate how the passive network of general of G_m-C filter influences its properties, especially the order of filter transfer function $H(s)$. First, we would like to introduce necessary notation and give some general remarks.

Recall that matrix $\boldsymbol{T}_C$ representing the passive network of the filter is symmetrical. By construction of $\boldsymbol{T}_C$, for any $i = 1,...,n$ we have (see (4a)):

$$|(\boldsymbol{T}_C)_{ij}| \geq \sum_{k=1,\, k\neq i}^{n} |(\boldsymbol{T}_C)_{ik}|, \quad \text{and} \quad |(\boldsymbol{T}_C)_{ii}| \geq \sum_{k=1,\, k\neq i}^{n} |(\boldsymbol{T}_C)_{ki}| \quad (18)$$

i.e. matrix $\boldsymbol{T}_C$ is diagonally dominant. From linear algebra (Gershgorin's theorem), $\boldsymbol{T}_C$ is then positively semidefinite (that is $\boldsymbol{x}^T\boldsymbol{T}_C\boldsymbol{x} \geq 0$ for any nonzero $n \times 1$ vector $\boldsymbol{x}$). Moreover, $\boldsymbol{T}_C$ is invertible ($\det(\boldsymbol{T}_C) \neq 0$) if and only if $\boldsymbol{T}_C$ is positively definite (i.e. $\boldsymbol{x}^T\boldsymbol{T}_C\boldsymbol{x} > 0$ for any nonzero $n \times 1$ vector $\boldsymbol{x}$). Since $\boldsymbol{T}_C$ is symmetrical, there exists an orthogonal matrix $\boldsymbol{T}$ (i.e. $\boldsymbol{T}\boldsymbol{T}^T = \boldsymbol{T}^T\boldsymbol{T} = \boldsymbol{I}$, which implies $\boldsymbol{T}^{-1} = \boldsymbol{T}^T$) such that $\boldsymbol{T}^{-1}\boldsymbol{T}_C\boldsymbol{T}$ is diagonal (with non-negative elements, since $\boldsymbol{T}_C$ is positively semidefinite).

It is useful to introduce graphs representing passive network of the G_m-C filter. Let $G_t(V_t, E_t)$ be a graph in which V_t is the set of nodes $V_t = \{v_0, v_1, ..., v_n\}$ and E_t is a set of edges. Node v_0 represents both the common node x_0 and input node x_{in} of the filter, while $v_1, ..., v_n$ represent internal nodes $x_1, ..., x_n$ of the filter. Edge $e_{ij} \in E_t$ if and only if x_i and x_j are connected by the floating capacitor C_{ij} ($j > i > 0$) or x_0 and x_i are connected by

the grounded capacitor $C_{ii}(i>0)$. Edge $e_{in,j} \in E_t$ if and only if x_{in} and x_j are connected by input capacitor C_{bj}, $j = 1,...,n$. We define also graph $G_f(V_f, E_f)$ as the subgraph of G_t, in which $V_f = \{v_1,..., v_n\}$ represents internal nodes of the filter and $E_f \subset E_t$ such that $e_{ij} \in E_f$ if and only $j>i>0$. Fig. 3 gives an example of the passive network and corresponding graphs G_t and G_f. It is seen from Fig. 3 that our example graph G_t has three disjoint components, while G_f has four disjoint components.

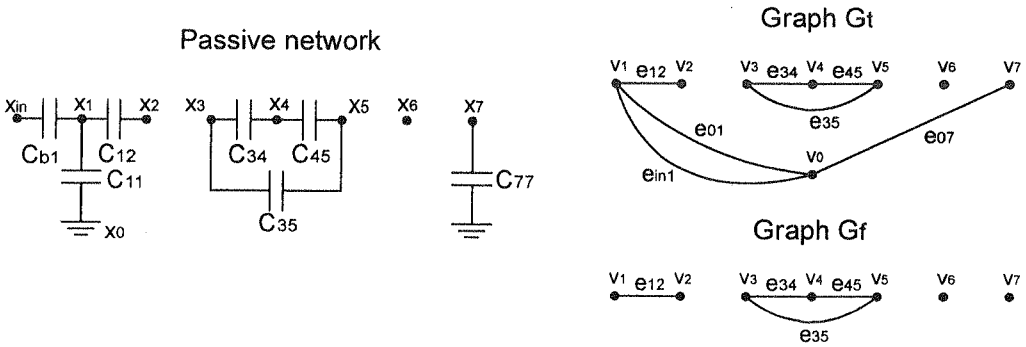


Fig. 3. Example of filter's passive network and related graphs G_t and G_f

Having defined graphs G_t and G_f, we can consider the structure of the matrix $\boldsymbol{T}_C$. It is seen from (4a) and the definition of graphs that $\boldsymbol{T}_C$ consists of square submatrices (*blocks*) situated along its diagonal (up to the renumbering of node indexes). The number of blocks is equal to the number of disjoint components of graph G_f reduced by the number of disjoint components of graph G_t having cardinality one and different from v_0. The latter represent just those internal nodes of the filter which are not connected to any other node by capacitance elements (we will call them *floating nodes*). The structure of matrix $\boldsymbol{T}_C$ for the passive network shown in Fig. 3 is the following:

$$\boldsymbol{T}_C \approx \begin{bmatrix} X & X & 0 & 0 & 0 & 0 & 0 \\ X & X & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & X & X & X & 0 & 0 \\ 0 & 0 & X & X & X & 0 & 0 \\ 0 & 0 & X & X & X & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & X \end{bmatrix} \tag{19}$$

It is evident that $\boldsymbol{T}_C$ cannot be invertible if filter has floating nodes. It is important for further applications to formulate necessary and sufficient conditions for invertibility of matrix $\boldsymbol{T}_C$. In the subsequent considerations we will identify the block of matrix $\boldsymbol{T}_C$ with the corresponding sets of capacitors. We have the following proposition:

Proposition 1. Let $\tilde{\boldsymbol{T}}_C$ be the block of matrix $\boldsymbol{T}_C$. Then, $\tilde{\boldsymbol{T}}_C$ is invertible if and only if the corresponding set of capacitors contains at least one grounded capacitor or input capacitor.

Proof. Matrix $\tilde{\boldsymbol{T}}_C$ is symmetrical and diagonally dominant since it inherits these properties from $\boldsymbol{T}_C$. Thus, $\tilde{\boldsymbol{T}}_C$ is invertible if and only if it is positively definite. Assume that $\tilde{\boldsymbol{T}}_C$ is $k\times k$ matrix. The general form of $\tilde{\boldsymbol{T}}_C$ is then analogous to those of $\boldsymbol{T}_C$ given by (4a) (with possible renumbering of node indexes if necessary). Let $\boldsymbol{y} = (y_1,...,y_k)^T$ be any nonzero vector. Then, we have (recall that $C_{ji} = C_{ij}$):

$$\boldsymbol{y}^T\tilde{\boldsymbol{T}}_C\boldsymbol{y} = \boldsymbol{y}^T\begin{bmatrix} C_{b1}+C_{11}+\sum_{j\neq 1}^{k}C_{1j} & -C_{12} & \cdots & -C_{1k} \\ -C_{21} & C_{b2}+C_{22}+\sum_{j\neq 2}^{k}C_{2j} & \cdots & -C_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ -C_{k1} & -C_{k2} & \cdots & C_{bk}+C_{kk}+\sum_{j\neq k}^{n}C_{kj} \end{bmatrix}\cdot\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_k \end{bmatrix} =$$

$$= \boldsymbol{y}^T\begin{bmatrix} y_1(C_{b1}+C_{11})+\sum_{j\neq 1}^{k}C_{1j}(y_1-y_j) \\ y_2(C_{b2}+C_{22})+\sum_{j\neq 2}^{k}C_{2j}(y_2-y_j) \\ \vdots \\ y_k(C_{bk}+C_{kk})+\sum_{j\neq k}^{k}C_{kj}(y_k-y_j) \end{bmatrix} = \sum_{i=1}^{k}y_i^2(C_{bi}+C_{ii})+\sum_{i=1}^{k}\sum_{j\neq 1}^{k}C_{ij}y_i(y_i-y_j) =$$

$$= \sum_{i=1}^{k}y_i^2(C_{bi}+C_{ii})+\sum_{i=1}^{k}\left(\sum_{j>i}^{k}C_{ij}y_i(y_i-y_j)+\sum_{i>j}^{k}C_{ij}y_i(y_i-y_j)\right) =$$

$$= \sum_{i=1}^{k}y_i^2(C_{bi}+C_{ii})+\sum_{i=1}^{k}\left(\sum_{j>i}^{k}C_{ij}y_i(y_i-y_j)+\sum_{j>i}^{k}C_{ij}y_j(y_j-y_i)\right) =$$

$$= \sum_{i=1}^{k}y_i^2(C_{bi}+C_{ii})+\sum_{i=1}^{k}\sum_{j>i}^{k}C_{ij}(y_i-y_j)^2$$

It is evident from (20) that $\boldsymbol{y}^T\tilde{\boldsymbol{T}}_C\boldsymbol{y} = 0$ if and only if $y_i = y_j$ for any $i,j = 1,...,k$ and $C_{bi}+C_{ii} = 0$ for $i = 1,...,k$. This proves our proposition since if $\tilde{\boldsymbol{T}}_C$ has at least one grounded or input capacitor then it is positively definite (so it is invertible). Otherwise, there exists $\boldsymbol{y}$ for which $\boldsymbol{y}^T\tilde{\boldsymbol{T}}_C\boldsymbol{y} = 0$ so $\tilde{\boldsymbol{T}}_C$ is not invertible.

Corollary 1. Let $\tilde{\boldsymbol{T}}_C$ be the block of matrix $\boldsymbol{T}_C$. Then, $\tilde{\boldsymbol{T}}_C$ is invertible if and only if the sum of its elements if positive.

Proof. It is apparent from (20) that the sum of elements of $\tilde{\boldsymbol{T}}_C$ equals $\sum_{i=1}^{k}(C_{bi}+C_{ii})$ and is positive if and only if $\tilde{\boldsymbol{T}}_C$ contains at least one grounded or input capacitor. Our assertion follows now from Proposition 1.

Corollary 2. Let $\tilde{\boldsymbol{T}}_C$ be the block of matrix $\boldsymbol{T}_C$. Suppose that the size of $\tilde{\boldsymbol{T}}_C$ is $k \times k$. Then, $rank(\tilde{\boldsymbol{T}}_C) = k$ if and only if at least one of the capacitors corresponding to $\tilde{\boldsymbol{T}}_C$ is grounded or input one. Otherwise, $rank(\tilde{\boldsymbol{T}}_C) = k-1$.

Proof. If at least one of the capacitors corresponding to $\tilde{\boldsymbol{T}}_C$ is grounded or input one, then it is invertible by Proposition 1, hence $rank(\tilde{\boldsymbol{T}}_C) = k$. Suppose now that $\tilde{\boldsymbol{T}}_C$ contains only floating capacitors $C_{ij}(j > i > 0)$. Then the sum of elements of $\tilde{\boldsymbol{T}}_C$ is zero by definition (see (20)). Consider any submatrix $(\tilde{\boldsymbol{T}}_C)_i$ of $\tilde{\boldsymbol{T}}_C$ defined by removing i-th row and i-th column of $\tilde{\boldsymbol{T}}_C$, $i = 1,...,k$. We assert that $(\tilde{\boldsymbol{T}}_C)_i$ is invertible. Obviously, $(\tilde{\boldsymbol{T}}_C)_i$ is symmetrical and diagonally dominant. Thus, it is enough to show that $(\tilde{\boldsymbol{T}}_C)_i$ is positively definite. Without loss of generality we can assume that $i = k$. Proceeding similarly as in the proof of Proposition 1 we obtain for any nonzero vector $\boldsymbol{y} = (y_1,...,y_{k-1})^T$:

$$\boldsymbol{y}^T(\tilde{\boldsymbol{T}}_C)_k\boldsymbol{y} = \sum_{i=1}^{k-1} y_i^2 C_{ik} + \sum_{i=1}^{k-1}\sum_{j>1}^{k-1} C_{ij}(y_i - y_j)^2 \tag{21}$$

This sum is positive since $C_{ik} > 0$ for at least one $i = 1,...,k-1$ by connectivity of the component of graph G_f corresponding to $\tilde{\boldsymbol{T}}_C$. We have proved that the $k-1 \times k-1$ submatrix $(\tilde{\boldsymbol{T}}_C)_k$ of $\tilde{\boldsymbol{T}}_C$ is invertible so $rank(\tilde{\boldsymbol{T}}_C) = k-1$.

Corollary 3. Let $\boldsymbol{T}_C$ be $n \times n$ matrix. Then, $rank(\boldsymbol{T}_C) = n+1-k$, where k is the number of disjoint components of the corresponding graph G_t.

Proof. It is apparent that $rank(\boldsymbol{T}_C) = \sum_{i=1}^{n} rank((\tilde{\boldsymbol{T}}_C)_i)$, where $(\tilde{\boldsymbol{T}}_C)_i$ is i-th block of $\boldsymbol{T}_C$ and N is the total number of blocks. Denote the number of floating nodes of the filter as n_f and number of blocks connected to the common node or to the input node (by at least one capacitor) as N_c. It is seen from the definition of graphs G_t and G_f that the number of disjoint components of graph G_t equals $k = N - N_c + 1 + n_f$. Indeed, components of G_t are the following: blocks not connected to the common node $(N-N_c)$, one component consisting of all blocks connected to the common node and n_f floating nodes. Now, $rank(\boldsymbol{T}_C) = n - n_f - (N - N_c)$ since each floating node 'produces' zero row and zero column in matrix $\boldsymbol{T}_C$ and, by Corollary 2, each block of $\boldsymbol{T}_C$ not connected to the common node reduces $rank(\boldsymbol{T}_C)$ by one. Thus $rank(\boldsymbol{T}_C) = n - n_f - (N - N_c) = n+1-(N-N_c+1+n_f) = n+1-k$ which concludes the proof.

It is easy to observe that matrix $\boldsymbol{T}_C$ is invertible if and only if the filter has no floating nodes and all blocks of $\boldsymbol{T}_C$ are invertible. Now, we formulate it in terms of graph G_t.

Proposition 2. Matrix $\boldsymbol{T}_C$ is invertible if and only if the corresponding graph G_t is connected.

Proof. Suppose that $\boldsymbol{T}_C$ is invertible. From the preceding paragraph there are no floating nodes and each block of $\boldsymbol{T}_C$ contains a grounded capacitor or input capacitor. This means that each component of graph G_f is connected as a subgraph of G_t to node v_0 by at least one edge. Thus, G_t is connected. Suppose now that G_t is connected. This implies in particular that the filter has no floating nodes. Moreover, if there exists a block without a grounded (or input) capacitor then G_t cannot be connected, which leads to contradiction. Thus, all blocks are invertible and so is $\boldsymbol{T}_C$.

Let us now turn to relationships between the passive network of the G_m-C filter and the order of the filter transmittance which we will denote as n_H. Recall that $H(s)$ is given by (11). Let $\boldsymbol{T}$ be any invertible $n\times n$ matrix (so $\boldsymbol{T}\boldsymbol{T}^{-1} = \boldsymbol{T}^{-1}\boldsymbol{T} = \boldsymbol{I}$ — identity matrix). Then, we have:

$$\begin{aligned} H(s) &= \tilde{\boldsymbol{C}}_v(s\boldsymbol{T}_C-\boldsymbol{G})^{-1}\tilde{\boldsymbol{C}}_c^T+\boldsymbol{D} = \tilde{\boldsymbol{C}}_v\boldsymbol{T}\boldsymbol{T}^{-1}(s\boldsymbol{T}-\boldsymbol{G})^{-1}\boldsymbol{T}\boldsymbol{T}^{-1}\tilde{\boldsymbol{C}}_c^T+\boldsymbol{D} = \\ &= \tilde{\boldsymbol{C}}_v\boldsymbol{T}(s\boldsymbol{T}^{-1}\boldsymbol{T}_c\boldsymbol{T}-\boldsymbol{T}^{-1}\boldsymbol{G}\boldsymbol{T})^{-1}\boldsymbol{T}^{-1}\tilde{\boldsymbol{C}}_c^T+\boldsymbol{D} \end{aligned} \tag{22}$$

This means that $H(s)$ is invariant under the transformation:

$$\tilde{\boldsymbol{C}}_v\to\tilde{\boldsymbol{C}}_v\boldsymbol{T}, \quad \tilde{\boldsymbol{C}}_c^T\to\boldsymbol{T}^{-1}\tilde{\boldsymbol{C}}_c^T, \quad \boldsymbol{T}_C\to\boldsymbol{T}^{-1}\boldsymbol{T}_C\boldsymbol{T}, \quad \boldsymbol{G}\to\boldsymbol{T}^{-1}\boldsymbol{G}\boldsymbol{T} \tag{23}$$

Using this fact we can prove the proposition which is the main goal of the section:

Proposition 3. Order n_H of the transmittance of a general G_m-C filter is not larger than the rank of matrix $\boldsymbol{T}_C$.

Proof. Transmittance of the filter is given by $H(s) = \tilde{\boldsymbol{C}}_v(s\boldsymbol{T}_C-\boldsymbol{G})^{-1}\tilde{\boldsymbol{C}}_c^T+\boldsymbol{D}$. It follows from the preceding paragraph that $H(s)$ is invariant under the transformation given by (23) for any invertible matrix $\boldsymbol{T}$. However, matrix $\boldsymbol{T}_C$ is symmetrical, hence there exists such $\boldsymbol{T}$ that $\boldsymbol{T}^{-1}\boldsymbol{T}_C\boldsymbol{T}$ is diagonal. Since $rank(\boldsymbol{T}_C)$ is invariant under similarity transformation, the number of nonzero (diagonal) elements of $\boldsymbol{T}^{-1}\boldsymbol{T}_C\boldsymbol{T}$ is exactly equal to $rank(\boldsymbol{T}_C)$. It follows that $det(s\boldsymbol{T}_C-\boldsymbol{G})$ is a polynomial in s of order $rank(\boldsymbol{T}_C)$. This means that $n_H \le rank(\boldsymbol{T}_C)$ (a strict inequality may occur when $H(s)$ has common poles and zeros).

Using Corollary 3 we can reformulate assertion of Proposition 3 in terms of graph G_t corresponding to matrix $\boldsymbol{T}_C$. Such a formulation may be more convenient for practical applications.

Proposition 4. Order n_H of the transmittance of a general G_m-C filter is not larger than $n+1-k$, where n is the number of internal nodes of the filter and k is the number of disjoint components of graph G_t corresponding to matrix $\boldsymbol{T}_C$.

Proof. The proof is straightforward and emerges from Proposition 3 and Corollary 3.

Note that the estimate similar to that given in Proposition 4 is known in literature (e.g. [33]), however here it is formulated in terms of matrix $\boldsymbol{T}_C$, hence more convenient for practical applications.

The results formulated in this section have both theoretical and practical significance. For example, Proposition 2 can be used to define some important class of G_m-C filters, which is done in the next section. Proposition 4 can be useful from the point of view of computer aided synthesis of G_m-C filters. This is because it allows us to estimate a priori the possible order of transmittance of a filter generated by any design algorithm without calculating this transmittance (which is usually a time consuming task) so it can be a convenient criterion while choosing of a particular filter structure within some design strategies.

We end this section with the following remark. It has been mentioned in the discussion preceding Proposition 3 that the transformation given by (23) does not influence transmittance of the filter for any invertible matrix $\boldsymbol{T}$. It is worth noticing that

such a transformation may determine a new realization of $H(s)$. Suppuse, in particular, that $\boldsymbol{T}$ is similarity matrix for which $\boldsymbol{T}^{-1}\boldsymbol{T}_C\boldsymbol{T}$ is diagonal, and that there are no input capacitors in the filter (i.e. $C_{bi} = 0$ for $i = 1,...,n$). Then, transformed matrices $\tilde{\boldsymbol{C}}_v\boldsymbol{T}$, $\boldsymbol{T}^{-1}\tilde{\boldsymbol{C}}_c^T$, $\boldsymbol{T}^{-1}\boldsymbol{T}_C\boldsymbol{T}$, $\boldsymbol{T}^{-1}\boldsymbol{G}\boldsymbol{T}$ determine a new structure that realizes the same transmittance (of course, the considered transformation is nontrivial if $\boldsymbol{T}_C$ is not diagonal). This new structure has only grounded capacitors with values equal to the values of nonzero diagonal elements of $\boldsymbol{T}^{-1}\boldsymbol{T}_C\boldsymbol{T}$.

4. STATE-SPACE G_m-C FILTERS

In this section we consider a certain class of filters which we call state-space G_m-C filters. Recall that a general description of linear filtering circuits can be obtained using the state-space representation. The appropriate state equations can be written as follows [7], [25]:

$$\begin{aligned} s\boldsymbol{X}(s) &= \boldsymbol{A}\boldsymbol{S}(s)+\boldsymbol{B}U(s) \\ Y(s) &= \boldsymbol{C}\boldsymbol{X}(s)+\boldsymbol{D}U(s) \end{aligned} \tag{24}$$

where $\boldsymbol{X}$, U, Y are, respectively: state vector consisting of output signals of n internal integrators (thus, $\boldsymbol{X} \in \boldsymbol{M}_{n\times 1}$, where $M_{n\times m}$ denotes the set of $n\times m$ matrices), input signal and output signal; $\boldsymbol{A} \in M_{n\times n}$, $\boldsymbol{B} \in M_{n\times 1}$, $\boldsymbol{C} \in M_{1\times n}$, $\boldsymbol{D} \in M_{1\times 1}$.

Using (50), the overall transfer function of the filter can be expressed in the form:

$$H(s) = \frac{Y(s)}{U(s)} = \boldsymbol{C}(s\boldsymbol{I}-\boldsymbol{A})^{-1}\boldsymbol{B}+\boldsymbol{D} \tag{25}$$

where $\boldsymbol{I} \in M_{n\times m}$ stands for the identity matrix. It is useful to introduce internal transfer functions f_i and g_i, i.e. the transfer function from the input of the filter to the output of integrator i (i.e. to x_i) and the transfer function from the input of integrator i to the output of the filter, respectively [7]. Using the following notation:

$$\boldsymbol{F} = (f_1\ f_2 \cdots f_n)^T, \qquad \boldsymbol{G} = (g_1\ g_2 \cdots g_n) \tag{26}$$

one can define the so-called observability and controllability Gramian matrices $\boldsymbol{W}$ and $\boldsymbol{K}$ [7]:

$$\boldsymbol{W} = \frac{1}{2\pi}\int_{-\infty}^{\infty} \boldsymbol{G}*\boldsymbol{G}d\omega, \qquad \boldsymbol{K} = \frac{1}{2\pi}\int_{-\infty}^{\infty} \boldsymbol{F}\boldsymbol{F}*d\omega \tag{27}$$

One can also prove that the matrices $\boldsymbol{W}$ and $\boldsymbol{K}$ satisfy the following equations:

$$\boldsymbol{A}^T\boldsymbol{W}+\boldsymbol{W}\boldsymbol{A} = -\boldsymbol{C}^T\boldsymbol{C}, \tag{28a}$$

$$\boldsymbol{A}\boldsymbol{K}+\boldsymbol{K}\boldsymbol{A}^T=-\boldsymbol{B}\boldsymbol{B}^T \tag{28b}$$

Now, we turn to the general description of a G_m-C filter presented in Section 2. Assume that matrix $\boldsymbol{T}_C$, corresponding to the passive network of the filter, is invertible, and that there are no input capacitors in the filter (i.e. $C_{bi}=0$ for $i=1,\ldots,n$ and $\tilde{\boldsymbol{C}}_c=[G_{mb1}\cdots G_{mbn}]$). Then, we can rewrite (5) and (6) in the following form:

$$\begin{aligned} s\boldsymbol{X} &= \boldsymbol{T}_C^{-1}\boldsymbol{G}\boldsymbol{X}+\boldsymbol{T}_C^{-1}\tilde{\boldsymbol{C}}_c^T u_i \\ u_o &= \tilde{\boldsymbol{C}}_v\boldsymbol{X}+\boldsymbol{D}u_i \end{aligned} \tag{29}$$

$$\begin{aligned} s\boldsymbol{X} &= \boldsymbol{T}_C^{-1}\boldsymbol{G}^T\boldsymbol{X}+\boldsymbol{T}_C^{-1}\tilde{\boldsymbol{C}}_v^T i_i \\ i_o &= \tilde{\boldsymbol{C}}_c\boldsymbol{X}+\boldsymbol{D}i_i \end{aligned} \tag{30}$$

We compare (29) and (30) to (24). It is apparent that all those equations are the same provided that $\boldsymbol{A}=\boldsymbol{T}_C^{-1}\boldsymbol{G}$, $\boldsymbol{B}=\boldsymbol{T}_C^{-1}\tilde{\boldsymbol{C}}_c^T$, $\boldsymbol{C}=\tilde{\boldsymbol{C}}_v$ in (29), $\boldsymbol{A}=\boldsymbol{T}_C^{-1}\boldsymbol{G}^T$, $\boldsymbol{B}=\boldsymbol{T}_C^{-1}\tilde{\boldsymbol{C}}_v^T$, $\boldsymbol{C}=\tilde{\boldsymbol{C}}_c$ in (30), and that intrinsic node voltages are chosen as state variables. Note that absence of input capacitors is crucial, since otherwise we would have a complex variable s in the right-hand sides of (29) and (30); in such a case (29) and (30) could not be compared to (24). Due to this fact we can introduce the following definition:

Definition 1. Any G_m-C filter without input capacitors and such that corresponding matrix $\boldsymbol{T}_C$ is invertible is called *a state-space G_m-C filter*. The state variables of such a filter, relating to equations (29)–(30), are just internal node voltages.

Thus, invertibility of matrix $\boldsymbol{T}_C$ and absence of input capacitors are the necessary and sufficient conditions for the existence of the state matrices. From the discussion preceding Definition 1 we obtain that the state matrices of voltage-mode filter denoted as $\boldsymbol{A}_v$, $\boldsymbol{B}_v$, $\boldsymbol{C}_v$, $\boldsymbol{D}_v$ and the state matrices of current-mode filter denoted as $\boldsymbol{A}_c$, $\boldsymbol{B}_c$, $\boldsymbol{C}_c$, $\boldsymbol{D}_c$ are the following:

$$\boldsymbol{A}_v=\boldsymbol{T}_C^{-1}\boldsymbol{G} \quad \boldsymbol{B}_v=\boldsymbol{T}_C^{-1}\tilde{\boldsymbol{C}}_c^T \quad \boldsymbol{C}_v=\tilde{\boldsymbol{C}}_v \quad \boldsymbol{D}_v=\boldsymbol{D} \tag{31}$$

$$\boldsymbol{A}_c=\boldsymbol{T}_C^{-1}\boldsymbol{G}^T \quad \boldsymbol{B}_c=\boldsymbol{T}_C^{-1}\tilde{\boldsymbol{C}}_v^T \quad \boldsymbol{C}_c=\tilde{\boldsymbol{C}}_c \quad \boldsymbol{D}_c=\boldsymbol{D} \tag{32}$$

Note that if the G_m-C filter has only grounded capacitors and no floating nodes, then the corresponding matrix $\boldsymbol{T}_C$ is diagonal and, of course, invertible. In literature (e.g. [7], [26]), only such filters are usually considered as state-space ones. We will refer to them as *direct state-space filters*. Our definition comprises a much wider class of filters. Using Proposition 2 (Section 4) one can easily reformulate Definition 1 to the form: any G_m-C filter for which graph G_t, corresponding to matrix $\boldsymbol{T}_C$ is connected, is called a state-space filter.

Using (31) and (32) we can obtain the following relationships between the state matrices for voltage- and current-mode filter:

$$\boldsymbol{A}_c=\boldsymbol{T}_C^{-1}\boldsymbol{A}_v^T\boldsymbol{T}_C \quad \boldsymbol{B}_c=\boldsymbol{T}_C^{-1}\boldsymbol{C}_v^T \quad \boldsymbol{C}_c=\boldsymbol{B}_v^T\boldsymbol{T}_C \quad \boldsymbol{D}_c=\boldsymbol{D}_v \tag{33}$$

Having (33) we can also derive analogous relations for the observability and controllability matrices $\boldsymbol{W}$ and $\boldsymbol{K}$. Rewriting (28a) for the voltage-mode filter and using (33) we obtain:

$$\boldsymbol{A}_v^T\boldsymbol{W}_v+\boldsymbol{W}_v\boldsymbol{A}_v=-\boldsymbol{C}_v^T\boldsymbol{C}_v \tag{34}$$

which yields:

$$(\boldsymbol{T}_C\boldsymbol{A}_c\boldsymbol{T}_C^{-1})\boldsymbol{W}_v+\boldsymbol{W}_v(\boldsymbol{T}_C^{-1}\boldsymbol{A}_c^T\boldsymbol{T}_C)=-\boldsymbol{T}_C\boldsymbol{B}_c\boldsymbol{B}_c^T\boldsymbol{T}_C \tag{35}$$

$$\boldsymbol{A}_c(\boldsymbol{T}_C^{-1}\boldsymbol{W}_v\boldsymbol{T}_C^{-1})+(\boldsymbol{T}_C^{-1}\boldsymbol{W}_v\boldsymbol{T}_C^{-1})\boldsymbol{A}_c^T=-\boldsymbol{B}_c\boldsymbol{B}_c^T \tag{36}$$

However, rewriting (28b) for the current-mode filter we have:

$$\boldsymbol{A}_c\boldsymbol{K}_c+\boldsymbol{K}_c\boldsymbol{A}_c^T=-\boldsymbol{B}_c\boldsymbol{B}_c^T \tag{37}$$

Thus, we get the following equality:

$$\boldsymbol{K}_c=\boldsymbol{T}_C^{-1}\boldsymbol{W}_v\boldsymbol{T}_C^{-1} \tag{38}$$

A similar reasoning that starts from (54b), yields the following for voltage-mode filter:

$$\boldsymbol{W}_c=\boldsymbol{T}_C\boldsymbol{K}_v\boldsymbol{T}_C \tag{39}$$

Relations (33), (38) and (39) are important in applications. For example, in [32] the authors considered dynamic range (*DR*) of voltage- and current-mode state-space G_m-C filters. It turned out that the general *DR* formulas for a current-mode filter can be (due to relations (33), (38) and (39)) easily expressed in terms of a voltage-mode filter, which significantly facilitates further comparison of both modes of the filter.

It is well known (e.g. [2]) that structures containing only grounded capacitors and no floating nodes (i.e. direct state-space filters in our terminology) are most suitable for monolithic implementation. The reason is that parasitic capacitances can be absorbed into the circuit capacitances in these structures and that grounded capacitors need smaller chip areas than floating ones. General description of G_m-C filters introduced in Section 2 gives an easy way to transform any state-space filter (i.e. containing, in general, also floating capacitors) into direct state-space structure. Below, we describe two 'canonical' transformations from a state-space structure into a direct one.

Let $\hat{\boldsymbol{T}}_C$ be any diagonal matrix with positive elements, i.e. $(\hat{\boldsymbol{T}}_C)_{ii}=\hat{C}_i$, $\hat{C}_i>0$, $i=1,\ldots,n$. Multiplying the first equation in (29) by $\hat{\boldsymbol{T}}_C$, one gets:

$$\begin{aligned} s\hat{\boldsymbol{T}}_C\boldsymbol{X}&=\hat{\boldsymbol{T}}_C\boldsymbol{T}_C^{-1}\boldsymbol{G}\boldsymbol{X}+\hat{\boldsymbol{T}}_C\boldsymbol{T}_C^{-1}\tilde{\boldsymbol{C}}_c^T u_i \\ u_o&=\tilde{\boldsymbol{C}}_v\boldsymbol{X}+\boldsymbol{D}u_i \end{aligned} \tag{40}$$

Introducing the following notation:

$$\hat{\boldsymbol{G}} = \hat{\boldsymbol{T}}_C \boldsymbol{T}_C^{-1} \boldsymbol{G}, \quad \hat{\boldsymbol{C}}_c = \tilde{\boldsymbol{C}}_c \boldsymbol{T}_C^{-1} \hat{\boldsymbol{T}}_C, \quad \hat{\boldsymbol{C}}_v = \tilde{\boldsymbol{C}}_v, \quad \hat{\boldsymbol{D}} = \boldsymbol{D} \tag{41}$$

we can rewrite (40) in the form

$$\begin{aligned} s\hat{\boldsymbol{T}}_C \boldsymbol{X} &= \hat{\boldsymbol{G}}\boldsymbol{X} + \hat{\boldsymbol{C}}_c^T \boldsymbol{u}_i \\ \boldsymbol{u}_o &= \hat{\boldsymbol{C}}_v \boldsymbol{X} + \hat{\boldsymbol{D}} \boldsymbol{u}_i \end{aligned} \tag{42}$$

which is the same as the equations (5) describing the general structure of a voltage-mode G_m-C filter. This means that matrices $\hat{\boldsymbol{T}}_C$, $\hat{\boldsymbol{G}}$, $\hat{\boldsymbol{C}}_c$, $\hat{\boldsymbol{C}}_v$ and $\hat{\boldsymbol{D}}$ determine the new filter structure. Due to the assumptions concerning matrix $\hat{\boldsymbol{T}}_C$, this structure is a direct state-space one and has the same transfer function as the original filter. Indeed:

$$\begin{aligned} \hat{H}(s) &= \hat{\boldsymbol{C}}_v (s\hat{\boldsymbol{T}}_C - \hat{\boldsymbol{G}})^{-1} \hat{\boldsymbol{C}}_c^T + \hat{\boldsymbol{D}} = \tilde{\boldsymbol{C}}_v (s\hat{\boldsymbol{T}}_C - \hat{\boldsymbol{T}}_C \boldsymbol{T}_C^{-1} \boldsymbol{G})^{-1} \hat{\boldsymbol{T}}_C \boldsymbol{T}_C^{-1} \tilde{\boldsymbol{C}}_c^T + \boldsymbol{D} = \\ &= \tilde{\boldsymbol{C}}_v (\boldsymbol{T}_C \hat{\boldsymbol{T}}_C^{-1} (s\hat{\boldsymbol{T}}_C - \hat{\boldsymbol{T}}_C \boldsymbol{T}_C^{-1} \boldsymbol{G}))^{-1} \tilde{\boldsymbol{C}}_c^T + \boldsymbol{D} = \tilde{\boldsymbol{C}}_v (s\boldsymbol{T}_C - \boldsymbol{G})^{-1} \tilde{\boldsymbol{C}}_c^T + \boldsymbol{D} = H(s) \end{aligned} \tag{43}$$

Note that invertibility of matrix $\hat{\boldsymbol{T}}_C$ is crucial in proving above equality. All elements of a new filter structure are given by the appropriate matrix elements, namely: $C_i = (\hat{\boldsymbol{T}}_C)_{ii}$, $G_{mij} = (\hat{\boldsymbol{G}})_{ij}$, $G_{mbi} = (\hat{\boldsymbol{C}}_c)$ and $c_i = (\hat{\boldsymbol{C}}_v)$. It is seen that changing elements $\hat{C}_i$ of matrix $\hat{\boldsymbol{T}}_C$ means changing the values of filter elements without changing its structure. This is very convenient since it makes possible to optimize the filter, e.g. with respect to the sensitivity or the dynamic range.

Now we describe another transformation of a state-space G_m-C filter to direct state-space one. Selecting any diagonal matrix $\check{\boldsymbol{T}}_C$ with positive elements $(\check{\boldsymbol{T}}_C)_{ii} = \check{C}_i$, $\check{C}_i > 0$, $i = 1, \ldots, n$, we can repeat, for the current-mode filter, the procedure presented in the preceding paragraph. Then, matrices $\check{\boldsymbol{G}}$, $\check{\boldsymbol{C}}_c$, $\check{\boldsymbol{C}}_v$ and $\check{\boldsymbol{D}}$, defined as:

$$\check{\boldsymbol{G}} = \boldsymbol{G}\boldsymbol{T}_C^{-1} \check{\boldsymbol{T}}_C, \quad \check{\boldsymbol{C}}_c = \tilde{\boldsymbol{C}}_c, \quad \check{\boldsymbol{C}}_v = \tilde{\boldsymbol{C}}_v \boldsymbol{T}_C^{-1} \check{\boldsymbol{T}}_C, \quad \check{\boldsymbol{D}} = \boldsymbol{D} \tag{44}$$

as well as matrix $\check{\boldsymbol{T}}_C$, determine another structure of the G_m-C filter which is, of course, a direct state-space one. Filter elements are defined similarly as in the previous case.

In order to see better the difference between filter structures generated by the two described above transformation procedures, consider the case when the original filter has only one input transconductor (no input signal distribution) and a trivial output summer, i.e. $\tilde{\boldsymbol{C}}_c = [0 \cdots 0\ G_{mbk}\ 0 \cdots 0]$ and $\tilde{\boldsymbol{C}}_v = [0 \cdots 0\ \ 1\ \ 0 \cdots 0] - 1$ at l-th position. Comparing (41) and (44) one can see that the filter structure determined by matrices $\hat{\boldsymbol{T}}_C$, $\hat{\boldsymbol{G}}$, $\hat{\boldsymbol{C}}_c$, $\hat{\boldsymbol{C}}_v$ and $\hat{\boldsymbol{D}}$ has (in general) input signal distribution and a trivial output summer, however, the structure generated by matrices $\check{\boldsymbol{T}}_C$, $\check{\boldsymbol{G}}$, $\check{\boldsymbol{C}}_c$, $\check{\boldsymbol{C}}_v$ and $\check{\boldsymbol{D}}$ has no input signal distribution and (in general) a non-trivial output summer. This is very convenient, since it gives us an additional degree of freedom while designing the filter for particular application.

It is worth noticing that the described above transformation procedures can be easily generalized. Suppose that $\hat{\boldsymbol{T}}_C$ is any invertible matrix, which is symmetrical, diagonally dominant, with positive diagonal and negative non-diagonal elements. Then, the transformation generated by $\hat{\boldsymbol{T}}_C$ determines a new filter structure, since any $n \times n$ matrix satisfying the above conditions represents the passive network of some state-space filter (see Section 3). Indeed, any non-diagonal element $(\hat{\boldsymbol{T}}_C)_{ij}$, $j > i$, can be interpreted as a floating capacitor and the element $(\hat{\boldsymbol{T}}_C)_{ij} - \sum_{j=1, j\neq i}^{n} (\hat{\boldsymbol{T}}_C)_{ij}$, $i = 1, \ldots, n$ as a grounded capacitor. Other filter elements are given as usual: $G_{mij} = (\hat{\boldsymbol{G}})_{ij}$, $G_{mbi} = (\hat{\boldsymbol{C}}_C)_i$ and $c_i = (\hat{\boldsymbol{C}}_v)_i$. The same is true for the transformation generated by matrix $\check{\boldsymbol{T}}_C$. Thus, we have proved the following result:

Proposition 5. Any state-space Gm-C filter can be transformed (without changing its transfer function) to the structure that has any passive network, provided that the corresponding matrix $\boldsymbol{T}_C$ is invertible. For each such passive network there exist two canonical transformations that yields different state-space filter structures.

The above result creates new possibilities of filter optimization. For instance, given any structure realizing the desired transfer function, by changing the passive network of the filter we can look for another structure with as small number of transconductors as possible. It is also possible to look for design trade-offs concerning complexity of passive and active network.

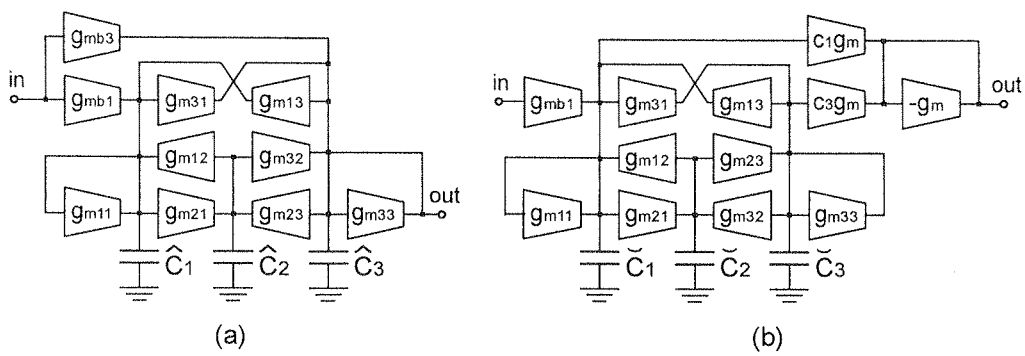


Fig. 4. Third order elliptic filter of Fig. 2 transformed to direct state-space structure by the transformation generated by matrix $\hat{\boldsymbol{T}}_C$ (a) and $\check{\boldsymbol{T}}_C$ (b)

Now we would like to give an example of the state-space filter transformation into the direct state-space structure. Consider the third order elliptic filter shown in Fig. 2 (Section 2). Fig. 4a shows the (voltage-mode) filter structure obtained by means of transformation generated by matrix $\hat{\boldsymbol{T}}_C$. Filter elements are the following: $\hat{C}_i = (\hat{\boldsymbol{T}}_C)_{ii}$, $g_{mij} = (\hat{\boldsymbol{G}})_{ij}$ and $g_{mgi} = (\hat{\boldsymbol{T}}_c)_i$ with matrices $\hat{\boldsymbol{G}}$ and $\hat{\boldsymbol{C}}_c$ calculated according to (41), where matrices $\boldsymbol{T}_C$, $\boldsymbol{G}$ and $\tilde{\boldsymbol{C}}_c$ of the original filter are given by (14). Fig. 4b shows the (voltage-mode) filter structure obtained using the transformation generated by matrix $\check{\boldsymbol{T}}_C$. Filter elements are the following: $\tilde{C}_i = (\check{\boldsymbol{T}}_C)_{ii}$, $g_{mij} = (\check{\boldsymbol{G}})_{ij}$, $g_{mbi} = (\check{\boldsymbol{C}}_c)_i$ and $c_i = \check{\boldsymbol{C}}_v)_i$ with

matrices $\check{G}$, $\check{C}_c$, $\check{C}_v$ calculated according to (44). In particular, matrices $\hat{T}_C$ and $\check{T}_C$ can be identical, which means that both resulting direct state-space structures have the same passive network.

Note that the structure in Fig. 4a has two input transconductors and no output summer (output signal is taken directly from one of the internal nodes), while structure in Fig. 4b has one input transconductor and nontrivial output summer consisting of two transconductors.

5. REDUCIBLE AND QUASI-STATE-SPACE STRUCTURES

In this section we focus our attention on certain interesting classes of G_m-C structures which we call reducible structures and quasi-state-space structures. Suppose that we have G_m-C filter with n internal nodes $x_1, x_2, \ldots, x_n$ and corresponding matrices T_C, G, $\tilde{C}_c$, $\tilde{C}_v$ and D defined by (4a)-(4c). Moreover, let the filter has a floating node, say x_k (recall that floating node is the node not connected to any other node by capacitance elements). It follows that if this k-th node has inner loop (i.e. $G_{mkk} \neq 0$) then node x_k can be removed, i.e. the filter structure can be reduced to the structure with n-1 internal nodes. The reduction does not influence the transmittance of the filter. Below, we give the detailed description of the reduction procedure.

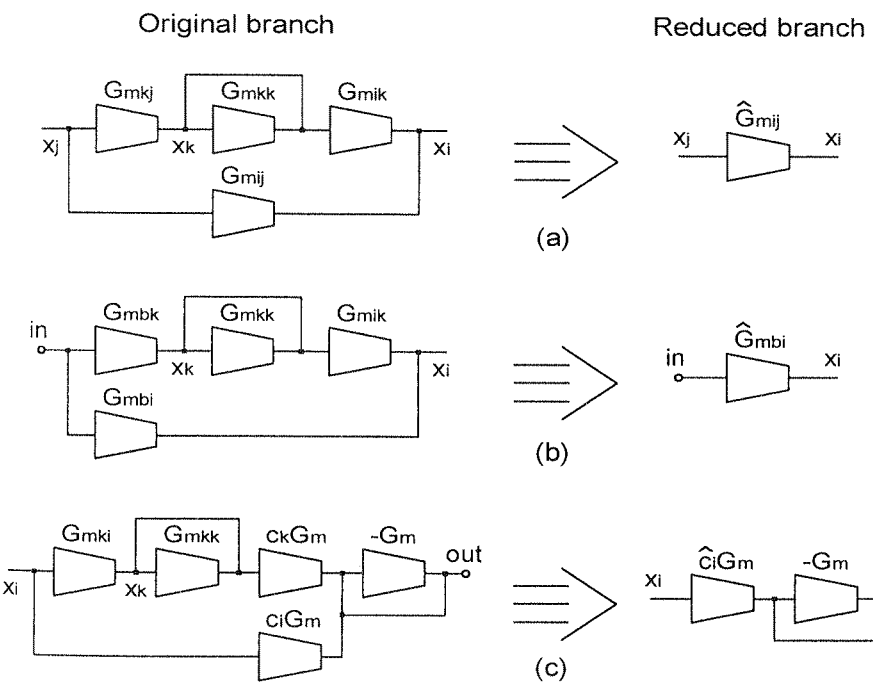


Fig. 5. Filter structure reduction: connection between input node and i-th internal node (a), j-th and i-th internal nodes (b), i-th internal node and output of the filter (c)

Fig. 5a shows the fragment of the active network of the filter concerning connections between input node and i-th internal node ($i \neq k$). It is seen that the current at i-th node due to the voltage u_i at the input of G_{mbi} and G_{mbk} is $u_i(G_{mbk} - G_{mbk}G_{mik}/G_{mkk})$. Thus, the two branches shown in Fig. 5a can be replaced by one transconductor $\hat{G}_{mbi}$ such that:

$$\hat{G}_{mbi} = G_{mbi} - \frac{G_{mbk}G_{mik}}{G_{mkk}} \tag{45}$$

Similar reasoning concerning connections between j-th and i-th internal nodes ($j \neq i \neq k$) and between i-th internal node and output of the filter ($i \neq k$) results in the reduction of appropriate branches shown in Figs. 5b and 5c. The new transconductance value $\hat{G}_{mij}$ and new coefficient $\hat{c}_i$ are the following:

$$\hat{G}_{mij} = G_{mij} - \frac{G_{mkj}G_{mik}}{G_{mkk}} \tag{46}$$

$$\hat{c}_i = c_i - \frac{c_k G_{mik}}{G_{mkk}} \tag{47}$$

Described procedure enables us to eliminate k-th internal node from the filter structure. This results in a new structure with n-1 internal nodes and corresponding matrices $\hat{\boldsymbol{T}}_C$, $\hat{\boldsymbol{G}}$, $\hat{\boldsymbol{C}}_c$, $\hat{\boldsymbol{C}}_v$ and $\hat{\boldsymbol{D}}$. Appropriate matrix elements are given below:

$$(\hat{\boldsymbol{T}}_C)_{ij} = \begin{cases} (\boldsymbol{T}_C)_{ij} & i, j<k \\ (\boldsymbol{T}_C)_{i,j-1} & i<k,\ j\geq k \\ (\boldsymbol{T}_C)_{i-1,j} & i\geq k,\ j<k \\ (\boldsymbol{T}_C)_{i-1,j-1} & i, j\geq k \end{cases} \qquad (\hat{\boldsymbol{G}})_{ij} = \begin{cases} \hat{G}_{m_{ij}} & i, j<k \\ \hat{G}_{m_{i,j-1}} & i<k,\ j\geq k \\ \hat{G}_{m_{i-1,j}} & i\geq k,\ j<k \\ \hat{G}_{m_{i-1,j-1}} & i, j\geq k \end{cases} \tag{48a}$$

$$(\hat{\boldsymbol{C}}_c)_i = \begin{cases} \hat{G}_{mb_i} + sC_{b_i} & i<k \\ \hat{G}_{mb_{i-1}} + sC_{bi-1} & i\geq k \end{cases} \qquad (\hat{\boldsymbol{C}}_v)_i = \begin{cases} \hat{c}_i & i<k \\ \hat{c}_{i-1} & i\geq k \end{cases} \qquad \hat{\boldsymbol{D}} = d \tag{48b}$$

It is seen that due to removing k-th node, some shift of indexes is necessary. Obviously, if the original filter structure has more than one floating node with inner loop, then all such nodes can be removed by iteration of described procedure.

On the basis of the above considerations we can formulate the formal definitions:

Definition 2. The G_m-C filter structure is called *reducible* if it has at least one floating node x_k with inner loop (i.e. $G_{mkk} \neq 0$).

Definition 3. The G_m-C filter structure is called *completely reducible* if each of its floating nodes has inner loop.

Reduction procedure can be helpful in computer aided synthesis of G_m-C filters since it allows to remove some redundancy in filter structure. Note, however that reduction is not always desirable — more complex structures are often used for their

flexibility (there exist universal structures making possible independent control of filter poles and zeros, e.g. [2]). Note also that some design techniques, e.g. Bruton Transformation [2], lead to the structures which are not reducible.

Now, we would like to introduce another class of G_m-C structures.

Definition 4. The G_m-C filter structure is called *quasi-state-space* structure if it has no input capacitors (i.e. $C_{bi} = 0$, $i = 1, \ldots, n$) and matrix $\boldsymbol{T}_C$ corresponding to filter's passive network is not invertible but each block of $\boldsymbol{T}_C$ is invertible.

Although for quasi-state-space filters do not exists state matrices, it follows that such structures exhibits similar properties as state-space structures. In particular, any quasi-state-space filter can be transformed to the direct structure (i.e. having only grounded capacitors). To show this, suppose that $n \times n$ matrix $\boldsymbol{T}_C$ has the following structure:

$$\boldsymbol{T}_C = \begin{bmatrix} (\tilde{\boldsymbol{T}}_C)_1 & & & & 0 \\ & \ddots & & & \\ & & (\tilde{\boldsymbol{T}}_C)_k & & \\ & & & 0 & \\ & & & & \ddots \\ 0 & & & & 0 \end{bmatrix} \tag{49}$$

where $(\tilde{\boldsymbol{T}}_C)_i$, $i = 1, \ldots, k$ are invertible blocks of sizes $n_i \times n_i$ and the rest of elements are zeros. Let $n^{\#} = n_1 + n_2 + \ldots + n_k$. Note, that matrix $\boldsymbol{T}_C$ of any quasi-state-space filter has this structure up to renumbering of indexes. Let $\boldsymbol{T}_C^{\#}$ be the matrix defined as below:

$$\boldsymbol{T}_C^{\#} = \begin{bmatrix} (\tilde{\boldsymbol{T}}_C)_1 & & & & 0 \\ & \ddots & & & \\ & & (\tilde{\boldsymbol{T}}_C)_k & & \\ & & & 1 & \\ & & & & \ddots \\ 0 & & & & 1 \end{bmatrix} \tag{50}$$

Then, $\boldsymbol{T}_C^{\#}$ is invertible and $(\boldsymbol{T}_C^{\#})^{-1} \cdot \boldsymbol{T}_C = \boldsymbol{T}_C \cdot (\boldsymbol{T}_C^{\#})^{-1} = \boldsymbol{I}^{\#}$, where $\boldsymbol{I}^{\#}$ is diagonal matrix such that $\boldsymbol{I}_{ii}^{\#} = 1$ for $1 < i \le n^{\#}$ and $\boldsymbol{I}_{ii}^{\#} = 0$ for $n^{\#} < i \le n$. Now, multiplying equation (5) from the left by $(\boldsymbol{T}_C^{\#})^{-1}$ we obtain an analog of equation (29) for quasi-state-space filters:

$$\begin{aligned} s\boldsymbol{I}^{\#}\boldsymbol{X} &= (\boldsymbol{T}_C^{\#})^{-1}\boldsymbol{G}\boldsymbol{X} + (\boldsymbol{T}_C^{\#})^{-1}\tilde{\boldsymbol{C}}_c^{T} u_i \\ u_o &= \tilde{\boldsymbol{C}}_v \boldsymbol{X} + \boldsymbol{D} u_i \end{aligned} \tag{51}$$

Let $\hat{\boldsymbol{T}}_C^{\#}$ be any diagonal matrix such that, i.e. $(\hat{\boldsymbol{T}}_C^{\#})_{ii} = \hat{C}_i$, $\hat{C}_i > 0$, for $1 \le i \le n^{\#}$ and $(\hat{\boldsymbol{T}}_C^{\#})_{ii} = 1$ for $n^{\#} < i \le n$. Multiplying first equation in (51) by $\hat{\boldsymbol{T}}_C^{\#}$ one gets:

$$s\hat{\boldsymbol{T}}_C^{\#\#}\boldsymbol{X} = \hat{\boldsymbol{T}}_C^{\#}(\boldsymbol{T}_C^{\#})^{-1}\boldsymbol{G}\boldsymbol{X} + \hat{\boldsymbol{T}}_C^{\#}(\boldsymbol{T}_C^{\#})^{-1}\check{\boldsymbol{C}}_c^{T}u_i$$
$$u_o = \tilde{\boldsymbol{C}}_v\boldsymbol{X} + \boldsymbol{D}u_i \tag{52}$$

where $\hat{\boldsymbol{T}}_C^{\#\#} = \hat{\boldsymbol{T}}_C^{\#}\boldsymbol{I}^{\#}$ is diagonal matrix such that $(\hat{\boldsymbol{T}}_C^{\#\#})_{ii} = \hat{C}_i$, $\hat{C}_i > 0$, for $1 \le i \le n^{\#}$ and $(\hat{\boldsymbol{T}}_C^{\#\#})_{ii} = 0$ for $n^{\#} < i \le n$. It is seen, when comparing (52) to (5), that $\hat{\boldsymbol{T}}_C^{\#\#}$ and matrices $\hat{\boldsymbol{G}}_C^{\#}$, $\hat{\boldsymbol{C}}_c^{\#}$, $\hat{\boldsymbol{C}}_v^{\#}$ and $\hat{\boldsymbol{D}}^{\#}$ defined as:

$$\hat{\boldsymbol{G}}^{\#} = \hat{\boldsymbol{T}}_C^{\#}(\boldsymbol{T}_C^{\#})^{-1}\boldsymbol{G}, \quad \hat{\boldsymbol{C}}_c^{\#} = \tilde{\boldsymbol{C}}_c(\boldsymbol{T}_C^{\#})^{-1}\hat{\boldsymbol{T}}_C^{\#}, \quad \hat{\boldsymbol{C}}_v^{\#} = \tilde{\boldsymbol{C}}_v, \quad \hat{\boldsymbol{D}}^{\#} = \boldsymbol{D} \tag{53}$$

determine new structure which is a direct (quasi-state-space) one. Of course, elements $\hat{C}_i$ are new grounded capacitances of the filter. Note that conclusions comprised in the discussion following (42) remain valid in this case. In particular, transmittance of the original and transformed filters are the same, since $\hat{\boldsymbol{T}}_C^{\#}$ is invertible matrix.

Similarly as in Section 5, we can define second transformation starting from description of the current-mode filter. If $\check{\boldsymbol{T}}_C^{\#}$ is any diagonal matrix such that, i.e. $(\check{\boldsymbol{T}}_C^{\#})_{ii} = \check{C}_i$, $\check{C}_i > 0$, for $1 \le i \le n^{\#}$ and $(\check{\boldsymbol{T}}_C^{\#})_{ii} = 1$ for $n^{\#} < i \le n$, then a new filter structure is determined by the following matrices:

$$\check{\boldsymbol{T}}_C^{\#\#} = \check{\boldsymbol{T}}_C^{\#}\boldsymbol{I}^{\#}, \quad \check{\boldsymbol{G}}^{\#} = \boldsymbol{G}(\boldsymbol{T}_C^{\#})^{-1}\check{\boldsymbol{T}}_C^{\#}, \quad \check{\boldsymbol{C}}_v^{\#} = \tilde{\boldsymbol{C}}_v, \quad \check{\boldsymbol{C}}_v^{\#} = \tilde{\boldsymbol{C}}_v(\boldsymbol{T}_C^{\#})^{-1}\check{\boldsymbol{T}}_C^{\#}, \quad \check{\boldsymbol{D}}^{\#} = \boldsymbol{D} \tag{54}$$

The above transformations can be generalized similarly as in the state-space filters case, i.e. diagonal matrix $\hat{\boldsymbol{T}}_C^{\#}$ can be replaced by any matrix $\hat{\boldsymbol{T}}_C^{\#}$ such that its $n^{\#} \times n^{\#}$ submatrix is symmetrical, diagonally dominant, having positive diagonal and negative non-diagonal elements and $(\hat{\boldsymbol{T}}_C^{\#})_{ij} = 1$ for $i, j > n^{\#}$. By the argument similar to that given in paragraph preceding Proposition 5, transformation generated by $\hat{\boldsymbol{T}}_C^{\#}$ determine new filter structure with passive network corresponding to matrix $\hat{\boldsymbol{T}}_C^{\#\#} = \hat{\boldsymbol{T}}_C^{\#}\boldsymbol{I}^{\#}$ and the rest of elements given by $G_{mij} = (\hat{\boldsymbol{G}}^{\#})_{ij}$, $G_{mbi} = (\hat{\boldsymbol{C}}_c^{\#})_i$ and $c_i = (\hat{\boldsymbol{C}}_v^{\#})$. We have also the result analogous to Proposition 5:

Proposition 6. Any quasi-state-space G_m-C filter can be transformed (without changing its transfer function) to the structure having any passive network provided that corresponding matrix $\boldsymbol{T}_C$ has the structure such as described in the preceding paragraph. For each such passive network there exist two canonical transformations that give different quasi-state-space filter structures.

We end this section with the remark that every quasi-state-space G_m-C filter that is completely reducible can be reduced to the state-space structure. Indeed, let the filter has k floating nodes, i.e. graph G_t corresponding to its passive network has $k+1$ connected components. Removing all floating nodes causes that graph $\tilde{G}_t$ representing passive network of the reduced filter becomes connected, hence corresponding matrix $\tilde{\boldsymbol{T}}_C$ is invertible. This means that the reduced filter is a state-space one.

6. CONCLUSION

In the paper, a novel and general approach to G_m-C filters based on matrix description has been presented. There have been shown general transfer function formulas for any G_m-C filter structure. We have investigated relationships between the passive network of the filter and its transfer function as well as conditions of invertibility of matrix corresponding to the filter's passive network. Based on these considerations we have given a new definition of state-space G_m-C filters and investigated their properties. In particular, we formulated relations between state matrices for voltage- and current-mode state-space G_m-C filters and defined two canonical transformations of state-space filters into direct state-space ones, i.e. those having only grounded capacitors. It has been shown that any state-space filter can be realized with any passive network (provided that the corresponding matrix is invertible). All the results presented in the paper can be used in computer aided filter synthesis and optimization. The reason is that matrix description can be easily handled by computer. Moreover, based on matrix description of G_m-C filter one can derive general formulas for sensitivity function of any G_m-C filter structure. The reduction procedures discussed in Section 5 can be used in automated filter design to simplify filter structures generated by computer programs within some design strategies.

It seems that the presented consistent approach to G_m-C filters can be interesting not only from a theoretical point of view but also it may be useful for filter designers. The matter of future work is to construct tools for automatic filter design based on the results drawn in this paper, with special care to the sensitivity optimization.

7. REFERENCES

1. R. Schaumann, M. S. Ghausi, K. R. Laker: *Design of Analog Filters, Passive, Active RC, and Switched Capacitor*. Englewood Cliff, NJ: Prentice-Hall, 1990.
2. T. Deliyannis, Y. Sun, J. K. Fidler: *Continuous-time active filter design*. CRC Press, USA, 1999.
3. C. Toumazou, J. Lidgey, D. Haigh, Eds.: *Analogue IC Design — The Current-Mode Approach*. London, U.K.: IEE, Apr. 1990.
4. R. L. Geiger, E. Sánchez-Sinencio: *Active filter design using operational transconductance amplifiers: A tutorial*. IEEE Circuit and Devices Mag., vol. 1, pp. 20–32, Mar. 1985.
5. Sánchez-Sinencio, R. L. Geiger, H. Nevarez-Lozano: *Generation of Continuous-Time Two Integrator Loop OTA Filter Structures*, IEEE Trans. Circuits Syst., vol. 35, pp. 936–946, Aug. 1988.
6. M. A. Tan, R. Schaumann: *Design of a General Biquadratic Filter Section with Only Transconductances and Grounded Capacitors*. IEEE Trans. Circuits Syst.-II, vol. 35, pp. 478–480, Apr. 1988.
7. G. Groenewold: *Optimal Dynamic Range Integrated Continuous-Time Filters*. Delft, The Netherlands: Delft Univ. Press, 1992.
8. G. Groenewold: *The Design of High Dynamic Range Continuous-Time Integratable Bandpass Filters*. IEEE Trans. Circuits and Syst., vol, 38, No.8, pp. 838-852, Aug. 1991.
9. B. Nauta: *Analog CMOS filters for very high frequencies*. Kluwer Academic Publishers, 1993.
10. J. Silva-Martinez, M. S. J. Steyaert, W. Sansen: *A 10.7MHz 68-dB CMOS continuous-time filter with on-chip automating tuning*. IEEE J. Solid-State Circuits, vol. 27, No. 12, pp. 1843–1853, 1992.

11. R. Allini, A. Baschirotto, R. Castello: *Tuneable BiCMOS continuous-time filter for high-frequency applications*. IEEE J. Solid-State Circuits, 27, (12), pp.1905–1915, 1992.
12. Y. P. Tsividis: *Integrated continuous-time filter design — An overview*. IEEE J. Solid-State Circuits, vol. 29, pp. 166–176, Mar. 1994.
13. S. Szczepanski, J. Jakusz, R. Schaumann: *A linear CMOS OTA for VHF applications*. IEEE Trans. Circuits Syst.-II, vol.44, pp. 174–187, Mar. 1997.
14. J. Mahattanakul, C. Toumazou: *Current-Mode Versus Voltage-Mode Gm-C Biquad Filters: What the Theory Says*. IEEE Trans. Circuits Syst.-II, vol. 45, pp. 173–186, Feb. 1998.
15. J. Glinianowicz, J. Jakusz, S. Szczepanski, Y. Sun: *High-frequency two-input CMOS OTA for continuous-time filter applications*. IEE Proc.-Circuits Dev. Syst., vol.147, No.1, pp. 13–18, Feb. 2000.
16. D. H. Chiang, R. Schaumann: *Performance comparison of high-order IFLF and cascade analogue integrated lowpass filters*., IEE Proc.-Circ. Dev. Syst., vol.147, No.1, pp. 19–27, Feb. 2000.
17. C. C. Hsu, W. S. Feng: *Structural generation of current-mode filters using tunable multiple-output OTAs and grounded capacitors*. IEICE Trans. Fund. Elect. Comm. Comp. Sc. Vol. E83A, No. 9, pp. 1778–1785, Sep. 2000,
18. J. Ramirez-Angulo, M. Robinson, E. Sanchez-Sinencio: *Current-mode continuous-time filters: Two design approaches*. IEEE Trans. Circuits Syst.-II, vol.39, pp. 337–341, June, 1992.
19. S.-S. Lee, R. H. Zele, D. J. Allstot, G. Liang: *A continuous-time current-mode integrator*., IEEE Trans. Circuit Syst., vol. 38, pp. 1236–1238, Oct. 1991.
20. S.-S. Lee, R. H. Zele, D. J. Allstot, G. Liang: *CMOS continuous-time current-mode filters for high-frequency applications*. IEEE J. Solid-State Circuits, vol. 28, pp. 323–329, Mar. 1993.
21. S. L. Smith, E. Sanchez-Sinencio: *Low voltage integrators for high-frequency CMOS filters using current-mode techniques*. IEEE Trans. Circuits Syst.-II, vil. 43, pp. 39–48, Jan. 1996.
22. T. Kaczorek: *Wektory i macierze w automatyce i elektrotechnice*. Warszawa, WNT, 1998.
23. G. W. Roberts, A. S. Sedra: *All current-mode frequency selective circuits*. Electron. Lett., vol. 25, pp. 759–761, June 1989.
24. J. D. Schoeffler: *The synthesis of minimum sensitivity networks*. IEEE Trans. Circuit Theory, vol. 11, pp. 271–276, 1964.
25. W. M. Snelgrove, A. S. Sedra: *Synthesis and Analysis of State-Space Active Filters Using Intermediate Transfer Functions*. IEEE Trans. Circuits Syst., vol.CAS-33, pp. 287–301, Mar. 1986.
26. R. Mackay, A.S. Sedra: *Generation of low-sensitivity state-space active filters*. IEEE Trans. Circuit Syst., vol. CAS-27, pp. 863–870, Oct. 1980.
27. G. A. De Veirman, R. G. Yamasaki: *Design of a bipolar 10 MHz programmable continuous-time 0.05(equiripple linear phase filter*. IEEE J. Solid-State Circuits, vol. 27, pp. 324–331, Mar. 1992.
28. H. Khorramabadi, M. J. Tarsie, N. S. Woo: *Baseband filters for IS-95 CDMA receiver applications featuring digital automatic frequency tuning*, in int. Solid State Circuits Conf., San Francisco, pp. 172–173, 1996.
29. W. Dehaene, M. S. J. Steyaert, W. SansenP: *A 50-MHz Standard CMOS Pulse Equalizer for Hard Disk Read Channels* IEEE J. Solid-State Circuits, vol. 32, pp. 977–988, July 1997.
30. C. A. Laber, P. R. Gray: *A 20-MHz Sixth-Order BiCMOS Parasitic-Insensitive Continuous-Time Filter and Second-Order Equalizer Optimized for Disk-Drive Read Channels* IEEE J. Solid-State Circuits, vol. 28, pp. 462-469, Mar. 1993.
31. P. K. D. Pai, A. A. Abidi: *A 40-mW 55 Mb/s CMOS Equalizer for Use in Magnetic Storage Read Channels*. IEEE J. Solid-State Circuits, vol. 29, pp. 489–499, Mar. 1994.
32. S. Koziel, S. Szczepański: *Dynamic Range Comparison of Voltage-Mode and Current-Mode State-Space Gm-C Biquad Filters in Reciprocal Structures, in Proc.* IEEE Int. Conf. Elect. Circuits Syst., vol. II, pp. 819–822, 2001.
33. L. O. Chua, P.-M. Lin: *Computer-Aided Analysis of Electronic Circuits. Algorithms and Computational Techniques*. Englewood Cliff, NJ: Prentice-Hall, 1975.

S. KOZIEŁ, S. SZCZEPAŃSKI

ANALIZA OGÓLNYCH STRUKTUR FILTRÓW G_m-C ZMIENNYCH STANU CZASU CIĄGŁEGO

Streszczenie

W pracy przedstawiono nowe podejście do ważnej i szeroko stosowanej klasy filtrów aktywnych czasu ciągłego, wykorzystujących wzmacniacze transkonduktancyjne i pojemności (tzw. filtry G_m-C). Zaproponowano ogólny opis macierzowy filtrów G_m-C, który obejmuje wszystkie możliwe realizacje filtrów w rozważanej klasie elementów. Zaletą tego opisu jest między innymi to, że wszystkie macierze występujące w równaniach mogą być tworzone bezpośrednio przez wgląd w schemat układu. Podano ogólne wzory określające transmitancję filtru, a także wykazano wzajemną odwracalność struktur napięciowych i prądowych. Jakkolwiek jest to cecha powszechnie znana, w ramach przedstawianego podejścia otrzymuje się ją jako prosty i bezpośredni wniosek z ogólnych równań opisujących filtr. W dalszej kolejności, przedstawiono zależności pomiędzy strukturą pasywną (tj. pojemnościową) filtru a rzędem transmitancji filtru. Sformułowano szereg twierdzeń ustalających warunki konieczne i dostateczne odwracalności macierzy odpowiadającej sieci pasywnej filtru, a także górne ograniczenia rzędu transmitancji filtru w zależności od spójności grafu opisującego sieć pasywną.

W drugiej części pracy wprowadzono nową definicję filtrów zmiennych stanu, zgodnie z którą filtrem zmiennych stanu jest taki filtr, dla którego macierz odpowiadająca sieci pasywnej jest odwracalna, co jest równoważne istnieniu macierzy zmiennych stanu dla takiej struktury. Jest to istotne uogólnienie istniejących definicji, ponieważ dotychczas, za filtry zmiennych stanu uważano te, które posiadają wyłącznie uziemione pojemności. Zgodnie z nową definicją do podklasy filtrów zmiennych stanu należy również szereg struktur zawierających pojemności nieuziemione. Pozwala to na użycie aparatu opisu zmiennych stanu w odniesieniu do szerszej klasy filtrów. Zaproponowano również dwa kanoniczne przekształcenia umożliwiające transformację dowolnego filtru zmiennych stanu do struktury zawierającej wyłącznie pojemności uziemione, co może mieć znaczenie w przypadku realizacji monolitycznych. Wykazano również, że każdy filtr zmiennych stanu może być przetransformowany do struktury zawierającej dowolną sieć pasywną (o ile tylko odpowiadająca macierz jest odwracalna).

W pracy wprowadzono również kolejne dwie klasy filtrów, tzw. filtry redukowalne oraz filtry pseudo — zmiennych stanu. Pierwsza z nich obejmuje struktury zawierające węzły wewnętrzne nie dołączone do żadnej pojemności w układzie ale zawierające wewnętrzną pętlę transkonduktancyjną. Struktury tego typu można zredukować poprzez usunięcie takich węzłów bez zmiany charakterystyk filtru, modyfikując jednocześnie wartości elementów układu. Druga klasa filtrów, to struktury o nieodwracalnej macierzy podukładu pasywnego, która ma tę własność, że jej dowolna kwadratowa podmacierz o niezerowych elementach diagonalnych jest odwracalna. Pokazano, że struktury takie mają w pewnej mierze własności analogiczne do filtrów zmiennych stanu. Należy także zauważyć, że podklasa struktur pseudo — zmiennych stanu zawiera nietrywialne przykłady filtrów, np. realizacje układowe powstające przy użyciu transformacji Brutona.

Wyniki zawarte w omawianej pracy mogą być one zastosowane do automatycznego projektowania i analizy filtrów G_m-C. W szczególności, należy zauważyć, że opis macierzowy może być łatwo implementowany w postaci programu komputerowego, podobnie jak wszelkie transformacje i przekształcenia realizowane z jego pomocą.

Słowa kluczowe: filtry zmiennych stanu, filtry G_m-C, opis macierzowy, reprezentacja zmiennych stanu

Tran
j

gram
dzy j
jakie
trans
punk
powo
równ
uwag

Słow

Żyjem
każdej dz
częściej
intelektu
posunąć
tomatyzo
zastąpieni
stąpiono j

Aby
wiedzy z

KWARTALNIK ELEKTRONIKI I TELEKOMUNIKACJI, 2002, 48, z. 3–4, ss. 523–551

Translacja automatyczna – podejścia oparte na koncepcji języka interlingua i na przykładach translacyjnych

MIROSŁAW GAJER

Katedra Automatyki AGH
al. Mickiewicza 30, 30-059 Kraków
e-mail: mgajeria@agh.edu.pl

Otrzymano 2001.11.12
Autoryzowano 2002.02.28

Translacja automatyczna jest dyscypliną nauki dostarczającą wiedzy o tym, jak programować komputery, aby były one w stanie dokonywać automatycznych przekładów pomiędzy językami naturalnymi. Translacja automatyczna była również jedną z pierwszych aplikacji, jakie zostały zaproponowane dla komputerów. Niestety, szybko okazało się, że zadanie translacji automatycznej jest znacznie trudniejsze, ale zarazem o wiele ciekawsze z naukowego punktu widzenia, niż pierwotnie sądzono. W artykule omówiono podstawowe przyczyny powodujące, że translacja automatyczna jest zadaniem tak niezwykle trudnym. Omówiono również najbardziej obiecujące kierunki rozwoju systemów translacji automatycznej. Osobną uwagę poświęcono również systemom translacji automatycznej w Polsce.

Słowa kluczowe: translacja automatyczna, lingwistyka komputerowa, przetwarzanie języków naturalnych

1. WPROWADZENIE

Żyjemy obecnie w epoce informatyzacji, komputery wkraczają bowiem do niemalże każdej dziedziny naszej działalności. Tak zwane systemy sztucznej inteligencji coraz częściej próbują, z lepszym bądź gorszym skutkiem, naśladować wybrane cechy intelektu człowieka. Powstaje, w związku z powyższym, ważne pytanie, jak dalece może posunąć się ten proces? Czy wszystkie dziedziny działalności człowieka można zautomatyzować wprowadzając do nich komputery? W szczególności, czy możliwe jest zastąpienie programem komputerowym człowieka tłumacza, tak jak na przykład zastąpiono już skutecznie programem grającym w szachy człowieka szachistę?

Aby odpowiedzieć na to pytanie należy przyjrzeć się bliżej pasjonującej dziedzinie wiedzy zwanej translacją automatyczną (ang. machine translation).

2. HISTORIA TRANSLACJI AUTOMATYCZNEJ

Celem badaczy zajmujących się translacją automatyczną jest opracowanie programów komputerowych, które byłyby w stanie dokonywać przekładów tekstów zapisanych w pewnym języku naturalnym, np. włoskim, na wybrany inny język naturalny, np. bułgarski. Wbrew pozorom, translacja automatyczna nie jest wcale młodą dziedziną wiedzy – jest prawie równie stara, jak sam wynalazek cyfrowego komputera. Za pioniera translacji automatycznej uznawany jest powszechnie Waren Weaver, który w 1949 roku wystosował do amerykańskiej fundacji wspierającej naukę (Rockefeller Foundation) memorandum, w którym postulował podjęcie na szeroką skalę badań w tej dziedzinie. Istotnie, memorandum Weavera odniosło pewne skutki, ponieważ dość szybko podjęto badania w zaproponowanym przez niego kierunku [1].

Pierwszy publiczny pokaz systemu komputerowego, który przetłumaczył z języka rosyjskiego na język angielski 49 prostych i wybranych odpowiednio wcześniej zdań, odbył się już w 1954 roku. Możliwości tego systemu były oczywiście bardzo ograniczone, podobnie zresztą, jak ograniczone były możliwości ówczesnych komputerów. Rozważany system korzystał z rosyjsko-angielskiego słownika o pojemności zaledwie 250 haseł oraz z jedynie 16 prostych reguł gramatycznych [2].

Ze wczesnymi latami badań nad translacją automatyczną wiąże się również pewna anegdota. Otóż podobno skonstruowano w Stanach Zjednoczonych system, który tłumaczył z języka angielskiego na język rosyjski i w odwrotnym kierunku. Systemowi temu zadano do przetłumaczenia zdanie:

The spirit is willing but the body weak.

czyli po polsku:

Duch wprawdzie ochoczy, ale ciało mdłe.

System automatycznej translacji przetłumaczył to zdanie na język rosyjski (już mniejsza z tym jak). Następnie rosyjski przekład zadano na wejście systemu tłumaczącego z języka rosyjskiego na język angielski i w rezultacie otrzymano zdanie:

The vodka is good but the meat is rotten.

co znaczy po polsku:

Wódka jest wprawdzie dobra, ale mięso jest zepsute.

Mimo wszystko, począwszy od tego pionierskiego możliwości systemów automatycznej translacji zaczęły systematycznie wzrastać. W związku z tym dość powszechnie zaczęła panować wówczas opinia, że stworzenie komputerowego programu, który byłby w stanie tłumaczyć teksty techniczne jest kwestią co najwyżej kilkunastu lat.

Niestety po pierwszych spektakularnych sukcesach, szybko jednak okazało się, że zautomatyzowanie procesu translacji pomiędzy językami naturalnymi jest zadaniem o wiele ciekawszym i niestety znacznie trudniejszym niż pierwotnie sądzono. Taki stan rzeczy spowodował pojawienie się wielu krytycznych wystąpień osób, które w ogóle

kwestionowały sens prowadzenia jakichkolwiek dalszych badań nad translacją automatyczną. Najsilniej z krytyką wystąpił w 1959 roku izraelski filozof Bar Hillel, który przedstawił dosyć silne argumenty na poparcie głoszonej przez siebie tezy, w myśl której zautomatyzowanie translacji w taki sposób, aby programy komputerowe produkowały przekłady o jakości porównywalnej z jakością pracy tłumaczy profesjonalistów, nie jest możliwe, i to nie tylko z powodu jakichś przejściowych ograniczeń natury technicznej, ale z zasady [3].

Wystąpienie Bar-Hillela zaowocowało ostatecznie wydaniem w 1964 roku przez Amerykańską Akademię Nauk raportu, nazwanego później raportem ALPAC (Automatic Language Processing Advisory Committee), który nakazywał wstrzymanie finansowania badań nad translacją automatyczną. Wskutek tego zainteresowanie badaczy tą dziedziną informatyki stosowanej zmalało praktycznie do zera na dobrych kilkanaście lat [4].

Renesans nastąpił dopiero pod koniec lat siedemdziesiątych. Pierwszym systemem translacji automatycznej, który odniósł duży sukces rynkowy był opracowany w 1977 roku w Kanadzie system METEO. Zadanie stawiane przed systemem METEO ograniczało się tylko i wyłącznie do jednej wybranej dziedziny zastosowań – tłumaczenia raportów i komunikatów meteorologicznych z języka angielskiego na język francuski. System ten pracuje do chwili obecnej, tłumacząc dziennie teksty o objętości około 50 tysięcy słów, które ukazują się już bez żadnych poprawek i przeróbek we francuskojęzycznej prasie, radiu i telewizji. Począwszy od końca lat siedemdziesiątych zainteresowanie translacją automatyczną wzrastało systematycznie na całym świecie, zwłaszcza w Japonii, Stanach Zjednoczonych i w Europie Zachodniej (uprzednio niezwykle intensywne badania prowadzone były również w Związku Radzieckim).

3. TEST TURINGA DLA SYSTEMÓW TRANSLACJI AUTOMATYCZNEJ

Osobiście uważam, iż śmiało można zaryzykować twierdzenie, że translacja automatyczna jest najtrudniejszym spośród wszystkich kierunków badań prowadzonych w ramach systemów sztucznej inteligencji. Ponadto uważam, że pomyślne rozwiązanie zagadnienia automatycznej translacji dla języka ogólnego, na poziomie człowieka tłumacza profesjonalisty, otwarłoby automatycznie furtkę do niezwykle burzliwego postępu w innych dziedzinach sztucznej inteligencji. Między innymi, z tego właśnie względu prowadzenie badań nad zautomatyzowaniem procesu translacji wydaje się być tak niezmiernie ważne. Oprócz oczywistych konsekwencji dla rozwoju nauki, pojawienie się istotnego przełomu w badaniach nad translacją automatyczną, miałoby również doniosłe konsekwencje natury politycznej, socjologicznej i filozoficznej. Z tego właśnie powodu translacja automatyczna wydaje się być niezwykle ciekawą, wręcz pasjonującą, interdyscyplinarną dziedziną wiedzy [4].

Warto w tym kontekście wspomnieć o tzw. teście Turinga, który pozwalałby na rozstrzygniecie dylematu, czy system sztucznej inteligencji osiągnął stopień sprawności umysłowej porównywalnej z intelektem przeciętnego człowieka. Warto również za-

stanowić się nad licznymi trudnościami, jakie mogą wystąpić podczas praktycznej próby realizacji tego testu. W szczególności komputer podszywający się pod człowieka musiałby chwilami ukrywać swoje rzeczywiste możliwości, w których przewyższa człowieka, tylko po to, aby się po prostu nie zdradzić.

Osobiście chciałbym zaproponować prostszą, aczkolwiek równie skuteczną, nową wersję testu Turinga. W teście tym komputerowi zostałby zadany pewien tekst, napisany w ustalonym języku naturalnym (np. czeskim), po czym komputer zostałby poproszony o dokonanie przekładu tego tekstu na wybrany inny język naturalny (np. duński) – oczywiście zakładamy, że komputer posiada w swych zasobach wbudowaną wiedzę o obu wymienionych wyżej językach. Następnie uzyskany w ten sposób przekład, wraz z tekstem oryginału, zostałby przedstawiony kompetentnej osobie (najlepiej tłumaczowi przysięgłemu z języka czeskiego na duński). Jeżeli osoba taka nie byłaby w stanie stwierdzić, czy przekład ten jest dziełem człowieka, czy też pochodzi z komputera, wówczas oznaczałoby to, że rozważany system komputerowy przeszedł zaproponowaną nową wersję testu Turinga – podczas realizacji przekładu pomiędzy językami naturalnymi był równie inteligentny jak człowiek tłumacz profesjonalista!

Trzeba od razu zaznaczyć, że do chwili obecnej nie powstał jeszcze żaden komputerowy program translacji automatycznej, w przypadku którego można by było pokusić się nawet o poddanie go opisanej wcześniej nowej wersji Turinga. Wręcz przeciwnie, współczesne programy translacji automatycznej, przewidziane dla języka ogólnego, który nie jest w jakiś sposób ograniczony do zadanej z góry dziedziny wiedzy (jak np. w wymienionym wcześniej systemie METEO), dają rezultaty bardzo odległe od zamierzonego ideału (jedymy wyjątek w tym względzie mogą stanowić systemy zastosowane do tłumaczenia pomiędzy dwoma językami bardzo blisko ze sobą spokrewnionymi). Co gorsze, nic nie wskazuje, aby w najbliższym czasie mógł nastąpić w tej dziedzinie jakiś znaczący postęp, aczkolwiek intensywnie prowadzone badania trwają w wielu przodujących ośrodkach światowych.

Co zatem sprawia, że translacja automatyczna jest tak trudna i dlaczego z takim niesłychanym oporem i z taką zdumiewającą skutecznością, dziedzina ta broni się przed poddaniem algorytmizacji?

Przecież istnieją już na przykład programy komputerowe, które potrafią realizować tak, wydawałoby się, niebanalne zadanie, jak gra w szachy, równie dobrze jak najlepsi światowi szachiści. Czy zatem rezultatów osiągniętych podczas udoskonalania algorytmów gry w szachy nie można by jakość przenieść na grunt translacji automatycznej?

4. CZY AUTOMATYZACJA PROCESU TRANSLACJI JEST W OGÓLE MOŻLIWA?

Istnieje wiele różnorodnych powodów faktu, że proces translacji pomiędzy językami naturalnymi z tak wielkimi oporami poddaje się implementacji komputerowej. Pierwszym i najważniejszym powodem jest wieloznaczność wypowiedzi, która jest stałą i nieuniknioną cechą każdego języka naturalnego. Wieloznaczność wypowiedzi wy-

stępuje na trzech poziomach: leksykalnym (dotyczy słownictwa), syntaktycznym (dotyczy analizy gramatycznej wypowiedzi) oraz semantycznej (dotyczy znaczenia wypowiedzi). Trzeba na wstępie podkreślić, że rozważana wieloznaczność wypowiedzi nie stanowi dla ludzi jakiegoś większego problemu, który utrudniałby wzajemną komunikację. Podobnie wieloznaczność nie jest czymś co utrudniałoby w stopniu znaczącym pracę tłumaczom profesjonalistom [5].

W związku z tym powstaje, niemalże w naturalny sposób, propozycja, aby przekładów dokonywać w ten sposób, aby owa wieloznaczność wypowiedzi została w nich zachowana. Niech zatem odbiorca przełożonych treści sam zada sobie trud rozwikłania ewentualnych wieloznaczności – po cóż więc zaprzątać tym „głowę" komputerowi zaangażowanemu w proces translacji? Niestety, jak się za chwilę okaże, sprawy nie mają się tak prosto. Wręcz przeciwnie, zachowanie wieloznaczności wypowiedzi w przełożonym tekście okazuje się bardzo często w praktyce po prostu niemożliwe. Jakie są zatem po temu powody?

Najbardziej rozpowszechnionym, i sprawiającym zarazem najwięcej kłopotów podczas automatyzacji procesu translacji pomiędzy językami naturalnymi, typem wieloznaczności jest wieloznaczność występująca na poziomie leksykalnym. Istotnie, otwierając jakikolwiek duży dwujęzyczny słownik (np. francusko-polski) trudno jest do prawdy znaleźć wyrazy, które posiadałyby tylko jedno znaczenie. Wręcz przeciwnie, reguła jest taka, że oprócz jednego znaczenia, które jest zwykle uznawane za podstawowe, wyrazy posiadają również cała masę pobocznych znaczeń. Językiem, który szczególnie obfituje w tego rodzaju wieloznaczności słownictwa jest język angielski. Na przykład w Wielkim Słowniku Angielsko-Polskim [32] można znaleźć następujące polskie ekwiwalenty angielskiego słowa *bow*, występującego w zdaniu w funkcji rzeczownika:

bow – 1. dziób (statku), 2. ukłon, 3. łuk, 4. kabłąk, 5. smyczek, 6. kokarda, 7. węzeł, 8. związanie, 9. łęk (siodła), 10. pióro kreślarskie do cyrkla, 11. motylek (krawat).

Jak widać tylko jedno angielskie słowo posiada aż jedenaście polskich odpowiedników. Jest rzeczą oczywistą, że rozważana wieloznaczność leksykalna musi stanowić poważną trudność dla twórców programów translacji komputerowej. Bowiem, gdy komputer w tłumaczonym przez siebie tekście napotyka angielskie słowo *bow*, wówczas musi podjąć decyzję odnośnie tego, którym z polskich odpowiedników to słowo należy zastąpić. Ponieważ wymienione powyżej polskie ekwiwalenty angielskiego słowa *bow* posiadają różne znaczenia, czasem zupełnie nie związane ze sobą (np. kokarda i smyczek), podjęcie podczas dokonywania wyboru błędnej decyzji, będzie z niezwykle dużą dozą prawdopodobieństwa mieć opłakane skutki. Po prostu, odbiorca przetłumaczonego w błędny sposób tekstu najprawdopodobniej w ogóle nie będzie wiedział o co w nim chodzi [6].

Powstaje tutaj oczywiście pytanie, skąd komputer niby ma wiedzieć, który z ekwiwalentów słowa *bow* należy w danej sytuacji wybrać? Człowiek tłumacz w takim wypadku kieruje się informacją wydobytą z kontekstu wypowiedzi, a pomaga mu w tym jego inteligencja, przenikliwość umysłu, inwencja twórcza, ogólna wiedza o świecie i najzwyczajniejszy ludzki zdrowy rozsądek. Tak, ale w jaki sposób można wbudować

w komputer taką ogólną wiedzę o świecie i zdrowy rozsądek? Zautomatyzowanie procesu wnioskowania, pozwalającego na wydobycie informacji z kontekstu wypowiedzi też może okazać się w praktyce niezwykle trudne [7].

Jeżeli natomiast założymy, że komputer będzie dobierał polskie odpowiedniki angielskiego słowa *bow* metodą na chybił trafił, to trzeba zauważyć, że prawdopodobieństwo popełnienia błędu wynosi aż $\frac{10}{11}$. Przy założeniu optymistycznego wariantu, że każde słowo posiada średnio tylko dwa różne znaczenia, prawdopodobieństwo to wynosi $\frac{1}{2}$. Jeżeli założymy ponadto, że typowe zdanie zbudowane jest z dziesięciu wyrazów, to prawdopodobieństwo uzyskania jego prawidłowego przekładu, przy rozwikływaniu wieloznaczności leksykalnych metodą czysto losową, wynosi zaledwie $\frac{1}{1024}$. Jeżeli tłumaczony za pomocą komputera typowy tekst składa się ze stu takich zdań, to prawdopodobieństwo trafnego doboru znaczeń wszystkich występujących w nim wyrazów wynosi zaledwie 1,0715 pomnożone przez dziesięć podniesione do potęgi minus 301. W porównaniu z uzyskaną tutaj wartością prawdopodobieństwa, prawdopodobieństwo trafienia szóstki na pojedynczym kuponie totolotka jest niewyobrażalnie duże, bo wynosi aż $\frac{1}{13983816}$ i jest, w związku z tym, o 293 rzędów wielkości większe!

Po przeprowadzonych powyżej wyliczeniach nietrudno odgadnąć, że przekład uzyskany za pomocą takiego programu komputerowego będzie nadawał się tylko i wyłącznie do kosza na śmieci (jest to zapewne miejsce odpowiednie również dla samego rozważanego tutaj programu translacji automatycznej). Niestety, zdecydowana większość programów automatycznej translacji, dotyczy to zwłaszcza prób stworzenia takich programów dla języka polskiego (przekład na język angielski bądź niemiecki i kierunku przeciwnym), działa na zasadach podobnych do wyżej opisanych – nic zatem dziwnego, że produkty te nie cieszą się na rynku zbytnią popularnością.

Innym rodzajem wieloznaczności, który sprawia programom komputerowym znaczne trudności jest wieloznaczność syntaktyczna. Jej źródłem jest fakt, że często analizę gramatyczną zdania można przeprowadzić na kilka różnych, ale całkowicie równoprawnych sposobów. Na przykład, w zależności od tego, angielskie zdanie:

I see a boy with a telescope.

można przełożyć jako:

Widzę chłopca z teleskopem.

albo

Widzę chłopca za pomocą teleskopu.

Jeżeli osoba dokonująca przekładu lub wyręczający ją w tym program komputerowy, nie wie dokładnie o co chodzi (kto ma właściwie ten teleskop, osoba obserwowana

czy obserwator?), to również nie będzie w stanie dokonać prawidłowego wyboru spośród przedstawionych powyżej dwóch możliwości. W praktyce człowiek tłumacz posługując się kontekstem wypowiedzi potrafi zwykle rozstrzygnąć tego typu dylematy, jednakże zautomatyzowanie takiego zdroworozsądkowego wnioskowania graniczy w praktyce z cudem.

Ostatnim rodzajem wieloznaczności, z którą borykać się muszą programy automatycznej translacji, jest wieloznaczność występująca na poziomie semantyki wypowiedzi. Na przykład angielskie zdanie:

Put some toner into the printer and then switch it on.

teoretycznie można przetłumaczyć na dwa równoprawne sposoby:

Dodaj toner do drukarki a następnie uruchom go.

albo

Dodaj toner do drukarki a następnie uruchom ją.

Oczywiście, w praktyce tylko drugi przekład uznany może zostać za poprawny. Trzeba po prostu wiedzieć, że w rozważanym przypadku, angielski zaimek *it* odnosi się do słowa *printer*, a nie do słowa *toner*. Człowiek tłumacz posługuje się w tym miejscu posiadaną przez siebie wiedzą dotyczącą danej dziedziny działalności ludzkiej, z której wynika, że uruchomić można na przykład drukarkę, ale nie można zrobić tego samego z tonerem. Niestety wiedza taka z niezwykle dużymi oporami poddaje się procesowi algorytmizacji i wbudowaniu w bezrozumny komputer.

Innym rodzajem trudności, występującym nieuchronnie podczas próby zautomatyzowania translacji, są różnice systemowe zachodzące pomiędzy poszczególnymi językami. Panuje w tym względzie żelazna reguła, w myśl której im mniejsze pokrewieństwo (genetyczne, typologiczne i geograficzne) pomiędzy danymi językami, tym rozważane różnice są większe.

Różnice systemowe występują zwłaszcza w systemach gramatycznych różnych języków. Prawdopodobnie (bo autor nie jest tego w stu procentach pewny), w każdym języku świata występują trzy osoby gramatyczne: pierwsza odpowiadająca osobie wypowiadającej się, druga osobie, do której wypowiedź jest kierowana (jest to tzw. interlokutor) oraz trzecia dla osobnika, rzeczy, zjawiska, pojęcia bądź stanu, będącego przedmiotem komunikatu językowego. Na tym jednak zwykle podobieństwa się kończą, dalej występują już tylko prawie same różnice, a im języki są bardziej od siebie odległe, tym różnice te są większe i jest ich zarazem więcej.

Na przykład w języku węgierskim, należącym do ugrofińskiej grupy językowej, nie jest w ogóle znane pojęcie rodzaju gramatycznego. Z tego powodu węgierski zaimek osobowy *ő* można przetłumaczyć na polski, w zależności od kontekstu wypowiedzi, jako: *on*, *ona* lub *ono*. Nie trzeba chyba dodawać, że wyekstrahowanie przez komputer potrzebnych do tego informacji z kontekstu wypowiedzi (trzeba wiedzieć, czy mowa jest o mężczyźnie, kobiecie, czy też o dziecku) wcale nie jest rzeczą łatwą – w przeciwieństwie do człowieka, który na ogół nie ma z tym jakichś szczególnych trudności.

Z kolei polskiemu zaimkowi osobowemu *ty* odpowiada arabski zaimek *'anta*, jeżeli wypowiedź kierowana jest do mężczyzny (co można by wyrazić po polsku, jako „ty mężczyzno") , lub *'anti*, jeżeli wypowiedź jest kierowana do kobiety (można by powiedzieć po polsku „ty kobieto"). Jak widać, aby dokonać prawidłowego przekładu, komputerowy tłumacz musi wiedzieć, czy osoba mówiąca zwraca się do kobiety, czy też do mężczyzny.

Kolejna osobliwość dotycząca języka arabskiego polega na występowaniu w tym języku tzw. liczby podwójnej, odnoszącej się do przypadku, w którym jest mowa o dokładnie dwóch osobnikach, rzeczach, pojęciach bądź zjawiskach. Jako ciekawostkę można podać, że liczba podwójna występowała w średniowieczu również w języku polskim i jej formy wyszły z użycia dopiero w XV wieku.

Zatem tłumacząc na język arabski polski zaimek osobowy *wy* trzeba wiedzieć, jaka jest liczba osób, do których mówca się zwraca. Bowiem, jeżeli są to tylko dwie osoby należy w przekładzie użyć zaimka *'antuma* (można by powiedzieć po polsku „wy dwaj" albo „wy dwie"). Jeżeli natomiast osób tych jest więcej, to trzeba jeszcze znać ich płeć, bowiem jeśli są to sami mężczyźni bądź grupa mieszana (damsko-męska), należy posłużyć się zaimkiem *'antum*, natomiast w przypadku, gdy są to same kobiety trzeba użyć zaimka *'antunna*.

Spośród żywych języków słowiańskich liczba podwójna występuje obecnie jeszcze tylko w języku słoweńskim. Na przykład, chcąc przetłumaczyć na język słoweński polski wyraz *bracia* trzeba wiedzieć, czy jest ich dwóch – należy wówczas użyć formy *brati* (można przełożyć na polski jako „dwaj bracia"), czy też jest ich więcej – w tym przypadku odpowiednią formą będzie *brata*.

Kolejną osobliwością zaczerpniętą, tym razem, z języka hiszpańskiego jest trójstopniowy system zaimków wskazujących, podczas gdy w większości języków europejskich występuje system dwustopniowy. Na przykład w języku polskim mamy dwie formy zaimków wskazujących: *to* i *tamto*, w zależności od stopnia bliskości osoby mówiącej i przedmiotu, o którym jest mowa. Natomiast w języku hiszpańskim występują następujące formy zaimków wskazujących: *esto* (można przełożyć na polski jako „to tutaj"), *eso* (można przełożyć na polski jako „to tam") i *aquello*.

Duże różnice systemowe zaobserwować można analizując systemy czasów gramatycznych wybranych języków. Na przykład, system czasów gramatycznych języka polskiego jest o wiele uboższy do systemu języka angielskiego. Dla przykładu rozważmy angielskie zdanie warunkowe:

If I had enough money I would buy this car.

które można przełożyć na język polski jako:

Gdybym miał wystarczającą sumę pieniędzy, kupiłbym ten samochód.

Należy jednak zauważyć, że identyczne tłumaczenie na język polski ma również następujące angielskie zdanie warunkowe:

If I had had enough money I would have bought this car.

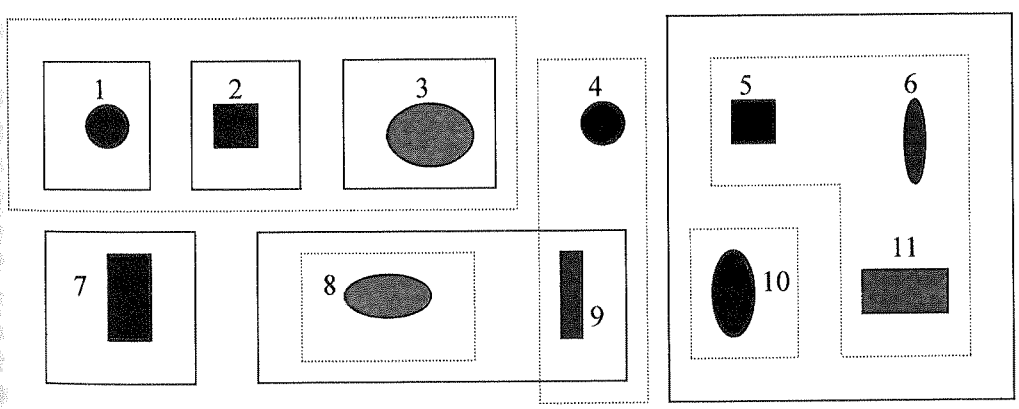
Natomiast znaczenia obu tych zdań jest zupełnie odmienne, bowiem w pierwszym przypadku rozważany warunek jest wciąż możliwy do spełnienia, podczas gdy drugie z wymienionych zdań odnosi się do przeszłości, w której warunek ten nie został spełniony. Należy zauważyć, że język polski nie posiada konstrukcji gramatycznych, pozwalających na oddanie wymienionych różnic znaczeniowych. Aby to uczynić, należałoby w języku polskim posłużyć się czasem zaprzeszłym. Problem polega jednakże na tym, że gramatyka współczesnego języka polskiego użycia takiego czasu nie dopuszcza, wyszedł on bowiem z użycia na przełomie XIX i XX wieku – posługiwali się nim jeszcze dość nagminnie w swoich utworach literackich, między innymi, Henryk Sienkiewicz i Eliza Orzeszkowa.

Zatem posługując się (niedopuszczalną oficjalnie) konstrukcją czasu zaprzeszłego, rozważane angielskie zdanie należało by przełożyć jako:

Gdybym był miał wystarczającą sumę pieniędzy, kupiłbym był ten samochód.

Warto zwrócić uwagę, że takie konstrukcje można jeszcze spotkać w polskim języku mówionym – zwłaszcza w przypadku starszych osób.

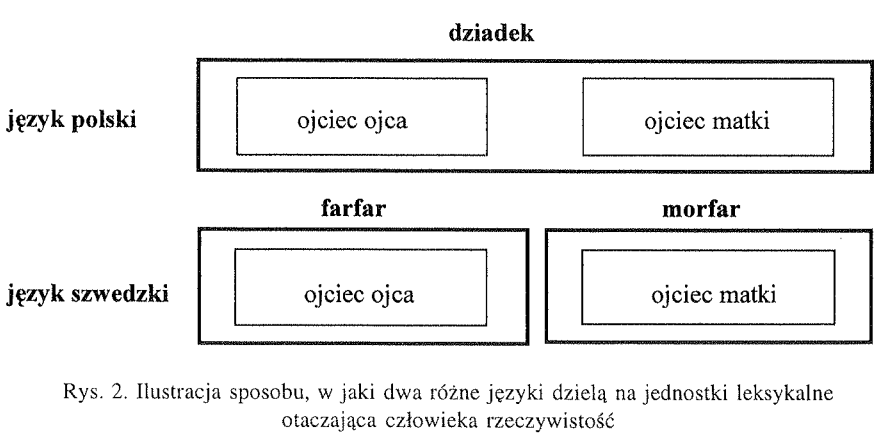
Różnice systemowe pomiędzy różnymi językami występują również w słownictwie, bowiem słownictwo każdego języka stanowi niezwykle skomplikowany system, o nie poznanych jeszcze do końca wzajemnych relacjach między poszczególnymi jednostkami leksykalnymi. Każdy z języków dzieli opisywaną przez niego rzeczywistość (świat osób, przedmiotów, zjawisk, pojęć itp.) we właściwy tylko dla niego sposób. Jest rzeczą zastanawiającą, że użytkownicy różnych języków postawieni w obliczu dokładnie takiej samej sytuacji, w formułowanych na jej temat wypowiedziach umieszczają zupełnie odmienne elementy tej sytuacji dotyczące. Bowiem każdy z języków dzieli otaczającą jego użytkowników rzeczywistość na odmienne fragmenty, którym następnie przypisywane są odrębne jednostki leksykalne. Proces ten został zilustrowany na rysunku 1.



Rys. 1. Podział rzeczywistości na jednostki leksykalne w dwóch różnych językach

Na rysunku 1 przedstawiono w sposób poglądowy jedenaście różnych obiektów, którymi mogą być zarówno przedmioty ze świata rzeczywistego, jak również pojęcia abstrakcyjne. Poszczególne języki łączą te przedmioty w klasy, którym przypisują jednostki leksykalne, jednakże robią to na zupełnie odmienne sposoby. Dla ustalenia uwagi, podział na jednostki leksykalne w pewnym języku A został zaznaczony linią ciągłą, a w języku B linią przerywaną. Na rysunku 1 widać, że język A przypisuje każdemu z obiektów 1,2 i 3 odmienne jednostki leksykalne, podczas gdy język B obiekty te określa tylko jedną, wspólną jednostką leksykalną. Można powiedzieć, że w tym przypadku język A odznacza się większą szczegółowością opisu rzeczywistości niż język B. Nie jest to jednakże jakąś bezwzględną regułą, bowiem język A określa wspólną jednostką leksykalną obiekty 5, 6, 10 i 11, podczas gdy język B rozróżnia tutaj dwie odrębne jednostki leksykalne. W języku B obiekty 4 i 9 określane są wspólną jednostką leksykalną, podczas gdy w języku A obiektowi 4 w ogóle nie jest przypisana żadna jednostka leksykalna, a obiekt 9 wraz z obiektem 8 stanowi jedną jednostkę leksykalną. Należy zauważyć, że w języku B obiekt 8 to całkowicie odrębna jednostka leksykalna. Również obiekt 7 stanowi w języku A odrębną jednostkę leksykalną, podczas gdy w języku B obiekt ten nie został zaliczony do żadnej klasy.

Powyższe, nieco abstrakcyjnie brzmiące, wywody zostaną teraz zilustrowane przykładami zaczerpniętymi z konkretnych języków świata. Na przykład, chcąc przetłumaczyć na język szwedzki polski wyraz *dziadek* można użyć słowa *morfar*, jeżeli dziadek ten jest ojcem matki, lub *farfar*, jeżeli rozważany dziadek jest ojcem ojca, co pokazano na rysunku 2. Jak widać bez dodatkowej informacji wydobytej, być może, z kontekstu wypowiedzi, dokonanie poprawnego przekładu w ogóle nie jest możliwe. Nie trzeba chyba dodawać, że stworzenie programu komputerowego, który przeprowadzałby w sposób automatyczny tego typu wnioskowanie może okazać się niezmiernie trudne do realizacji.



Rys. 2. Ilustracja sposobu, w jaki dwa różne języki dzielą na jednostki leksykalne otaczającą człowieka rzeczywistość

Inny ciekawy przykład pochodzi z języka francuskiego. Otóż, gdy chcemy przetłumaczyć na język francuski polskie słowo *rzeka*, mamy do wyboru wyraz *fleuve*, który określa rzekę uchodząca bezpośrednio do morza (rzeką taką jest np. Rodan) lub *rivière*,

który określa rzekę nie posiadającą bezpośredniego połączenia z morzem. Zatem program automatycznej translacji dokonujący przekładu z języka polskiego na francuski, musiałby posiadać wbudowaną weń bazę wiedzy geograficznej, aby wiedzieć w danym przypadku, jakiego francuskiego ekwiwalentu polskiego słowa *rzeka* użyć.

Również chcąc przełożyć na język angielski polski wyraz *skóra*, trzeba wiedzieć, czy *skóra* ta jest już w jakiś sposób przetworzona (wygarbowana, zabarwiona itp.), wówczas należy użyć słowa *leather*, lub czy jest to *skóra* żywa bądź przetworzona w taki sposób, że przypomina żywą *skórę* (np. skórę węża na torebce z wężowej skóry), wówczas należy użyć słowa *skin*. Nie trzeba chyba dodawać, że wyciąganie tego typu wniosków przez program komputerowy jest mało prawdopodobne.

Podobnie w języku niemieckim, polskiemu czasownikowi *jeść* odpowiada czasownik *essen*, gdy mowa jest o jedzącym człowieku lub *fressen*, gdy mowa jest o jedzącym zwierzęciu. Zatem komputer musiałby jakoś umieć wywnioskować z kontekstu wypowiedzi, czy mowa jest o człowieku, czy też o zwierzęciu – tak, ale w jaki sposób zautomatyzować takie wnioskowanie?

Z jeszcze dziwniejszymi przypadkami mamy do czynienia, gdy rozważymy dwa języki bardzo od siebie odległe pod względem genetycznym i geograficznym. Na przykład, chcąc przetłumaczyć na język japoński polskie słowo *siostra*, należy użyć słowa *onesan*, gdy siostra ta jest młodsza lub *imoto*, gdy owa siostra jest starsza. Podobnie, w indiańskim języku navaho nie ma odpowiednika polskiego czasownika *iść*, istnieje natomiast cały szereg czasowników, które określają odpowiednio poruszanie się pojedynczej osoby, dwóch osób, małej grupki osób, dużej grupy osób itp. Również walijski przymiotnik *glas* odpowiada polskim przymiotnikom *zielony*, *niebieski* i *szary* – zatem użytkownicy języka walijskiego nie rozróżniają w mowie w ogóle tych kolorów, ponieważ określają je wszystkie wspólną nazwą. W języku polskim możemy zwrócić się do drugiej osoby albo używając formy *ty*, albo też formy *pan* względnie *pani*. Natomiast w języku koreańskim istnieje aż sześć różnych stopni grzeczności, z których każdy wymaga użycia odmiennych form czasownikowych. Należy zdać sobie sprawę z faktu, że problemu tego nie można w żaden sposób ominąć, bo w końcu na którąś z alternatywnych form trzeba się przecież zdecydować.

Ponadto pewne jednostki leksykalne zaczerpnięte z jednego języka mogą nie mieć swoich odpowiedników w innym języku. Dotyczy to przede wszystkim pojęć abstrakcyjnych, będących wytworem pewnej specyficznej kultury (np. pojęcia filozofii hinduskiej), jak również nazw obiektów rzeczywistych związanych z pewnymi realiami lokalnymi. Przykładowo w języku serbskim słowo *badnjak* oznacza kłodę dębową spalaną według starych serbskich zwyczajów w wigilię święta Bożego Narodzenia. Podobnie w języku arabskim słowo *halhalun* określa rodzaj specjalnej bransoletki, którą muzułmańskie kobiety zakładają na nogę. Podobnie arabskie słowo *badrun* oznacza księżyc, który akurat jest w pełni, a słowo *thukla* kobietę, która utraciła jedyne swe dziecko. Takie jednostki leksykalne nie posiadają swoich bezpośrednich odpowiedników w języku polskim i, w związku z tym, mogą zostać przetłumaczone na język polski jedynie w sposób opisowy, co w niektórych przypadkach może sprawiać nieco trudności.

Odrębnym problemem występującym podczas translacji jest tłumaczenie różnego rodzaju skostniałych związków frazeologicznych, wyrażeń idiomatycznych i przysłów. Przykładem takich skostniałych związków frazeologicznych mogą być na przykład połączenia rzeczowników ze specyficznymi przymiotnikami. Zjawisko to powoduje, że dosłowne tłumaczenie takiego przymiotnika brzmi nienaturalnie w połączeniu z towarzyszącym mu rzeczownikiem. Na przykład dosłowne tłumaczenie włoskiego zdania:

Lui è un uomo bello.

jako:

On jest pięknym mężczyzną.

brzmi w języku polskim sztucznie. Należałoby raczej powiedzieć:

On jest przystojnym mężczyzną.

Jednakże włoski przymiotnik *bello* znaczy *piękny*, ewentualnie *ładny*, natomiast w rozważanym przypadku należy mu przypisać jeszcze jedno dodatkowe znaczenie: *przystojny*.

Z kolei gdyby przetłumaczyć dosłownie szwedzkie porzekadło:

Många bäckar små gör en stor a.

na język polski, otrzymano by zdanie:

Wiele małych strumyków czyni rzekę.

Należy oczekiwać, że tłumacz profesjonalista posłużyłby się zapewne polskim analogiem tego szwedzkiego porzekadła i przełożył je jako:

Kropla drąży skałę.

Podobnie niemieckie przysłowie:

Schlafende Hunde soll man nicht wecken.

można by dosłownie przełożyć na język polski, jako:

Nie powinno się budzić śpiących psów.

Natomiast właściwym przekładem będzie polski odpowiednik tego przysłowia:

Nie wywołuj wilka z lasu.

Aby komputer mógł prawidłowo tłumaczyć tego typu skostniałe związki frazeologiczne, porzekadła i przysłowia należałoby wyposażyć go w odpowiednią bazę danych, w której zawarte byłyby wzorce takich tłumaczeń. W zasadzie można to zrobić bez jakichś większych problemów, natomiast pewien niepokój budzić mogą rozmiary takiej bazy danych.

O wiele bardziej skomplikowanie przedstawiają się sprawy w przypadku próby automatyzacji przekładu wyrażeń idiomatycznych, czyli takich związków składniowych,

których znaczenia nie jesteśmy w stanie wywnioskować na podstawie analizy znaczeń ich elementów składniowych. Na przykład amerykańskie zdanie zawierające konstrukcję idiomatyczną:

He had ants in his pants.

posiada następujące znaczenie:

Wiercił się i nie mógł spokojnie usiedzieć na miejscu.

Jedyny problem z przekładem tego zdania polega na tym, że zdanie takie może zostać przetłumaczone również dosłownie, jako:

Miał mrówki w spodniach.

co też może być przekładem poprawnym w przypadku, gdy opisywana jest pewna rzeczywista sytuacja, bo przecież ktoś mógł wybrać się na przykład na wycieczkę do lasu i jakieś mrówki mogły mu istotnie powłazić do spodni (co nie było z pewnością dla tej osoby zbyt przyjemne).

Podobnie, angielskie zdanie:

Sam kicked the bucket.

znaczy w tłumaczeniu dosłownym, że:

Sam kopnął wiadro.

Natomiast w przenośni (jako dość popularny w Stanach Zjednoczonych idiom) zdanie to znaczy, że: *Sam umarł* – można by posłużyć się tutaj polskim odpowiednikiem tego idiomu i powiedzieć, że: *Sam kopnął w kalendarz*. Problem polega jedynie na tym, że w pewnych sytuacjach właśnie tłumaczenie dosłowne może być tłumaczeniem właściwym, bo przecież nasz Sam mógł rzeczywiście kopnąć w zdenerwowaniu jakieś stare wiadro, które gdzieś raczyło stanąć mu na drodze.

Oczywiście, człowiek tłumacz prawie zawsze jest w stanie wywnioskować z kontekstu wypowiedzi, czy daną frazę należy traktować dosłownie, czy też, jako wyrażenie idiomatyczne. Natomiast opracowanie algorytmu, który byłby w stanie przeprowadzić tego typu wnioskowanie pozostawia się czytelnikowi, jako ćwiczenie, które powinien wykonać, gdy tylko znajdzie chwilę wolnego czasu...

Powyższe przykłady, zaczerpnięte z różnych języków świata, uświadamiają nam, jak niezwykle złożonym i trudnym jest proces translacji. Bowiem, czytając jakąkolwiek książkę przełożoną z języka obcego na język polski, zwykle nie zdajemy sobie nawet sprawy, jak wiele trudu musiał włożyć tłumacz, aby przedstawić czytelnikowi opisywany w tej książce świat, w sposób jak najbardziej zbliżony do sposobu, w jaki opisał go w oryginale autor.

Biorąc pod uwagę rozważane trudności, nasuwa się nieuchronnie pytanie, czy w związku z tym badania nad translacją automatyczną mają jeszcze w ogóle jakikolwiek sens? Czy możliwe jest zalgorytmizowanie i zautomatyzowanie tego procesu? A jeżeli tak, to jakie należy stosować do tego metody?

5. NAJBARDZIEJ POPULARNE ROZWIĄZANIA W SYSTEMACH TRANSLACJI AUTOMATYCZNEJ

W wielu ośrodkach badawczych na całym świecie opracowano do tej pory całą masę różnorodnych podejść do zagadnienia translacji automatycznej. Wymienić w tym miejscu można podejścia oparte na tzw. transferze struktur morfologicznych, bazie reguł translacyjnych, bazie wiedzy o języku, koncepcji abstrakcyjnego języka – interlingua oraz na metodach statystycznych, takich jak np. niejawne modele Markowa i sztuczne sieci neuronowe.

Obecnie najbardziej obiecującym kierunkiem badań wydaje się być tłumaczenie oparte na przykładach. Technika ta oznaczana jest skrótem EBMT (ang. Example-Based Machine Translation). Idea leżąca u podstaw tego podejścia jest bardzo prosta. Załóżmy, że dysponujemy pewnym tekstem napisanym, dla ustalenia uwagi, w języku polskim oraz innym tekstem, który stanowi jego przekład na np. język hiszpański. Ponadto załóżmy, że każdemu zdaniu z tekstu polskiego odpowiada dokładnie jedno zdanie w tekście hiszpańskim. Zatem zdania z obu tekstów możemy uporządkować w dwujęzyczne pary, a następnie umieścić je w pewnej bazie danych. Po zgromadzeniu takiej bazy danych o odpowiednio dużych rozmiarach można już przystąpić do tłumaczenia techniką EBMT.

Jeżeli tłumaczymy teraz jakiś zupełnie nowy tekst napisany w języku polskim i w tekście tym wystąpi pewne zdanie, którego tłumaczenie zostało już wcześniej umieszczone w bazie danych, to istnieje bardzo duża szansa, że umieszczając tłumaczenie tego zdania zaczerpnięte z bazy danych w tworzonym aktualnie w języku hiszpańskim tekście, dokonamy prawidłowego przekładu tego zdania. Ktoś może od razu zapytać, dlaczego nie ma całkowitej pewności, że taki przekład jest całkowicie poprawny? Czyżby do rozważanej bazy danych zakradły się jakieś błędy?

Otóż nie, wręcz przeciwnie, jedynym powodem pojawienia się nieprawidłowego przekładu nie są błędy, od których zakładamy, że nasza baza danych jest wolna, ale różnice systemowe pomiędzy dwoma językami. Jeżeli przypomnimy sobie teraz wcześniejsze rozważania na temat możliwości i ograniczeń dokonywania przekładów pomiędzy różnymi językami naturalnymi, sprawa stanie się jasna. Dodatkowo zagadnienie to zostanie zilustrowane jeszcze kilkoma dalszymi przykładami. Otóż, chcąc przetłumaczyć z języka węgierskiego na polski zdanie:

Ő orvos.

do wyboru są dwie możliwości:

On jest lekarzem.

albo

Ona jest lekarką.

Jest to bezpośrednią konsekwencją faktu, że język węgierski nie zna pojęcia rodzaju gramatycznego. Ponieważ w bazie danych zdania polskie i węgierskie ułożone są

parami, do bazy takiej można wpisać tylko jeden z dwóch podanych powyżej wariantów polskiego przekładu. Jeżeli założymy, że w bazie tej wpisano zdanie: *Ona jest lekarką*, a w tłumaczonym tekście chodzi akurat o mężczyznę, to oczywiście przekład taki będzie błędny.

Płynie stąd ważna wskazówka, w tekstach, które mają zostać poddane automatycznej translacji, należy wystrzegać się stosowania jakichkolwiek enigmatycznych wyrażeń. W szczególności należy unikać stosowania zaimków, a w ich miejsce należy używać pełnych nazw. Na przykład, gdyby tłumaczone węgierskie zdanie brzmiało:

István orvos.

miałoby ono dokładnie jedno tłumaczenie na język polski:

Stefan jest lekarzem.

Jednak nie zawsze można się w ten sposób ustrzec przed wszelkimi pułapkami językowymi. Przykład takiej sytuacji pochodzi z języka szwedzkiego. Załóżmy, że chcemy przetłumaczyć na język szwedzki następujące polskie zdanie:

Babcia pytała kiedy przyjedziemy.

Mimo, iż nie ma w tym zdaniu żadnych zaimków, istnieją dwie alternatywne wersje jego przekładu:

Mormor fragade när vi skulle komma.

albo

Farmor fragade när vi skulle komma.

Pierwszy przekład odnosi się do przypadku, w którym jest mowa o babci, będącej matką matki, natomiast w drugim mowa jest o babci będącej matką ojca. Ponieważ język polski nie wytworzył odrębnych kategorii leksykalnych, pozwalających na odróżnienie takich koligacji rodzinnych, w sytuacji, gdy wzięty zostanie z bazy danych przekład: *Mormor fragade när vi skulle komma*, a w tekście będzie chodziło o babcię, która jest matką ojca, to oczywiście przekład taki będzie nieprawidłowy, a ponadto u szwedzkiego czytelnika wywoła on zapewne konsternację, bo czytelnik ten nie będzie wiedział, o którą babcię w końcu chodzi.

Podobnie, gdy chcemy przetłumaczyć na język hiszpański polskie zdanie:

Rozmawialiśmy o was.

również istnieją dwie alternatywne możliwości przekładu:

Hemos hablado de vosotros.

oraz

Hemos hablado de vosotras.

Pierwszy przypadek dotyczy sytuacji gdy rozmawiano o samych mężczyznach lub o grupie mieszanej, w której byli zarówno mężczyźni, jak i kobiety. Natomiast przypadek drugi dotyczy sytuacji, w której rozmawiano o osobach będących kobietami.

Zatem gdyby w bazie danych znajdował się przekład: *Hemos hablado de vosotras*, a w rzeczywistości chodziło o mężczyzn, przekład taki byłby po prostu dla czytających go Hiszpanów zabawny.

Językiem, który potencjalnie może wyjątkowo obfitować w tego typu pułapki, jest język niderlandzki. Sytuacja taka spowodowana jest faktem, że w języku tym nie istnieją żadne specjalne formy służące do wyrażania trybu warunkowego zdania. Zatem w sytuacji wyrwania zdań z kontekstu trudno jest ustalić jednoznacznie ich znaczenie. Zostanie to pokazane na kilku następujących przykładach.

Między innymi, niderlandzkie zdanie:

Als ik in Amsterdam ben, ga ik naar het Rijksmuseum.

może zostać przetłumaczone, jako:

Jak jestem w Amsterdamie, to idę do Muzeum Państwowego.

albo:

Gdy będę w Amsterdamie, to pójdę do Muzeum Państwowego.

Również zdanie:

Als ik in Amsterdam was, ging ik naar het Rijksmuseum.

można przełożyć jako:

Jak byłem w Amsterdamie, to poszedłem do Muzeum Państwowego.

albo

Gdybym był w Amsterdamie, to poszedłbym do Muzeum Państwowego.

Zadania takie mogą, w przypadku zastosowania rozważanej metody automatycznej translacji opartej na przykładach, przysporzyć sporo trudności.

Rzeczą, która może zastanawiać czytelnika jest zapewne rozmiar bazy danych, która byłaby w stanie pomieścić wszystkie przykłady tłumaczeń zdań pochodzących z danego języka. Ponieważ liczba możliwych do utworzenia zdań w dowolnie wybranym języku jest zapewne astronomiczna, nikt jeszcze takiej bazy danych nie opracował. W praktyce stosuje się sklejanie potrzebnych zdań z dostępnych z bazy danych fragmentów. Rozwiązanie takie rodzi jednakże liczne problemy, które mają swe źródło głównie na styku takich fragmentów – ich wzajemne dopasowanie nie jest wcale sprawą prostą.

Jednakże trzeba zauważyć, że już obecnie metoda translacji automatycznej oparta na przykładach EBMT cieszy się coraz większą popularnością. Prace w tym kierunku prowadzone są, między innymi, na prestiżowej amerykańskiej uczelni – *Carnegie Mellon University* w Pittsburgu w stanie Pensylwania. Na uczelni tej opracowano, w oparciu o technikę EBMT, dwa programy komputerowe, służące do dokonywania automatycznej translacji z języka angielskiego na język francuski i hiszpański, dla tekstów o tematyce ogólnej. Uzyskane wyniki są zachęcające, ponieważ, pomimo

względnie niewielkich rozmiarów bazy danych, w której mieszczą się przykłady translacyjne – zgromadzono około trzech milionów takich przykładów o średniej długości nieco ponad trzech słów – uzyskano pokrycie tłumaczonych dokumentów tymi przykładami rzędu 97%. Ponadto badacze z *Carnegie Mellon University* zaproponowali oryginalne podejście polegające na uogólnianiu przykładów translacyjnych, poprzez zastępowanie w nich wybranych słów tzw. żetonami (ang. tokens). Taki uogólniony przykład zawierający pewną liczbę żetonów jest w stanie pomieścić w sobie wiele indywidualnych przykładów, podpadających pod daną kategorię, co posiada kolosalny wpływ na rozmiary potrzebnej do translacji bazy danych.

Spektakularny sukces, związany z zastosowaniem techniki translacji automatycznej opartej na przykładach EBMT odniesiono również w Hiszpanii, gdzie opracowany został system służący do dokonywania automatycznych przekładów z języka hiszpańskiego na język kataloński, który jest językiem ojczystym dla około 10 milionów osób – Katalończyków – zamieszkujących południową Hiszpanię (między innymi rejon Barcelony i wyspy Baleary) oraz część Sardynii (okolice miasta Alghero). Rozważany system został zastosowany do automatycznego tłumaczenia, z urzędowego języka hiszpańskiego na język kataloński, ukazującego się w Barcelonie czasopisma regionalnego zatytułowanego *Periódico de Catalunya*. Co najciekawsze, uzyskany tą drogą przekład jest na tyle dobrej jakości, że nie są już później wykonywane żadne dalsze poprawki, a czasopismo kierowane jest bezpośrednio do druku. Nakład tego czasopisma wynosi ponad 60 tysięcy egzemplarzy oraz ukazuje się ono codziennie, a tłumaczone jest automatycznie począwszy do 1999 roku. Jest to chyba pierwszy i jedyny przykład zastosowania translacji automatycznej, do języka o tematyce ogólnej, na tak dużą skalę i z tak dobrym skutkiem. Równie intrygujący jest fakt, że w rozważanym systemie translacji automatycznej nie wykorzystano żadnych wyników badań prowadzonych w dziedzinie tzw. lingwistyki komputerowej. Wręcz przeciwnie, system ten analizuje podawane na jego wejście zdania w możliwie najprostszy sposób, podobny do pracy programu sprawdzającego pisownię. Po prostu słowa hiszpańskie, bądź zlepki takich słów, czyli tzw. związki frazeologiczne, o maksymalnej długości sześciu wyrazów, są zastępowane ich katalońskimi odpowiednikami. Rozważany system translacji automatycznej dysponuje jedynie słownikiem o bardzo dużych rozmiarach, w którym mieszczą się wszelkie potrzebne wzorce tłumaczeń poszczególnych wyrazów i całych związków frazeologicznych. Uzyskane za pomocą tego systemu teksty odznaczają się wysoką jakością i wysokim stopniem dokładności lingwistycznej przekładu. Również godna uznania jest duża szybkość pracy rozważanego systemu. Mianowicie, jest on w stanie przetłumaczyć dokument o rozmiarach 15KB w ciągu zaledwie dwóch sekund. System ten znajduje się ciągle jeszcze w fazie rozwojowej, ponieważ zaangażowana w jego powstanie grupa lingwistów pracuje nieustannie nad uaktualnianiem jego słownika, poprzez dopisywanie do niego nowych słów i nowych form czasowników oraz sekwencji słów – związków frazeologicznych – o długości nie przekraczającej jednakże sześciu wyrazów. Podsumowując można powiedzieć, że jest to typowy przykład praktycznej implementacji systemu translacji automatycznej, bazującego jedynie na dopasowywaniu przykładów translacyjnych i nie korzystającego z żadnych wyrafinowanych technik translacyjnych.

Gdzie zatem leży tajemnica odniesionego sukcesu? Jak to możliwe, że tak trudne zadanie jak translacja automatyczna, można było rozwiązać w tak prosty sposób?

Otóż, tajemnica odniesionego sukcesu leży w bardzo bliskim pokrewieństwie języka hiszpańskiego i katalońskiego. Są to bowiem języki romańskiej grupy językowej, należące ponadto do tej samej podgrupy iberyjskiej tych języków. Różnice pomiędzy tymi dwoma językami występują głównie w ich fonetyce, natomiast różnice we frazeologii i w gramatyce tych języków są znikome.

Wypływa stąd ważny wniosek, odnoście przydatności techniki automatycznej translacji opartej na przykładach EBMT. Mianowicie, należy oczekiwać, że jakość uzyskiwanych tą drogą przekładów będzie bardzo silnie zależała od stopnia pokrewieństwa genetycznego pomiędzy językami zaangażowanymi w przekład. Prawdopodobnie technikę EBMT można by z bardzo dobrym skutkiem zastosować do tłumaczenia tekstów z języka polskiego na języki czeski i słowacki (ewentualnie na języki górno i dolno łużyckie oraz na język kaszubski – oczywiście gdyby w ogóle istniała taka potrzeba), ponieważ wszystkie z wymienionych języków należą do jednej tzw. zachodniej podrodziny języków słowiańskich. Nieco gorszych rezultatów należało by się spodziewać w przypadku tłumaczenia techniką EBMT z języka polskiego na inne języki słowiańskie, należące np. do wschodniej grupy tych języków (rosyjski, ukraiński i białoruski) lub do ich grupy południowej (słoweński, chorwacki, serbski, macedoński i bułgarski) – zwłaszcza język bułgarski i macedoński znacznie odbiegają od innych współczesnych języków słowiańskich. Natomiast mam duże wątpliwości, czy zastosowanie metody EBMT przyniosłoby tak dobre rezultaty w przypadku tłumaczenia z języka polskiego na języki romańskiej i germańskiej grupy językowej. Jeszcze gorszych rezultatów należałoby się spodziewać w przypadku tłumaczenia systemem EBMT z języka polskiego na języki innych grup językowych indoeuropejskiej rodziny językowej, bardziej terytorialnie i kulturowo odległe, takie jak np. języki grupy indoirańskiej czy celtyckiej. Zapewne jeszcze o wiele gorzej miałyby się sprawy w przypadku przekładu na języki należące do odmiennych rodzin językowych, bardzo odległych w sensie klasyfikacji genetycznej i geograficznej, takich jak np. język arabski, suahili, hausa, japoński, chiński, tajski, wietnamski, koreański itp.

6. OCENA MOŻLIWOŚCI ZASTOSOWANIA TRANSLACJI OPARTEJ NA PRZYKŁADACH DLA JĘZYKA POLSKIEGO

W poprzednim punkcie rozważono dwie zupełnie odmienne koncepcje podejścia do zagadnienia automatycznej translacji. Pora obecnie zastanowić się na tym, które z nich jest lepsze, a zwłaszcza które z nich rokuje większe nadzieje na przyszłość.

Podejście oparte na wiedzy lingwistycznej i wiedzy dziedzinowej KBMT, wykorzystujące dodatkowo koncepcję interlingua, można scharakteryzować głównie jako podejście o charakterze intencjonalnym. W istocie, opracowanie systemu KBMT wymaga konstrukcji dużej bazy wiedzy, obejmującej swym zakresem reguły dotyczące zarówno języka źródłowego, jak i języka przekładu, a także reguły pochodzące z wiedzy

dziedzinowej przekładanych tekstów. Zatem podejście KBMT wymaga dużego nakładu pracy osób zaangażowanych w tworzenie rozważanej bazy wiedzy. Ponadto osoby takie muszą posiadać dużą wiedzę lingwistyczną, która umożliwi im wyekstrahowanie reguł analizy tekstów źródłowych i przekształcenie ich do języka interlingua oraz syntezy tekstów w języku docelowym na podstawie zapisu w języku interlingua.

W chwili obecnej opracowanie uniwersalnego, sztucznego języka interlingua, który swym zakresem obejmowałby wybraną grupę języków naturalnych nie wydaje się możliwe. Ponadto rozważana baza wiedzy obejmująca reguły analizy i syntezy do i z języka interlingua musiałaby mieć olbrzymie rozmiary – prawdopodobnie w grę wchodziłoby co najmniej kilka milionów reguł analizy i syntezy. W praktyce zarządzanie taką baza danych, a zwłaszcza jej formalna weryfikacja, wykazanie kompletności tych reguł oraz braku ich wzajemnej sprzeczności, mogłoby okazać się, przynajmniej przy obecnym stanie techniki, niemożliwe.

Wynika stąd automatycznie ograniczony zakres zastosowań podejścia KBMT. Systemy takie nadają się bowiem w praktyce jedynie w przypadku tzw. języków kontrolowanych, które stanowią pewien ograniczony podzbiór języków naturalnych. Ponadto tłumaczone teksty muszą dotyczyć tylko jednej wybranej bardzo wąskiej dziedziny wiedzy i działalności człowieka (np. podręczniki obsługi pewnego typu urządzeń technicznych), dla której możliwe jest opracowanie bazy wiedzy ujmującej tę dziedzinę w pewne sztywne reguły.

Jeżeli chodzi natomiast o tłumaczenie tekstów dotyczących języka ogólnego o dowolnej tematyce niespecjalistycznej (np. język tekstów prasowych), to w chwili obecnej wydaje się, że przyszłość należy do systemów typu EBMT. Istotnie, technika translacji automatycznej oparta na przykładach translacyjnych odniosła już duże sukcesy, jak na przykład w przypadku tłumaczenia z języka hiszpańskiego na język kataloński, a w przyszłości należy oczekiwać dalszego wzrostu wydajności i jakości pracy takich systemów. Również dobre rezultaty uzyskano na Carnegie Mellon University w przypadku tłumaczenia techniką EBMT tekstów w języku francuskim i hiszpańskim na język angielski.

Z kolei, translacja automatyczna typu EBMT scharakteryzowana może zostać jako podejście typu ekstensywnego, bowiem nie jest wymagane posiadanie podczas opracowywania takiego systemu praktycznie żadnej wiedzy lingwistycznej, ponieważ liczy się tylko ilość przykładów translacyjnych umieszczonych w bazie danych – im będzie ich więcej, tym większe prawdopodobieństwo pokrycia nimi tłumaczonego tekstu.

Należy zwrócić uwagę, że wszystkie znane przypadki użycia techniki EBMT, o których wiemy, że zakończyły się dużym sukcesem dotyczyły języków stosunkowo bliskich sobie pod względem klasyfikacji genetycznej, typograficznej i geograficznej. Powstaje w związku z powyższym ważne pytanie, czy uzyskane dotychczas rezultaty w tłumaczeniu typu EBMT mogą zostać równie dobrze przeniesione na grunt języka polskiego, który różni się w sposób istotny od języków używanych w Europie Zachodniej.

Należy oczekiwać, że w miarę wzrostu odległości w klasyfikacji genetycznej pomiędzy językami zaangażowanymi w przekład średnia długość przykładów translacyjnych będzie również wzrastać. Z kolei im dłuższe będą potrzebne przykłady translacyj-

ne, również tym większa będzie musiała być ich ilość. Jeżeli liczba przykładów translacyjnych przekroczy wszelkie rozsądne granice, może ich nie pomieścić żadna baza danych. Spróbujmy zatem oszacować ile takich przykładów translacyjnych powinno być, aby system EBMT działał w miarę sprawnie.

Na początku należy zastanowić się ile może być w rozważanym języku naturalnym różnych form wyrazowych. Trzeba od razu zaznaczyć, że istnieją pod tym względem języki lepsze i gorsze. Z pewnością językami lepszymi będą języki analityczne, w których fleksja wyrazowa została zredukowana do niezbędnego minimum. Natomiast o wiele gorzej wyglądała będzie sprawa w przypadku języków typu syntetycznego, do którego zalicza się wszelkie języki aglutacyjne (np. węgierski, turecki), alternacyjne (np. hebrajski, arabski) oraz fleksyjne (np. język polski i inne języki słowiańskie oprócz bułgarskiego, który jest językiem analitycznym).

Ocenia się, że przeciętnie w języku o tematyce ogólnej (nie dotyczącej jakichś wysoce wyspecjalizowanych dziedzin wiedzy) występuje około różnych wyrazów – tyle mniej więcej haseł zawierają duże słowniki dwujęzyczne. Oczywiście dopuszczalnych form wyrazowych jest więcej, ponieważ każdy z wyrazów ulegając prawidłom odmiany gramatycznej może występować w wielu różnych formach. Na przykład angielski czasownik *to go* może przyjmować następujące formy: *go*, *went*, *gone*, *goes*. W języku polskim dopuszczalnych form czasownika *iść* jest oczywiście więcej: idę, idziesz, idzie, idziemy, idziecie, idą, szedłem, szedłeś, szedł, szła, szło, szliśmy, szłyśmy, szliście, szłyście, szli, szły. W związku z tym liczba dopuszczalnych form wyrazowych w języku polskim będzie o wiele większa niż w języku angielskim – ocenia się ją na około 3 miliony. Niestety są języki jeszcze gorsze pod tym względem niż język polski, na przykład w języku węgierskim nie występują przysłówki, ale zamiast nich są partykuły, które dołącza się do wyrazów. Na przykład słowo *orvos* znaczy lekarz, a *orvoshoz* do lekarza, *orvosnak* lekarzowi, *orvosok* lekarze *orvosoknák* u lekarzy itd. W istocie, system gramatyczny języka węgierskiego powoduje prawdziwą eksplozję różnych dopuszczalnych form wyrazowych.

Dla ustalenia uwagi zostanie przyjęte, że przeciętnie w języku naturalnym występuje 10^6 różnych form wyrazowych. Ponieważ zdecydowana większość języków świata to języki typu analitycznego, stwierdzenie powyższe może być dla większości języków dosyć bliskie prawdy. Zapewne posługując się milionem różnych form wyrazowych, pokryć można ponad 99,9% tekstów o tematyce ogólnej, w przypadku dowolnego języka analitycznego – a to w zupełności wystarcza w praktyce.

Zatem liczba przykładów translacyjnych o długości jednego wyrazu powinna wynosić co najmniej 10^6 – taką liczbę przykładów można spokojnie zmieścić w dowolnej bazie danych. Natomiast w przypadku przykładów translacyjnych o długości dwóch wyrazów ich maksymalna liczba może już wynosić 10^{12}. W rzeczywistości liczba ta będzie zapewne o kilka rzędów wielkości mniejsza, ponieważ nie każde połączenie dwóch różnych form wyrazowych jest dopuszczalne gramatycznie (np. nie można powiedzieć „pies zjadła" tylko „pies zjadł") lub semantycznie (np. pozbawiona sensu jest fraza „kiełbasa zjadła" bo oczywiście kiełbasa niczego zjeść nie może).

W przypadku przykładów translacyjnych o długości trzech wyrazów ich maksymalna ilość wynosi już 10^{18}, natomiast ich rzeczywista ilość będzie o wiele mniejsza, ponieważ zdecydowana większość z takich trójwyrazowych zlepków będzie niepoprawna pod względem gramatycznym, a ponadto pozbawiona jakiegokolwiek sensu. Ile zatem przykładów translacyjnych jest rzeczywiście potrzebnych?

Wątpliwości nie ma jedynie w przypadku przykładów jednowyrazowych – potrzeba ich bowiem milion. A co w innych przypadkach?

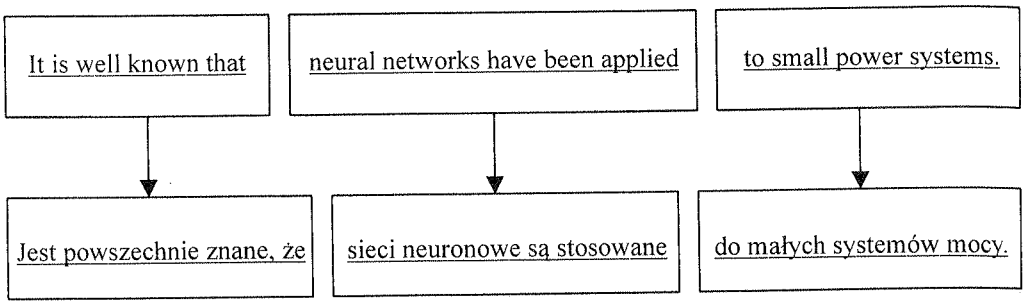
Autor niniejszego artykułu uważa na podstawie przeprowadzonych przez siebie obserwacji, że można zaryzykować w tym względzie pewną hipotezę, w myśl której liczba przykładów translacyjnych dwuwyrazowych będzie o jeden rząd wielkości większa niż jednowyrazowych, a następnie liczba przykładów trójwyrazowych będzie o jeden rząd wielkości większa od liczby przykładów dwuwyrazowych itd. Zatem przykładów dwuwyrazowych byłoby około 10^7, trójwyrazowych około 10^8, czterowyrazowych około 10^9, a pięciowyrazowych około 10^{10}. Zatem całkowita liczba przykładów translacyjnych w systemie EBMT, w którym maksymalną długość przykładów ograniczono by do pięciu słów wynosiłaby około 10^{10}. Zakładając, że każdy z przykładów wymaga zapisu na około 100 bajtach (oprócz podstawowych znaków ASCI potrzebne są zapewne również narodowe modyfikacje podstawowych znaków alfabetu, które zapisywane na więcej niż jednym bajcie danych), rozmiary dysku twardego, który przechowywałby rozważaną bazę danych szacować należy na 10^{12} bajtów. W chwili pisania niniejszego artykułu jeszcze żaden dysk twardy nie osiągnął pojemności 1TB, choć różni producenci są już blisko tej granicy (autorowi znany jest przypadek wyprodukowania dysku o pojemności 170GB). Zatem fizyczna realizacja takiej bazy danych z wszystkimi możliwymi przykładami translacyjnymi o długości ograniczonej do pięciu wyrazów wydaje się już całkiem realna. Problem polega jedynie na tym czy w przypadku języka polskiego pięciowyrazowe przykłady translacyjne okażą się wystarczające.

Aby podjąć próbę udzielenia odpowiedzi na to pytanie, autor niniejszego artykułu postanowił przeprowadzić proste eksperymenty dla kilkunastu wybranych języków, których znajomość posiadł w stopniu dostatecznym do dokonywania przekładów na język polski nieskomplikowanych zbytnio tekstów zapisanych w tych językach.

7. TRANSLACJA OPARTA NA PRZYKŁADACH DLA JĘZYKA POLSKIEGO

Poniżej zamieszczone zostały fragmenty tekstów zapisanych w wybranych językach obcych, które zostały przetłumaczone przez autora metodą opartą na przykładach translacyjnych. Jako przykłady translacyjne zostały wybrane frazy, składające się zwykle z kilku wyrazów. Długość przykładów translacyjnych została wybrana jako minimalna, to znaczy taka, dla której otrzymuje się jeszcze w miarę poprawnie brzmiące przekłady zdań. Rozbicie zamieszczonych poniżej przykładów translacyjnych na większą liczbę krótszych przykładów, daje w efekcie niepoprawnie zbudowane zdania w języku polskim. W tłumaczonych zdaniach w językach obcych i w ich polskich

przekładach kolejne przykłady translacyjne zostały podkreślone, a kolejność ich jest w obu przypadkach taka sama. Zatem łatwo można ustalić, jak przykłady te i ich tłumaczenia należy połączyć w odpowiadające sobie pary. Zagadnienie to zostało zilustrowane na następującym przykładzie translacji z języka angielskiego, przedstawionym na rysunku 3.



Rys. 3. Tłumaczenie angielskiego zdania metodą przykładów translacyjnych

Celem przeprowadzonych przez autora eksperymentów translacyjnych było ustalenie średniej długości (podanej w liczbie wyrazów) przykładów translacyjnych dla różnych języków. W tym celu po każdym zdaniu w języku źródłowym podano w nawiasach liczoną w wyrazach długość przykładów translacyjnych.

Poniżej zamieszczono przykład tłumaczenia fragmentu tekstu z języka szwedzkiego z wykorzystaniem techniki automatycznej translacji opartej na przykładach translacyjnych.

Jag är på semester på Gotland. (2, 2, 2)

Ja jestem na wakacjach na Gotlandii.

I dag har jag varit och titat på den här stenen. (2, 6, 3)

Dzisiaj byłem popatrzeć na ten oto kamień.

Den är mycket fin med många bilder men det är inga runor på den. (4, 3, 1, 4, 2)

On jest bardzo fajny z wieloma obrazkami ale nie ma żadnych napisów runicznych na nim.

Här har varit fint väder hela tiden. (1, 4, 2)

Tutaj była ładna pogoda przez cały tydzień.

Jag har solat och badad varje dag. (5, 2)

Opalałem i kąpałem się każdego dnia.

W przypadku rozważanego tekstu w języku szwedzkim średnia długość przykładów translacyjnych wyniosła 2,81 wyrazów.

Poniżej zamieszczono przykład tłumaczenia fragmentu tekstu z języka norweskiego z wykorzystaniem techniki automatycznej translacji opartej na przykładach translacyjnych.

Om kvelden liker vi best å vare hjemme og ta det med ro. (2, 3, 3, 1, 4)

Wieczorami wolimy być w domu i mieć spokój.

Vi snakker sammen vi horer på radio eller vi ser på TV. (2, 1, 4, 1, 4)

Jemy razem słuchamy radia albo oglądamy telewizję.

Kanskje drikker far og mor kaffe ved 6-tiden. (1, 5, 2)

Czasami ojciec i matka piją kawę około godziny szóstej.

Vi får melk eller saft. (2, 1, 1, 1)

Dostajemy mleko albo sok.

W przypadku rozważanego tekstu w języku norweskim średnia długość przykładów translacyjnych wyniosła 2,24 wyrazów.

Następnie poniżej zamieszczono próbkę tłumaczenia tekstu zapisanego w języku duńskim z wykorzystaniem metody przykładów translacyjnych.

Slottet stammer fra det syttende århundrede. (2, 4)

Zamek ten pochodzi z siódmego wieku.

En beromt forfatter boede og skabte her. (6, 1)

Znany pisarz mieszkał i tworzył tutaj.

Det er udgravninger fra vikingtiden. (3, 2)

To są wykopaliska z czasów Wikingów.

W przypadku rozważanego tekstu w języku duńskim średnia długość przykładów translacyjnych wyniosła 3,00 wyrazów

Poniżej zamieszczono próbkę tłumaczenia tekstu zapisanego w języku niderlandzkim z wykorzystaniem metody przykładów translacyjnych.

Voor bewoners van Friesland is de moedertall niet Nederlands maar Fries. (4, 5, 2)

Dla mieszkańców Fryzji nie jest językiem ojczystym niderlandzki ale fryzyjski.

Dit is geen dialect het is een cultuurtaal. (4, 4)

To nie jest dialect to jest język kulturalny.

Men kan aan de universiteiten Friese taal- en lettterkunde studeren en men kann daarin doctoraalexamen doen. (2, 3, 5, 1, 2, 3)

Można na uniwersytetach studiować fryzyjską filologię i można zdawać z tej dziedziny egzamin magisterski.

Op de scholen in Friesland bij rechtszittingen in de kerk op vergaderingen van gemeenteraden en van de Provinciale Staten is heel vaak Fries de voertaal. (5, 2, 3, 9, 6)

We fryzyjskich szkołach przy posiedzeniach sądowych w kościele na zgromadzeniach rad okręgowych i rad prowincji bardzo często fryzyjski jest językiem obrad.

Aangezien alle Friezen ook Nederlands kennen is Friesland een tweetalige provincie. (1, 5, 5)

Ponieważ jednak wszyscy Fryzowie znają również język niderlandzki Fryzja jest dwujęzyczną prowincją.

W przypadku rozważanego tekstu w języku niderlandzkim średnia długość przykładów translacyjnych wyniosła 3,74 wyrazów.

Następnie technika automatycznej translacji oparta na przykładach translacyjnych została zastosowana do próbki tekstu w języku hiszpańskim.

Es un asunto un poco delicado pero te lo voy a contar. (3, 3, 1, 5)

Jest to zagadnienie trochę delikatne ale opowiem ci je.

Hace una semana llegaron los primos de Luis para pasar aqu unos das. (3, 5, 5)

Tydzień temu przyjechali kuzyni Luisa aby spędzić tu kilka dni.

Tú sabes que el estudio de Luis es muy pequeño y por eso les dije que se quedaran en mi casa. (3, 4, 3, 3, 2, 6)

Wiesz że gabinet Luisa jest bardzo mały i dlatego powiedziałem im aby zostali u mnie w domu.

La verdad es que éramos buenos amigos. (4, 3)

Prawda jest taka że byliśmy dobrymi przyjaciółmi.

W przypadku rozważanego tekstu w języku hiszpańskim średnia długość przykładów translacyjnych wyniosła 4,08 wyrazów.

Następny przykład dotyczy próby zastosowania translacji opartej na przykładach dla języka włoskiego.

É un fatto abbastanza strano per noi polacchi. (3, 2, 3)

Jest to fakt dosyć dziwny dla nas Polaków.

Da noi le differenze di cui parlo sono meno grandi. (2, 5, 3)

U nas różnice o których mówię są mniejsze.

In Italia invece mentre viaggiavo dal Nord verso il Sud mi sembrava di attraversare non uno ma più Paesi. (3, 2, 5, 8)

Natomiast we Włoszech podczas gdy podróżowałem z północy na południe wydawało mi się że przemierzam nie jeden ale więcej krajów.

Cambiava tutto il paesaggio il clima l'architettura gli abitanti il loro modo di vivere e persino la lingua. (2, 2, 2, 1, 2, 5, 2, 2)

Zmieniało się wszystko pejzaż klimat architektura mieszkańcy ich sposób życia a nawet język.

W przypadku rozważanego tekstu w języku hiszpańskim średnia długość przykładów translacyjnych wyniosła 3,00 wyrazów.

Kolejny przykład zastosowania translacji opartej na przykładach dotyczy próbki tekstu w języku francuskim.

Parfois le soir apre's une journée bien remplie j'ai l'impression de n'avoir rien fait. (3, 5, 6)

Czasami wieczorem po całym dniu mocno wypełnionym pracą mam wrażenie że nic nie zrobiłam.

Et je sius fatiguée. (1, 3)

I jestem zmęczona.

Il y a une très grande distance entre le travail que j'ai fait et le résultat. (3, 4, 9)

Jest bardzo duża różnica pomiędzy pracą którą wykonuję a jej rezultatami.

Par example j'ai prepare à manger j'ai fait la vesselle j'ai tout range je me suis occupée de la petite et il me dit tu n'as pas repassé mes chemises. (2, 4, 3, 3, 7, 4, 6)

Na przykład przygotowałam posiłek zmyłam naczynia wszystko poukładałam zajęłam się dzieckiem a on mi mówi nie wyprasowałaś mi koszul.

W przypadku rozważanego tekstu w języku francuskim średnia długość przykładów translacyjnych wyniosła 4,20 wyrazów.

Kolejny przykład zastosowania translacji opartej na przykładach dotyczy próbki tekstu w języku niemieckim.

Mit über 900.000 Einwohner ist Köln heute die drittgrößte Stadt in der Bundesrepublik Deutschland nach Hamburg und München. (4, 6, 4, 4)

Z ponad 900.000 tysiącami mieszkańców jest obecnie Kolonia trzecim co do wielkości miastem w Republice Federalnej Niemiec po Hamburgu i Monachium.

Die Stadt liegt 45 km von der Bundeshauptstadt Bon entfernt. (2, 1, 7)

Miasto leży w odległości 45 km od stolicy związku Bon.

Ihre Geschichte reicht bis in die Römerzeit zurück. (2, 1, 5)

Jego historia sięga aż do czasów rzymskich.

Früher war der Name der Stadt Colonia. (1, 5, 1)

Wcześniej nazwa miasta brzmiała Colonia.

W przypadku rozważanego tekstu w języku niemieckim średnia długość przykładów translacyjnych wyniosła 3,31 wyrazów.

Kolejny przykład zastosowania translacji opartej na przykładach dotyczy próbki tekstu w języku słowackim.

Jeden z najstaršch známych písomných dokladov o škole na Slovensku pochádza z Nitry. (1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)

Jeden z najstarszych znanych pisemnych dokumentów o szkole na Słowacji pochodzi z Nitry.

Nitra mala pri biskupskej katedrále kapitulnu školu. (1, 1, 1, 1, 1, 2)

Nitra miała przy biskupiej katedrze szkołę zakonną.

Neškor na tejto škole učili aj vzdelaní laickí učitelia. (1, 3, 1, 1, 1, 2)

Później w szkole tej uczyli również wykształceni nauczyciele świeccy.

W przypadku rozważanego tekstu w języku słowackim średnia długość przykładów translacyjnych wyniosła 1,16 wyrazów.

Kolejny przykład zastosowania translacji opartej na przykładach dotyczy próbki tekstu w języku czeskim.

Česká republika patři mezi nejkráseňjší země na světě. (2, 1, 3, 1, 1)

Czeska Republika należy najpiękniejszych krajów na świecie.

Značnou část jejího území tvoří lesy. (1, 1, 1, 1,2)

Znaczną część jej terytorium tworzą lasy.

Jsou zde také rozsáhlé nížiny s loukami pastvinami a poli. (1,1,1,1,1,1,1,1,1,1,1)

Są tam także obszerne niziny z łąkami pastwiskami i polami.

W przypadku rozważanego tekstu w języku czeskim średnia długość przykładów translacyjnych wyniosła 1,20 wyrazów.

Kolejny przykład zastosowania translacji opartej na przykładach dotyczy próbki tekstu w języku serbsko-chorwackim.

U drugoj sobi stoji krevet moga sina stolić sa dve male stolice ormar za odelo i igračke. (3, 1, 1, 2, 1, 4, 1, 2, 1, 1)

W drugim pokoju stoi łóżko mojego syna stół z dwoma małymi krzesłami szafa na ubrania i zabawki.

Tu se nalazi naš radio aparat. (1, 2, 1, 2)

Tu znajduje się nasz radioodbiornik.

Na zidu vise slike a na prozoru stoje saksije sa cvećem. (2, 2, 1, 2, 2, 2)

Na ścianie wiszą obrazy a na oknie stoją doniczki z kwiatami.

W przypadku rozważanego tekstu w języku serbsko-chorwackim średnia długość przykładów translacyjnych wyniosła 1,70 wyrazów.

Jak widać z zamieszczonych powyżej przykładów, uzyskane wstępnie wyniki wyglądają bardzo obiecująco. W żadnym z rozważanych przypadków średnia długość przykładów translacyjnych nie przekroczyła pięciu wyrazów i wynosiła typowo od trzech do czterech wyrazów dla języków z romańskiej i germańskiej grupy językowej. W przypadku języków słowiańskich długość przykładów translacyjnych była jeszcze mniejsza, czego należało się oczywiście spodziewać, w związku z dużym stopniem pokrewieństwa genetycznego tych języków.

Uzyskane rezultaty dobrze rokują dla systemów translacji automatycznej wykorzystujących metodę EBMT, zastosowanych dla języka polskiego. Niestety zgodnie z wiedzą autora, jak dotychczas nie przeprowadzono jeszcze w naszym kraju żadnych badań w tym względzie. Również badacze z Carnegi Mellon University, stosujący powszechnie różnorodne techniki automatycznej translacji nawet dla przeróżnych egzotycznych języków, nie zajęli się, jak dotąd, językiem polskim (zajmowali się natomiast między innymi językiem chorwackim).

Jedyne programy translacji automatycznej powstające w naszym kraju dotyczą przekładów pomiędzy językami polskim i angielskim. Najnowszym produktem z tej rodziny jest program English Translator 2 firmy Techland [8]. Niestety w produkcie tym zastosowano najstarszą (i prawdopodobnie najgorszą) metodę bezpośredniego transferu struktur morfologicznych, co powoduje, że ogólna jakość tłumaczonych tekstów nie jest najlepsza, a dosyć często nie można nawet odgadnąć, o co w uzyskanym w ten sposób przekładzie chodzi.

Zdaniem autora czas najwyższy zaapelować (podobnie jak uczynił to ponad 50 lat temu Warren Weaver) o podjęcie poważnych badań nad translacja automatyczna w Polsce. Metoda translacji EBMT jest stosunkowo prosta, a przy tym, jak wykazały badania przeprowadzone w innych krajach, daje bardzo dobre rezultaty. Szczególnie godny uwagi jest przypadek automatycznego tłumaczenia z zastosowaniem metody EBMT z języka hiszpańskiego na kataloński, ponieważ prawdopodobnie stosując tą metodę, można by stosunkowo tanim kosztem i niezbyt wielkim wysiłkiem intelektualnym, opracować odznaczające się wysoką jakością pracy systemy automatycznej translacji tłumaczące pomiędzy językiem polskim a innymi językami słowiańskimi.

8. ZAKOŃCZENIE

Pora zatem na podsumowanie prowadzonych w artykule rozważań. Jak widać z zamieszczonych w artykule przykładów zaczerpniętych z różnych języków świata, translacja automatyczna jest zagadnieniem o wiele bardziej trudnym, ale zarazem o wiele bardziej ciekawym niż się to powszechnie uważa. Bowiem, u podstaw konstrukcji takich systemów leżą pewne ograniczenia natury fundamentalnej, związane z tym, że zasady działania ludzkiego umysłu z wielkim trudem poddają się próbom algorytmizacji. Jest zatem rzeczą ważną, aby badacze zajmujący się problematyką systemów sztucznej inteligencji byli świadomi tych ograniczeń. Trzeba dobrze wiedzieć, czego można się po systemach sztucznej inteligencji spodziewać i czego można w związku z tym od ich twórców żądać.

Pamiętając przez cały czas o ograniczonych możliwościach algorytmów zaangażowanych w rolę substytutów ludzkiego intelektu, należy mimo wszystko wyraźnie podkreślić, że twórcy systemów sztucznej inteligencji z całą pewnością nie powiedzieli jeszcze w sprawie opracowywanych przez siebie algorytmów ostatniego słowa. Na pewno pozostało jeszcze mnóstwo pracy do wykonania, a nowopowstałe systemy sztucznej inteligencji będą zapewne coraz dokładniej i lepiej naśladować wybrane cechy intelektu człowieka. Z tego, miedzy innymi, powodu twierdzę, że w ciągu najbliższych kilkunastu lub kilkudziesięciu lat opracowane zostaną jednak programy translacji automatycznej, które będą tłumaczyć powierzone im teksty z równie wysoką sprawnością, jak istniejące już programy komputerowe potrafią grać w szachy. Zatem już w niezbyt odległej przyszłości powstanie zapewne komputerowy tłumacz, równie skuteczny jak komputerowy szachista, choć tłumaczowi podobnie, jak komputerowemu szachiście, od czasu do czasu, zdarzać się będą potknięcia. Komputery zatem będą wykorzystywane do dokonywania przekładów pomiędzy językami naturalnymi, podobnie jak obecnie są wykorzystywane do gry w szachy, chociaż, podobnie jak w przypadku rozgrywanej partii szachów, nic z przekładanych tekstów nie będą one rozumieć.

9. BIBLIOGRAFIA

1. K. L. Baker, A. M. Franz, P. W. Jordan, T. Mitamura, E. H. Nyberg: *Coping with ambiguity in a large-scale machine translation system.* Pittsburgh, Center for Machine Translation, Carnegie Mellon University, 1992
2. K. L. Baker, A. M. Franz, P. W. Jordan, T. Mitamura, E. H. Nyberg: *Coping with ambiguity in a large-scale machine translation system.* Center for Machine Translation, Pittsburgh, Carnegie Mellon University, 1992
3. D. Arnold, L. Balkan, S. Meijer, R. L. Humphreys, L. Sadler: *Machine translation: An introductory guide.* London, NCC Blachwell, 1994
4. A. Melby: *Machine translation and philosophy of language.* Machine Translation Review, No. 9, April 1999, pp. 6–17
5. M. Blekhman, B. Pevzner: *First Steps of language engineering in the USSR: The 50s through 70s.* Machine Translation Review, Issue No. 11, December 2000, pp. 5–7

6. A. Waibel, P. Geutner, L. M. Tomokiyo, T. Schultz, M. Woszczyna: *Multilinguality in speech and spoken language systems*. Proceedings of the IEEE, vol. 88, no. 8, August 2000, pp. 1297–1313
7. V. W. Zue, J. R. Glass: *Conversational interfaces: advances and challenges*. Proceedings of the IEEE, vol. 88, no. 8, August 2000, pp. 1166–1180
8. strona internetowa firmy Techland o adresie http://www.techland.com.pl

Pracę zrealizowano w ramach grantu KBN nr 8T11C 01917 „Formalne metody i narzędzia wspomagania Panalizy oraz projektowania baz danych i baz wiedzy".

M. GAJER

MACHINE TRANSLATION – THE APPROACHES OF INTERLINGUA LANGUAGE AND TRANSLATION EXAMPLES

Summary

Machine translation is a branch of science that teaches us how to program the computers, so as they were able to translate between different human languages. Machine translation was also one of the first application that was proposed for computers. Nonetheless, it soon appeared that the task of machine translation is much more difficult, but also much more interesting from the scientific point of view, than one had ever thought before. In the paper it is thoroughly explained why machine translation is so extremely hard. The most promising directions of development of machine translation systems are also briefly described. The special attention is paid to machine translation systems that are developed for Polish language.

Keywords: Machine translation, Computational linguistics, Natural language processing

unie
mów
apli
kom

naci
pod

Słow

COR
logiczną
Procesy
realizacji
cyjnych.
CORBA
ten umoż
lizowane

W ar
niane do
referencje
tury COR
Proces ap

KWARTALNIK ELEKTRONIKI I TELEKOMUNIKACJI, 2002, 48, z. 3–4, ss. 553–569

Protokoły komunikacyjne architektury CORBA

PAWEŁ PAPLIŃSKI

Instytut Telekomunikacji, Politechnika Warszawska
ul. Nowowiejska 15/19, 00-665 Warszawa
mail: ppaplins@tele.pw.edu.pl

Otrzymano 2001.12.07
Autoryzowano 2002.03.17

Podstawowym elementem architektury CORBA jest ORB – pośrednik komunikacyjny uniezależniający wymianę informacji pomiędzy częściami rozproszonej aplikacji od mechanizmów sieci transmisji danych. Sposób wykorzystywania mechanizmów komunikacyjnych przez aplikację jest określony przez interfejs do pośrednika ORB, jednak rzeczywista realizacja komunikacji w architekturze CORBA opiera się na wewnętrznych protokołach pośrednika ORB.

Praca ta stanowi opis protokołów komunikacyjnych architektury CORBA, ze szczególnym naciskiem na protokoły GIOP i IIOP — mechanizmy uznane przez twórców architektury za podstawowe.

Słowa kluczowe: CORBA, ORB, GIOP, IIOP.

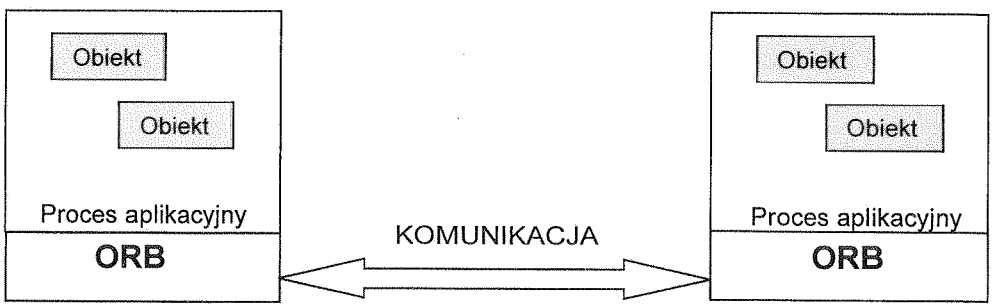
1. WPROWADZENIE

CORBA jest architekturą rozproszonych aplikacji — programów pozostających logiczną całością, podzielonych na działające równolegle części — procesy aplikacyjne. Procesy aplikacyjne są wykonywane niezależnie, jednak współdziałają ze sobą dla realizacji wspólnych zadań. Współdziałanie wymaga istnienia mechanizmów komunikacyjnych. Podstawowym mechanizmem komunikacyjnym zastosowanym w architekturze CORBA jest zdalne wywołanie procedury RPC (*Remote Procedure Call*). Mechanizm ten umożliwia procesom aplikacyjnym uruchamiać zdalnie procedury (programy) zlokalizowane w innej przestrzeni adresowej, na innym urządzeniu.

W architekturze CORBA zastosowano podejście obiektowe — procedury udostępniane do zdalnego uruchamiania pogrupowano w obiektach, identyfikowanych przez referencje, umożliwiające odnalezienie obiektu w rozproszonym środowisku architektury CORBA. Obiekty utrzymywane są przez procesy aplikacyjne nazywane serwerami. Proces aplikacyjny uruchamiający zdalnie procedurę nazywany jest klientem. Oprócz

udostępnianych procedur (nazywanych usługami) obiekt może zawierać również atrybuty — zmienne, w których przechowywane są wartości. Zestaw procedur, które są udostępniane przez dany obiekt do zdalnego uruchomienia oraz zestaw atrybutów, których wartości mogą być zdalnie zmieniane, określony jest w interfejsie obiektu. Interfejsy obiektów specyfikowane są w języku IDL.

Jednostką odpowiedzialną za przenoszenie żądania zdalnego uruchomienia procedury jest pośrednik ORB. Z poziomu procesów aplikacyjnych, ORB widziany jest jako jednolita platforma, ukrywająca fizyczne szczegóły komunikacji. W rzeczywistości, umożliwienie komunikacji pomiędzy odległymi urządzeniami wymaga implementacji pośrednika ORB w postaci mechanizmu, którego instancje znajdują się zarówno w procesach zdalnie wywołujących usługi obiektów, jak również w procesach udostępniających te obiekty. Przedstawione jest to na rysunku 1.



Rys. 1. Procesy aplikacyjne i fizyczna budowa pośrednika ORB

Częścią pośrednika ORB są protokoły komunikacyjne, odpowiadające za przenoszenie informacji potrzebnych do działania mechanizmów zdalnego wywołania procedur. Protokoły te można umiejscowić w warstwie aplikacji i prezentacji modelu OSI-RM [4]. W specyfikacji architektury CORBA określono standardowe protokoły, pozostawiając jednocześnie możliwość stosowania innych protokołów. Standardowe protokoły architektury CORBA to GIOP (*General Inter-ORB Protocol*) — protokół odpowiedzialny za przekaz pomiędzy klientem i serwerem żądań zdalnych wywołań usług obiektów oraz IIOP (*Internet Inter-ORB Protocol*) — interfejs pomiędzy protokołem GIOP a warstwą transportową TCP.

2. PROTOKÓŁ GIOP

2.1. FUNKCJONALNOŚĆ I WYMAGANIA TRANSPORTOWE

W protokole GIOP pomiędzy stronami połączenia, zestawionego w warstwie transportowej [4], przenoszone są jednostki informacji nazywane komunikatami.[1] W komunikatach przesyłane są informacje potrzebne do realizacji wywołania obiektu przez

[1] Ang. *message*

klienta.[2] Komunikaty GIOP mogą być przenoszone przez dowolny protokół warstwy transportowej, spełniający następujące założenia:

- Protokół jest zorientowany połączeniowo.
- Protokół zapewnia rzetelny przekaz.
- Przekaz może być postrzegany w postaci strumienia bajtów, przy czym protokół nie narzuca ograniczeń związanych z długością, fragmentację czy też wyrównywaniem komunikatów.
- Protokół jest wyposażony w mechanizmy powiadamiania o niespodziewanym zerwaniu połączenia.
- Mechanizm nawiązywania połączenia może być odwzorowany na mechanizm z TCP/IP.

Podstawowym założeniem konstrukcji protokołu GIOP jest jego niezależność od wszelkiego oprogramowania innego od oprogramowania warstwy transportowej, przy czym zależność od warstwy transportowej dotyczy tylko wymienionych wyżej cech protokołu tej warstwy. Dzięki takiej właściwości łatwe jest przeniesienie aplikacji, w architekturze CORBA, ze środowiska jednej sieci do innej, w której stosowany jest inny stos protokołów.

2.2. BUDOWA I DZIAŁANIE

Protokół GIOP występuje w trzech wersjach: 1.0, 1.1 i 1.2, związanych z rozwojem specyfikacji architektury CORBA. GIOP 1.0 i 1.1 różnią się pojedynczymi polami w składni komunikatów. Dodatkowo, w wersji 1.1 wprowadzono komunikat Fragment, poprawiający efektywność działania protokołu. GIOP 1.1 jest usprawnioną wersją poprzednika, do której nie wprowadzono istotnych zmian w filozofii funkcjonowania protokołu.

GIOP 1.0 i 1.1 jest protokołem asymetrycznym. Asymetria oznacza tu ścisłe określenie, w ramach jednego połączenia, która strona pełni funkcję klienta, a która serwera. Związane jest to z asymetrią modelu komunikacyjnego architektury: klient inicjuje połączenie, a serwer akceptuje je i obsługuje wywołania. W czasie trwania połączenia, przez które przesyłane są komunikaty GIOP 1.0 lub 1.1, zawsze określone jest, która strona jest klientem, a która serwerem. W zbiorze wszystkich typów komunikatów występujących w protokole GIOP określone jest, które z nich mogą być wysyłane przez serwer, a które przez klienta. Jeżeli procesy aplikacyjne pełnią obydwie role jednocześnie, zmiana kierunku komunikacji wymaga rozłączenia połączenia i nawiązania go ponownie.

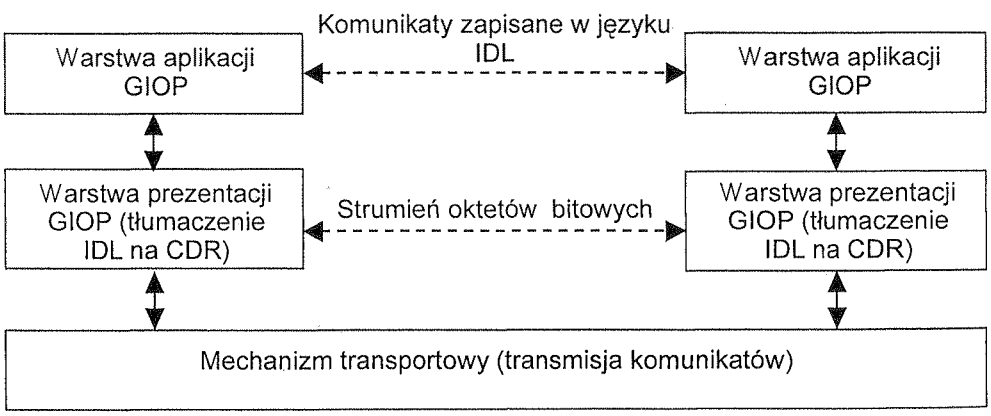
W protokole GIOP 1.2 wprowadzono możliwość komunikacji dwukierunkowej (*Bidirectional GIOP*). Klient i serwer mogą w tym przypadku zamieniać się swoimi rolami w obrębie jednego połączenia. Taka funkcjonalność może być przydatna w sytuacji, gdy obydwie strony połączenia posiadają obiekty, których usługami wymieniają się

[2] Zgodnie z modelem klient-serwer, zdefiniowanym w architekturze CORBA

wzajemnie (są jednocześnie klientem, jak i serwerem), jednak jedna z nich nie ma możliwości nawiązania połączenia (np. drugi proces aplikacyjny znajduje się za urządzeniem typu firewall). Podejście takie odchodzi od klasycznej asymetrii modelu komunikacji pomiędzy klientem i serwerem. Dalsze rozważania dotyczą wersji asymetrycznej GIOP. Umieszczony niżej opis komunikatów GIOP oparty jest o wersję GIOP 1.0.

Dla implementacji architektury CORBA wymagana jest zgodność protokołów „w dół". W wiadomości GIOP zapisana jest wersja protokołu. Serwer wspierający protokół w wersji 1.n (gdzie $n>0$ i $n<3$) musi umieć przetworzyć wiadomość w wersji 1.m, gdzie $m<n$. Serwer odbierający wywołanie z wyższym numerem wersji niż ta, którą wspiera powinien odpowiedzieć wiadomością o błędzie z najwyższym numerem wersji protokołu, który wspiera i rozłączyć połączenie.

Budowa protokołu GIOP jest dwuwarstwowa. Warstwa wyższa, którą można umiejscowić w warstwie aplikacji modelu OSI-RM, odpowiada za tworzenie i interpretację komunikatów. Komunikaty umożliwiają przenoszenie wywołań usług obiektów, modyfikację ich atrybutów oraz uzyskiwanie odpowiedzi. Dane zgodne z definicją IDL są tłumaczone na reprezentację bajtową (odpowiednie typy IDL są wyrażane w postaci strumienia oktetów bitowych), przygotowaną do transportu pomiędzy stronami połączenia, zgodną ze specyfikacją, nazwaną w specyfikacji architektury CORBA, CDR (*Common Data Representation*). Funkcjonalność mapowania na CDR można umieścić w warstwie prezentacji modelu OSI-RM. Warstwowa budowa protokołu GIOP została przedstawiona na rysunku 2.



Rys. 2. Warstwowa budowa protokołu GIOP

2.2.1. Reprezentacja bajtowa CDR

Składnia komunikatów GIOP, przesyłanych pomiędzy stronami połączenia, jest zdefiniowana przy użyciu języka IDL. W komunikatach przesyłane są wartości atrybutów, parametrów wywoływanych procedur i rezultaty zwracane przez procedury, które przekazywane są pomiędzy serwerem — udostępniającym usługi obiektów i klien-

tem — korzystającym z tych usług. Przesyłane wartości mogą być dowolnego typu prostego lub złożonego, zdefiniowanego w języku IDL. Sposób wyrażenia typów w postaci ciągu bajtów zależy od urządzenia, na którym funkcjonuje proces aplikacyjny oraz od implementacji architektury CORBA. Warstwa prezentacji GIOP ma na celu umożliwienie współpracy pomiędzy systemami różnie interpretującymi zapis bajtowy. W specyfikacji protokołu GIOP zdefiniowano sposób wyrażenia, w postaci strumienia bajtów, struktury wszystkich typów języka IDL, aby uniknąć błędów w komunikacji pomiędzy różnymi implementacjami architektury CORBA.[3] W celu uniknięcia błędnego odczytywania przekazywanych wartości, każdy komunikat GIOP zawiera pole, w którym strona przesyłająca komunikat określa przyjęty porządek bajtów.

Funkcje wyrażenia typów IDL oraz interpretacji wartości na podstawie strumienia bajtów i pola porządku bajtów wykonywane są przez warstwę prezentacji GIOP. Dodatkowo wykonywane jest tam ograniczanie zakresu typów prostych do granic, które są określone przez IDL. Ma to znaczenie np. w przypadku, gdy język programowania, w którym zaimplementowany jest dany proces aplikacyjny, pozwala przyjąć zmiennej typu prostego wartość z zakresu większego, niż jest to dopuszczone przez specyfikację IDL.

2.2.2. Składnia komunikatów

Warstwa aplikacji protokołu GIOP odpowiada za formułowanie i interpretowanie komunikatów. Komunikat GIOP poprzedzony jest nagłówkiem przedstawionym w tabeli 1. Poszczególne pola nagłówka mają opisane niżej znaczenie.

Tabela 1

Nagłówek komunikatu protokołu GIOP

```
struct MessageHeader_1_0 {
    char            magic[ 4];
    Version         GIOP_version;
    boolean         byte_order;
    octet           message_type;
    unsigned long   message_size;
};
```

magic —informacja, że komunikat należy do protokołu GIOP — pole zawiera czteroznakowy tekst „GIOP''.

GIOP_version — numer wersji protokołu.

byte_order — pole typu logicznego opisuje, w jakiej kolejności należy interpretować bajty opisujące, przenoszone w komunikacie, wartości numeryczne. Wartość FALSE oznacza kolejność big-endian, natomiast TRUE — little-endian.

message_type — typ komunikatu wg tabeli 2.

[3] Opis specyfikacji wyrażania typów IDL w reprezentacji CDR został tutaj pominięty. Znajduje się on w: [6], punkt 15.3.

message_size — rozmiar komunikatu, zapisany w postaci liczby całkowitej bez znaku.

Nagłówek komunikatu poprzedza treść komunikatu (*Message Body*). Postać treści komunikatu zależy od jego typu. W tabeli 2 wymienione są wszystkie typy komunikatów protokołu, wraz z wersjami GIOP, w których występują. Niżej przedstawione jest znaczenie i struktura poszczególnych komunikatów.

Tabela 2

Komunikaty protokołu GIOP

Typ wiadomości	Wartość pola mesage_type	Strona przesyłająca	Wersja GIOP
Request	0	klient	1.0, 1.1, 1.2
Reply	1	serwer	1.0, 1.1, 1.2
CancelRequest	2	klient	1.0, 1.1, 1.2
LocateRequest	3	klient	1.0, 1.1, 1.2
LocateReply	4	serwer	1.0, 1.1, 1.2
CloseConnection	5	serwer	1.0, 1.1, 1.2
MessageError	6	obydwie	1.0, 1.1, 1.2
Fragment	7	obydwie	1.1, 1.2

Komunikat Request

Komunikat wysyłany jest tylko od klienta do serwera,[4] w momencie gdy klient chce skorzystać z usługi obiektu. Komunikat przenosi wywołanie procedury danego obiektu lub operacje pobrania wartości atrybutu: _get_<attribute> i ustawienia wartości atrybutu _set_<attribute>, gdzie <attribute> jest nazwą atrybutu. Komunikat składa się z:

- nagłówka komunikatu GIOP (opisanego wyżej),
- nagłówka komunikatu Request (*Request Header*),
- treści komunikatu Request (*Request Body*).

Składnia nagłówka komunikatu Request przedstawiona jest w tabeli 3. Poszczególne pola mają opisane niżej znaczenie.

Tabela 3

Składnia nagłówka komunikatu Request

```
struct RequestHeader_1_0 {
   IOP::ServiceContextList      service_context;
   unsigned long                request_id;
   boolean                      response_expected;
   sequence <octet>             object_key;
   string                       operation;
   Principal                    requesting_principal;
};
```

[4] W asymetrycznych wersjach protokołu: GIOP 1.0 i 1.1.

service_context — pole służy do przesyłania specyficznych informacji związanych ze sposobem obsługi wywołania obiektu i jest wykorzystywane np. przez usługi architektury CORBA, pośredniczące w wywołaniu.

request_id — pole zawiera identyfikator wywołania, zapisany w postaci liczby całkowitej, istotny dla przyporządkowania otrzymywanych odpowiedzi do wysyłanych wywołań Request. Identyfikator ten jest potrzebny, ponieważ poprzez jedno połączenie transportowe może zostać wysłanych od klienta do serwera wiele wywołań.

response_expected — pole typu logicznego. Zawiera informacje, czy klient oczekuje odpowiedzi na wywołanie.

object_key — pole jest sekwencją bajtów (sequence<octet>) identyfikującą obiekt, którego dotyczy wywołanie. Sposób wypełnienia pola zależy od stosowanego protokołu transportowego. W przypadku przesyłania komunikatu GIOP przez połączenie TCP, w pole object_key wpisywany jest identyfikator obiektu pozyskany z profilu IIOP, znajdującego się w referencji do obiektu, którą otrzymał klient.

operation — pole jest ciągiem znaków, w który wpisywana jest nazwa procedury wywoływanej przez klienta lub operacje _get_<attribute>, bądź _set<attribute>, gdzie <attribute> jest nazwą atrybutu, którego dotyczy wywołanie.

requesting_principal — pole pochodzi ze starszych wersji architektury CORBA i związane jest ze stosowaniem adaptera obiektów BOA (*Basic Object Adaptor*).[5] W specyfikacji architektury w wersji 2.3 nie zaleca się stosowania adaptera BOA, a zatem pole requesting_principal nie powinno być stosowane. W GIOP 1.2 pole to nie występuje.

Po nagłówku komunikatu Request przesyłana jest treść komunikatu Request. Znajdują się w niej wartości parametrów wywoływanej akcji, zakodowane zgodnie z formatem CDR.

Komunikat Reply

Komunikat Reply jest odpowiedzią na komunikat Request, przesyłaną przez serwer do klienta w sytuacji, kiedy odpowiedź jest wymagana. Komunikat Reply składa się z:

- nagłówka komunikatu GIOP,
- nagłówka komunikatu Reply (*Reply Header*),
- treści komunikatu Reply (*Reply Body*).

Składnia nagłówka komunikatu Reply przedstawiona jest w tabeli 4. Poszczególne pola mają opisane niżej znaczenie.

Tabela 4

Składnia nagłówka komunikatu Reply

```
struct ReplyHeader_1_0 {
   IOP::ServiceContextList      service_context;
   unsigned long                request_id;
   ReplyStatusType_1_0          reply_status;
};
```

[5] Adapter obiektów BOA był poprzednikiem (o uproszczonej funkcjonalności) adaptera obiektów POA, zdefiniowanym w starszych niż 2.3 specyfikacjach architektury CORBA

service_context — pole pełni funkcję analogiczną do pola o tej samej nazwie, które znajduje się w nagłówku komunikatu Request.
request_id — dzięki informacji zapisanej w polu, które zawiera całkowitoliczbowy numer wywołania, możliwe jest przyporządkowanie komunikatu Reply do komunikatu Request, którego dotyczy odpowiedź.
reply_status — pole zawiera informację o charakterze odpowiedzi. Możliwe są cztery rodzaje odpowiedzi, zapisane w typie pola: ReplyStatusType. Definicja tego typu przedstawiona jest w tabeli 5. Stałe opisujące charakter odpowiedzi mają następujące znaczenie:

Tabela 5

Definicja typu ReplyStatusType

```
enum ReplyStatusType_1_0 {
  NO_EXCEPTION,
  USER_EXCEPTION,
  SYSTEM_EXCEPTION,
  LOCATION_FORWARD
};
```

NO_EXCEPTION — operacja powiodła się: w treści komunikatu znajdują się zwracane przez usługę (operację udostępnianą przez obiekt) wartości, jako struktura wyjściowych parametrów operacji;
USER_EXCEPTION lub SYSTEM_EXCEPTION — serwer zwrócił błąd (wyjątek): w treści komunikatu znajdują się informacje o błędzie;
LOCATION_FORWARD — fizyczna lokalizacja obiektu zmieniła się: w treści komunikatu znajduje się nowa referencja do danego obiektu.

Treść wiadomości zawiera rodzaj informacji zależny od charakteru odpowiedzi.

Komunikat CancelRequest

Komunikat CancelRequest przesyłany jest tylko przez klienta do serwera,[6] informując serwer o skasowaniu uprzednio zgłoszonego wywołania o danym identyfikatorze request_id. Komunikat składa się z:

- nagłówka komunikatu GIOP,
- nagłówka komunikatu CancelRequest (*Cancel Request Header*).

Drugi z powyższych elementów zawiera jedynie identyfikator wywołania Request. W tabeli 6 przedstawiona jest składnia nagłówka Komunikatu ConcelRequest.

Tabela 6

Składnia nagłówka komunikatu CencelRequest

```
struct CancelRequestHeader {
    unsigned long          request_id;
};
```

[6] W asymetrycznych wersjach protokołu: GIOP 1.0 i 1.1.

Komunikat LocateRequest

Komunikat LocateRequest stanowi uproszczoną wersję komunikatu Request, stosowaną w sytuacji, gdy klient posiada informację, że poszukiwany obiekt zmienił fizyczną lokalizację i chce jedynie uzyskać jego nową referencję, bez zbędnego przesyłania wszystkich informacji zawartych w komunikacie Request. Komunikat LocateRequest składa się z:

- nagłówka komunikatu GIOP,
- nagłówka komunikatu LocateRequest (*Locate Request Header*).

Składnia nagłówka komunikatu LocateRequest ma postać przedstawioną w tabeli 7. Obydwa pola zawarte w nagłówku mają identyczną interpretację jak pola o tej samej nazwie, znajdujące się w nagłówku komunikatu Request.

Tabela 7

Składnia nagłówka komunikatu LocateRequest

```
struct LocateRequestHeader_1_0 {
  unsigned long                request_id;
  sequence <octet>             object_key;
};
```

Komunikat LocateReply

W odpowiedzi na komunikat LocateRequest serwer przesyła komunikat LocateReply. Komunikat LocateReply składa się z:

- nagłówka komunikatu GIOP,
- nagłówka komunikatu LocateReply (*Locate Reply Header*),
- treści komunikatu LocateReply (*Locate Reply Body*).

Składnia nagłówka komunikatu LocateReply ma postać przedstawioną w tabeli 8. Poszczególne pola mają opisane niżej znaczenie.

Tabela 8

Składnia nagłówka komunikatu LocateReply

```
struct LocateReplyHeader_1_0 {
  unsigned long                request_id;
  LocateStatusType_1_0         locate_status;
};
```

request_id — identyfikator wywołania, którego dotyczy odpowiedź.

locate_status — informacja o rodzaju odpowiedzi na zapytanie o lokalizację obiektu. Pole zawiera jedną z wartości określonych w definicji typu wyliczeniowego Locate StatusType. Definicja tego typu ma postać przedstawioną w tabeli 9. Zdefiniowane stałe mają następujące znaczenie:

UNKNOWN_OBJECT — obiekt nie jest znany po stronie serwera: treść komunikatu LocateReply jest pusta;

Tabela 9

Definicja typu LocateStatusType

```
enum LocateStatusType_1_0 {
  UNKNOWN_OBJECT,
  OBJECT_HERE,
  OBJECT_FORWARD
};
```

OBJECT_HERE — obiekt znajduje się bezpośrednio na serwerze, do którego skierowane zostało wywołanie LocateRequest — treść komunikatu jest pusta;
OBJECT_FORWARD — obiekt zmienił lokalizację — treść komunikatu LocateReply zawiera nową referencję obiektu.

Komunikat CloseConnection

Komunikat CloseConnection może być wysyłany tylko przez serwer.[7] Komunikat ten informuje klienta o zamiarze zerwania połączenia i braku możliwości odpowiedzi na wszelkie wywołania. Klient, po otrzymaniu komunikatu, wie, że nie zostaną również udzielone odpowiedzi na wywołania już przesłane. Komunikat składa się jedynie z nagłówka komunikatu GIOP, zawierającego, w polu message_type, informację identyfikującą typ komunikatu. Treść komunikatu pozostaje pusta.

Komunikat MessageError

Komunikat jest przesyłany w sytuacji niezgodności wersji GIOP lub błędnego wypełnienia nagłówka wiadomości. Komunikat MessageError składa się jedynie z nagłówka komunikatu GIOP. Treść komunikatu pozostaje pusta.

Komunikat Fragment

Dodany w wersjach GIOP 1.1 i 1.2 komunikat umożliwia przesłanie informacji potrzebnych do wywołania obiektu lub odpowiedzi na wywołanie w kilku komunikatach (fragmentami) — dzielenia komunikatów na części. Ponieważ w niniejszym tekście skoncentrowano się na najprostszej wersji protokołu GIOP 1.0, mechanizm fragmentacji komunikatów nie będzie omawiany.

2.2.3. Przykład

Poniżej przedstawiono dwa przykłady komunikacji pomiędzy klientem i serwerem przy użyciu protokołu GIOP. W tabeli 10 znajduje się deklaracja prostego obiektu w języku IDL. Obiekt example_object zawiera procedurę print_string drukującą

Tabela 10

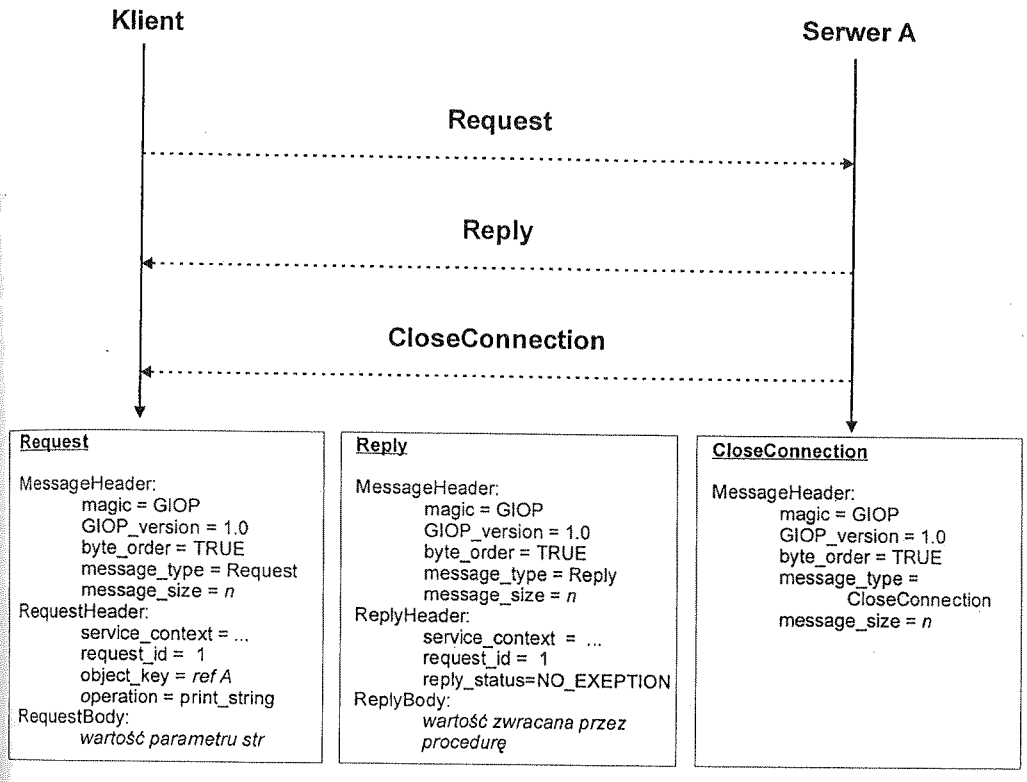
Deklaracja przykładowego obiektu

```
interface example_object {
    boolean print_string ( in string str);
};
```

[7] W asymetrycznych wersjach protokołu: GIOP 1.0 i 1.1.

łańcuch znaków podany jako parametr. Procedura zwraca wartość logiczną świadczącą o poprawnym bądź niepoprawnym wykonaniu operacji.

Na rysunku 3 przedstawiono przebiegi czasowe komunikacji pomiędzy procesem klienta a serwerem, utrzymującym instancję obiektu example_object. Na rysunku przedstawiono również wartości poszczególnych pól komunikatów.[8] Klient wywołuje



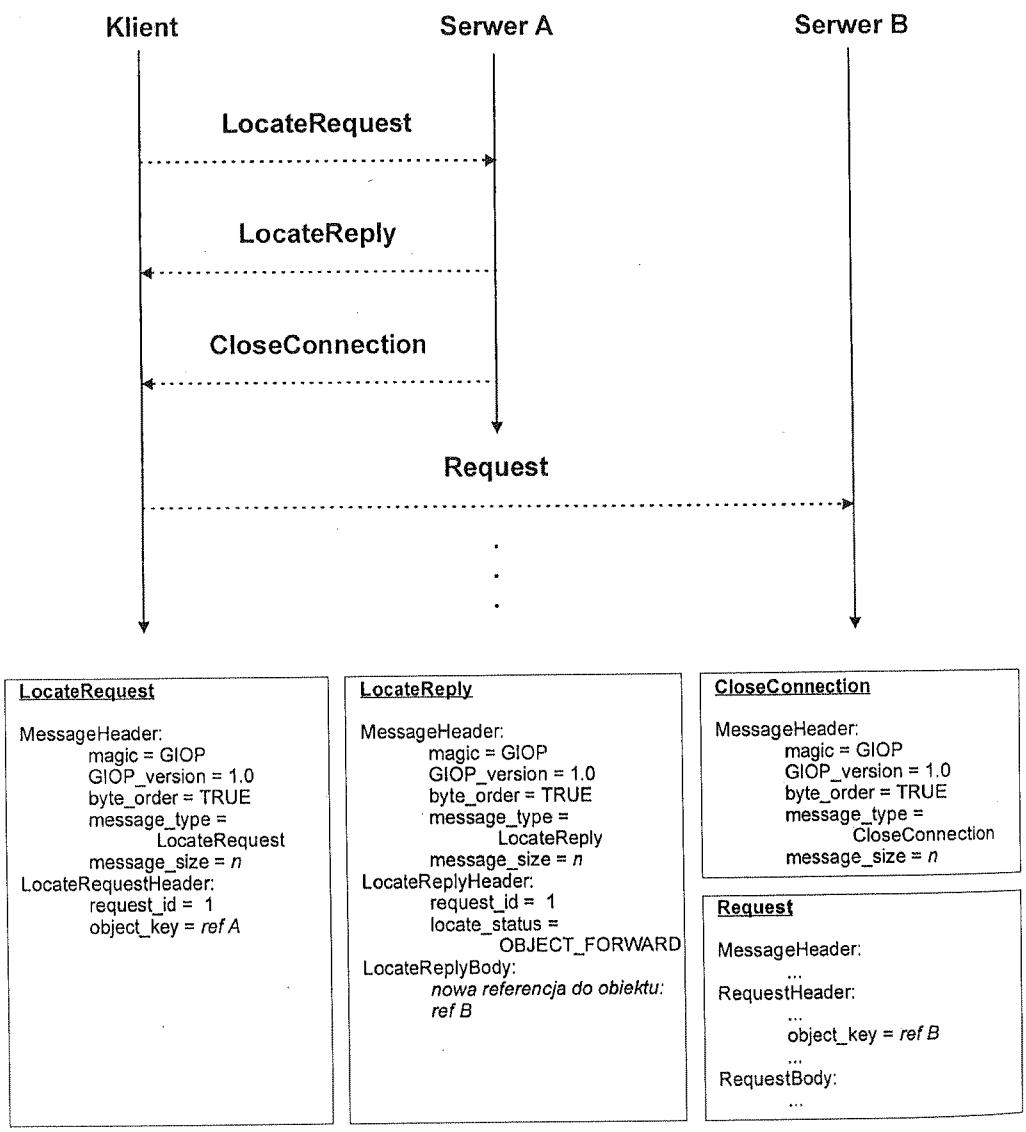
Rys. 3. Przebieg wymiany komunikatów GIOP przy wywołaniu procedury obiektu

procedurę print_string, przekazując w treści komunikatu Request wartość ciągu znaków. Założono, że klient zna prawidłową referencję do obiektu utrzymywanego przez serwer A, którą na rysunku oznaczono symbolicznie jako *ref A*. Procedura została wykonana poprawnie, o czym świadczy wartość pola reply_status w komunikacie Reply. W przedstawionym przykładzie serwer zamyka połączenie komunikatem CloseConnection, co jednak nie musi się wydarzyć — komunikacja może odbywać się dalej w oparciu o otwarte połączenie TCP.

W przypadku kiedy klient posiada informację o przeniesieniu fizycznej lokalizacji obiektu i jednocześnie zna nieaktualną referencję do poszukiwanej instancji

[8] Długość przekazywanych komunikatów oznaczono symbolicznie jako *n*. Nie przedstawiono wartości pól service_context, które w niniejszej pracy nie zostały szczegółowo omówione

obiektu w serwerze A (oznaczoną symbolicznie jako *ref A*), może przesłać komunikat LocateRequest. Sytuacje taką przedstawiono na rysunku 4. Serwer odpowiada komunikatem LocateReply, przesyłając w treści komunikatu LocateReply nową referencję (*ref B*). Na rysunku 4 zaznaczono zamknięcie połączenia przez serwer A, co analogicznie jak w poprzednim przykładzie, nie musi się wydarzyć. Po otrzymaniu nowej referencji, klient prowadzi komunikację z serwerem B analogicznie jak w pierwszym przykładzie.



Rys. 4. Przebieg wymiany komunikatów GIOP przy zmianie lokalizacji obiektu

3. PROTOKÓŁ IIOP

3.1. FUNKCJONALNOŚĆ I CECHY

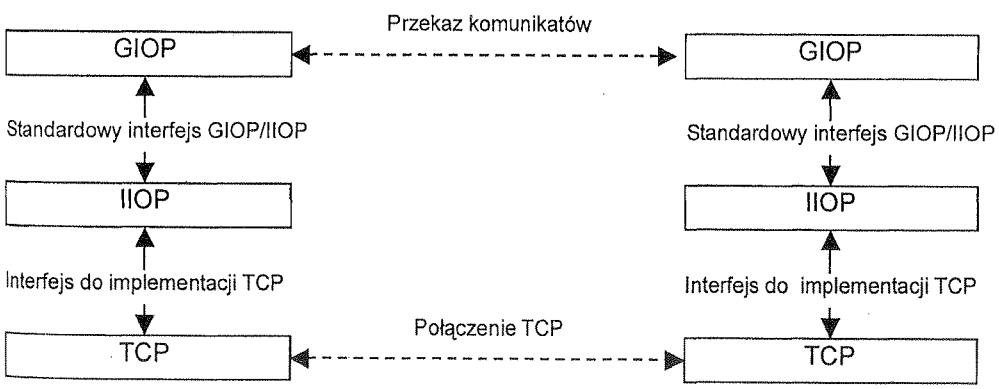
Wiadomości GIOP są przesyłane przez protokoły niższych warstw. Ze względu na popularność architektury TCP/IP, zdecydowano się na przyjęcie TCP/IP jako podstawowego mechanizmu transportowego dla architektury CORBA. Dlatego też zdefiniowany w specyfikacji mechanizm IIOP (*Internet Inter-ORB Protocol*) określający sposób nawiązywania połączenia TCP w celu przekazania komunikatów GIOP oraz umieszczanie komunikatów w segmentach TCP.

Zdefiniowanie IIOP nie ogranicza możliwości wyboru protokołu transportowego. Istotną cechą architektury CORBA jest oddzielenie mechanizmów zależnych od protokołu transportowego od mechanizmów uniwersalnych. IIOP jest pierwszym z wymienionych typów mechanizmów. GIOP jest mechanizmem niezależnym od platformy transportowej.

Producenci implementacji architektury CORBA mogą tworzyć własne rozwiązania, oparte o inne architektury teleinformatyczne (np. OSI). Zgodnie ze specyfikacją, implementacje architektury CORBA muszą jednak być wyposażone w mechanizmy GIOP oraz IIOP tak, aby istniała możliwość współpracy pomiędzy różnymi implementacjami.

3.2. BUDOWA I DZIAŁANIE

Mechanizm IIOP jest, podobnie jak GIOP, częścią pośrednika ORB. IIOP znajduje się pomiędzy warstwą protokołu GIOP, a warstwą TCP. Implementacje protokołu TCP różnią się interfejsem, który udostępniany jest warstwom wyższym. IIOP ujednolica sposób, w jaki GIOP odwołuje się do warstwy transportowej, przekazując komunikaty GIOP do warstwy TCP. Usytuowanie IIOP względem GIOP i TCP przedstawione jest na rysunku 5.



Rys. 5. Usytuowanie IIOP względem warstw GIOP i TCP

Instancja protokołu IIOP pełni następujące funkcje:

- W wyniku żądania przesłania komunikatu Request lub LocateRequest:
  - na podstawie referencji do obiektu identyfikuje maszynę (hosta) i port TCP na tej maszynie, z którym należy nawiązać połączenie;
  - zleca warstwie TCP zestawienie połączenia, używając interfejsu udostępnianego przez daną implementację protokołu TCP;
  - przekazuje komunikat warstwie TCP i zleca jego przesłanie poprzez zestawione połączenie.
- W wyniku żądania przesłania pozostałych komunikatów:
  - na podstawie pola request_id w komunikacie GIOP identyfikuje, które połączenie użyć do przesłania danego komunikatu;
  - przekazuje komunikat warstwie TCP i zleca jego przesłanie poprzez zestawione połączenie.
- Dodatkowo, po przesłaniu komunikatu CloseConnection, instancja protokołu IIOP przekazuje warstwie TCP żądanie zamknięcia połączenia natychmiast lub po otrzymaniu potwierdzenia zamknięcia połączenia po drugiej stronie łącza (w zależności od implementacji i konfiguracji pośrednika ORB).
- W przypadku otrzymania komunikatu CloseConnection instancja IIOP natychmiast przekazuje warstwie TCP żądanie zamknięcia połączenia, ewentualnie żądając przesłania potwierdzenia wykonywania tej operacji na drugą stronę połączenia.

W przypadku udostępnienia obiektu przez proces aplikacyjny, instancja IIOP, znajdująca się w instancji pośrednika ORB obsługującej ten proces, zleca jednostce protokołu TCP otwarcie portu i przekazywanie przychodzących do tego portu informacji, które mogą być komunikatami GIOP. Informacje potrzebne do nawiązania połączenia TCP/IP z serwerem znajdują się w referencji obiektu. Zapisany jest w niej tzw. profil IIOP, zawierający między innymi adres IP urządzenia, na którym działa serwer oraz numer portu TCP. Protokół IIOP występuje w trzech wersjach odpowiadających wersjom protokołu GIOP. Profil IIOP dla wersji protokołu 1.0 przedstawiony jest w tabeli 11. Poszczególne pola mają opisane niżej znaczenie.

Tabela 11

Profil IIOP dla wersji protokołu 1.0

```
struct ProfileBody_1_0 {
  Version                 iiop_version;
  string                  host;
  usigned short           port;
  sequence <octet>        object_key;
};
```

iiop_version — pole zawiera numer wersji protokołu IIOP.
host, port — pola opisane wyżej.
object_key — sekwencja zawiera wartość, stosowaną do identyfikacji obiektu, do którego skierowane jest wywołanie.

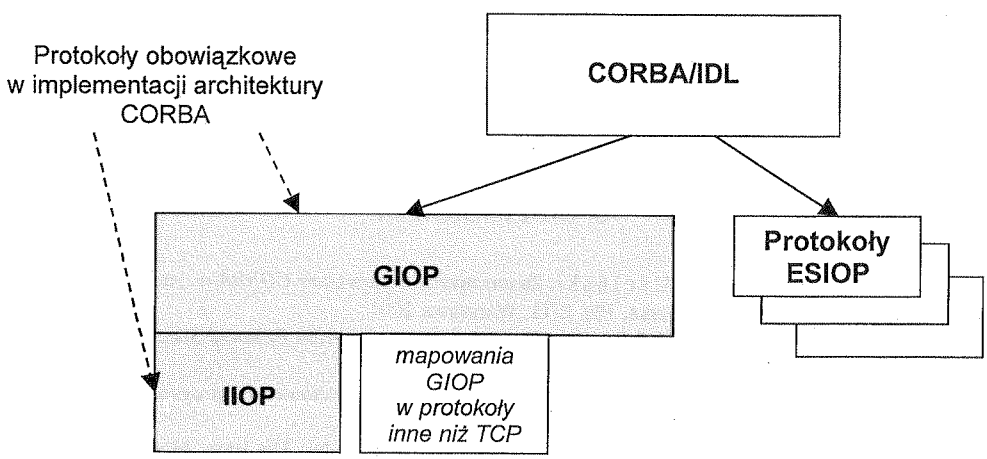
W IIOP 1.2 dodano w profilu IIOP sekwencję:

sequence< IOP::TaggedComponent> components;

Zawiera ona określone znaczniki (ze zdefiniowanego, zamkniętego zbioru). Rodzaj znaczników znajdujących się w danej sekwencji stanowi dodatkowe informacje, które mogą być wykorzystane przez pośrednika ORB do świadczenia usług, związanych np. z bezpieczeństwem (np. informacja o konieczności szyfrowanie transmisji komunikatów związanych z wywołaniem usługi danego obiektu).

4. INNE PROTOKOŁY ARCHITEKTURY CORBA

Możliwości komunikacji pomiędzy pośrednikami ORB nie ograniczono jedynie do protokołów GIOP/IIOP. Wprowadzono otwarty zbiór protokołów specyficznych dla środowiska — ESIOP (*Environment Specific Inter-ORB Protocol*), które mogą być stosowane, jeżeli komunikacja pomiędzy pośrednikami przy użyciu protokołu GIOP jest, w konkretnym przypadku, gorszym rozwiązaniem (np. ze względów wydajnościowych). W specyfikacji zdefiniowano jeden protokół z rodziny ESIOP — dla środowiska DCE[9] — DCE-CIOP (*DCE Common Inter-ORB Protocol*). Zależności pomiędzy protokołami komunikacji pomiędzy pośrednikami ORB przedstawione są graficznie na rysunku 6.



Rys. 6. Zależności pomiędzy protokołami komunikacyjnymi architektury CORBA

[9] DCE (*Distributed Computing Environment*) jest przemysłowym standardem, opracowanym przez organizację OSF (Open Software Fundation), opisującym mechanizmy umożliwiające zarządzanie przetwarzaniem i wymianą danych w rozproszonym systemie komputerowym. Architektura CORBA w takim środowisku może być traktowana jako nakładka, która w miejsce swoich mechanizmów wprowadza i wykorzystuje analogiczne mechanizmy środowiska DCE

Składnia komunikatów protokołów GIOP oraz ESIOP zdefiniowana jest zawsze w języku IDL. GIOP przenoszony jest przez stosowany protokół warstwy transportowej. Specyfikacja architektury CORBA wymaga aby każda implementacja była wyposażona w protokoły GIOP oraz IIOP. Nie narzuca ograniczeń dotyczących dostępnego zbioru pozostałych protokołów.

5. PODSUMOWANIE

W pracy przedstawiono opis protokołów komunikacyjnych architektury CORBA. Określono rolę jaką pełnią wewnętrzne protokoły pośrednika ORB i jaki jest ich związek z działaniem rozproszonych aplikacji opartych o architekturę CORBA. Na podstawie specyfikacji architektury CORBA przeprowadzono analizę protokołu GIOP, którą poprzedzono sformułowaniem wymagań, jakie musi spełniać protokół warstwy transportowej, aby możliwe było wykorzystanie go do przesyłania informacji z protokołem GIOP. W analizie skoncentrowano się na składni i znaczeniu komunikatów protokołu GIOP w wersji 1.0. Określono również rolę jaką pełni warstwa prezentacji protokołu GIOP, w której zastosowano reprezentację bajtową CDR. Analizę protokołu GIOP uzupełniono przedstawieniem funkcjonalności protokołu IIOP. Sformułowano wniosek, że protokół IIOP jest elementem, uniezależniającym działanie protokołu GIOP od implementacji protokołu warstwy transportowej TCP.

W końcowej części pracy omówione zostały w skrócie inne mechanizmy komunikacyjne dopuszczone do stosowania w architekturze CORBA. Przedstawiono ogólny model odniesienia, określający zależności pomiędzy poszczególnymi mechanizmami.

6. BIBLIOGRAFIA

1. R. Idziak, P. Papliński, T. Uściński: *Zastosowanie architektury CORBA w zarządzaniu systemami SDH*. Praca Dyplomowa Magisterska, PW EiTI, Warszawa 2001
2. J. Postel: *Transmission Control Protocol – DARPA Internet Program Protocol Specification*. RFC-793, Information Sciences Institute, September 1981
3. S. Vinoski: *CORBA: Integrating Diverse Applications Within Distributed Heterogeneous Environments*. IEEE Communications Magazine, Vol. 35, No. 2, February 1997
4. ITU-T X.200, Open Systems Interconnection – Basic Reference Model: The Basic Model
5. Object Management Group (OMG): A Discussion of the Object Management Architecture, January 1997
6. Object Management Group (OMG): The Common Object Request Broker: Architecture and Specification, Revision 2.3.1, October 1999
7. The Open Group: DCE Overview, http://www.opengroup.org/dce/info/papers/tog-dce-pd-1296.htm, 1996

P. PAPLIŃSKI

COMMUNICATION PROTOCOLS OF COMMON OBJECT REQUEST BROKER ARCHITECTURE

Summary

Object Request Broker (ORB) is the main element of Common Object Request Broker Architecture (CORBA). ORB is a communication intermediate broker making transmission of information among distributed application parts independent on mechanisms of data transmission network. The way of using of communication mechanisms by an application is determinate by the ORB interface, but the real implementation of communication mechanisms in CORBA is based on internal ORB protocols.

This work is a description of CORBA communication protocols, focused on GIOP and IIOP protocols – mechanisms treated by artificers of CORBA as the capital ones.

Keywords: CORBA, ORB, GIOP, IIOP.

KWARTALNIK ELEKTRONIKI I TELEKOMUNIKACJI, 2002, 48, z. 3–4, ss. 571–589

Implementacja układów dodających wchodzących w skład konwolwera w układach programowalnych FPGA

KAZIMIERZ WIATR, ERNEST JAMRO

Katedra Elektroniki, Akademia Górniczo-Hutnicza w Krakowie
Al. Mickiewicza 30, 30-059 Kraków

Otrzymano 2001.09.21
Autoryzowano 2002.02.02

Operacja dodawania jest podstawową operacją wykonywaną podczas obliczania operacji konwolucji (filtracji typu FIR) o stałych współczynnikach. W układach FPGA operacja dodawania powinna być implementowana z wykorzystaniem układu dodającego z przeniesieniem skrośnym RCA (ang. *Ripple Carry Adder*), w porównaniu z układami ASIC, dla których optymalną architekturą jest układ dodający z przechowaniem przeniesienia CSA (ang. *Carry Save Adder*). W konsekwencji w niniejszym opracowaniu zostały przedstawione różne algorytmy znajdujące optymalną sieć połączeń w bloku dodającym: przeszukiwania wyczerpującego ES (ang. *Exhaustive Search*), algorytmu zachłannego GrA (ang. *Greedy Algorithm*). Ponadto zostały przedstawione różne architektury układu konwolwera w układach FPGA oraz ich wpływ na parametry wejściowe układu dodającego, w szczególności zakresu danych wejściowych (wartość minimalna i maksymalna) oraz korelacji pomiędzy wejściami.

Słowa kluczowe: procesory specjalizowane, architektury systemów procesorowych, programowalne układy logiczne, arytmetyka rozproszona, systemy czasu rzeczywistego

1. WSTĘP

1.1. IMPLEMENTACJA OPERACJI KONWOLUCJI W UKŁADACH FPGA

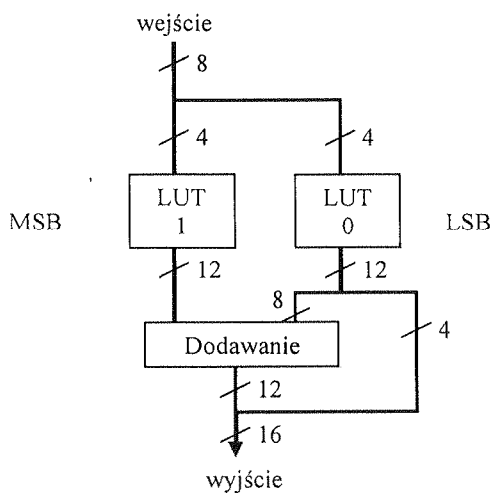
Operacja konwolucji może być przedstawiona jako suma iloczynów odpowiednio opóźnionego wejścia:

$$y(i) = \sum_{k=0}^{N-1} h(k) \cdot x(i-k) \tag{1}$$

gdzie: $y(i)$, $x(i)$ oraz $h(k)$ odpowiednio: odpowiedź, wejście w czasie i współczynniki konwolucji.

Mnożenie we wzorze (1) może być zaimplementowane poprzez wykorzystanie trzech różnych technik:

1. Mnożenie bezmnożne MM (ang. *Multiplierless Multiplication*) [1] dla którego mnożenie wykonywane jest jako przesunięcia i dodawania argumentu wejściowego, których struktura określona jest na podstawie współczynnika mnożenia. Dla przykładu; A pomnożone przez $B = 14 = 1110_2$ jest realizowane w postaci $(A<<1)+(A<<2)+(A<<3)$, gdzie „$<<n$" — oznacza przesunięcie bitowe w lewo o n bitów. W celu zredukowania liczby operacji (niezerowych bitów w binarnej reprezentacji współczynnika mnożenia), stosuje się postać kanoniczną ze znakiem CSD (ang. *Canonic Sign Digit*) [2], dla której oprócz bitów 0, 1 występujących w reprezentacji binarnej wprowadzono dodatkowy symbol $\bar{1}$, który symbolizuje operacje odejmowania. Podsumowując, dla architektury MM operacja konwolucji jest wykonywana poprzez wykorzystanie tylko operacji dodawania i odejmowania.
2. Mnożenia oparte o pamięć LUT (ang. *Look-Up Table*) LM (ang. *LUT — based Multiplication*). Teoretycznie, obliczenie dowolnej skończonej funkcji może być zaimplementowane z użyciem pamięci LUT, dla której wartość argumentu wejściowego jest podawana na wejście adresowe, a rezultat funkcji pobierany jest z wyjścia pamięci, która została uprzednio odpowiednio zakodowana. Niestety, użycie pojedynczej pamięci LUT podczas operacji mnożenia może być praktyczne tylko dla małej szerokości (ilości bitów) argumentu wejściowego, jako że rozmiar pamięci LUT wzrasta wykładniczo wraz ze wzrostem szerokości danej wejściowej. Dlatego w praktyce argument wejściowy poddaje się podziałowi, stosując większą liczbę mniejszych pamięci LUT oraz dodatkowe drzewo sumujące [1, 3, 4, 5]. Pokazuje to przykład przedstawiony na rysunku 1.



Rys. 1. Mnożenie LM dla 8-bitowego wejścia i 8-bitowego współczynnika mnożenia

3. Arytmetyka rozproszona DA (ang. *Distributed Arithmetic*) [5, 6]. Dla DA operacja konwolucji jest obliczana w innej kolejności niż w standardowym podejściu LM. Zastosowana jest następująca matematyczna transformacja:

$$y(i) = \sum_{k=0}^{N-1} h(k) \cdot x(i-k) = \sum_{j=0}^{L-1} 2^j \cdot \sum_{i=0}^{N-1} h(k) \cdot x_j(i-k) \tag{2}$$

gdzie: L — szerokość danej wejściowej x w bitach, $x_j(i-k)$ — j-ty bit danej wejściowej w czasie $(i-k)$.

W konsekwencji architektura DA wykonuje operacje konwolucji w sekwencji bitowej, tzn. każdy bit danej wejściowej jest rozpatrywany oddzielnie. W porównaniu z architekturą LM, szerokość wyjściowa pamięci LUT dla architektury DA jest mniejsza, jako że dane wejściowe mają taką samą wagę bitową, dlatego wymagane są mniejsze pamięci przez co architektura DA jest bardziej efektywna niż architektura LM.

Dla powyższych trzech technik, operacja dodawania jest bardzo często zaniedbywana i nie poddawana żadnej dyskusji. Dla przykładu w [5] pokazano strukturę architektur LM i DA i budowę drzewa dodającego, jednakże kolejność dodawania wydaje się być raczej intuicyjna niż oparta na dogłębnych badaniach. W konsekwencji w artykule tym zamieszczono dogłębne studium parametrów bloku dodającego oraz sposobów jego optymalizacji. W rezultacie skonstruowano narzędzie automatycznie znajdujące optymalną strukturę bloku dodającego oraz uwzględniającego parametry wejściowe różne dla różnych architektur.

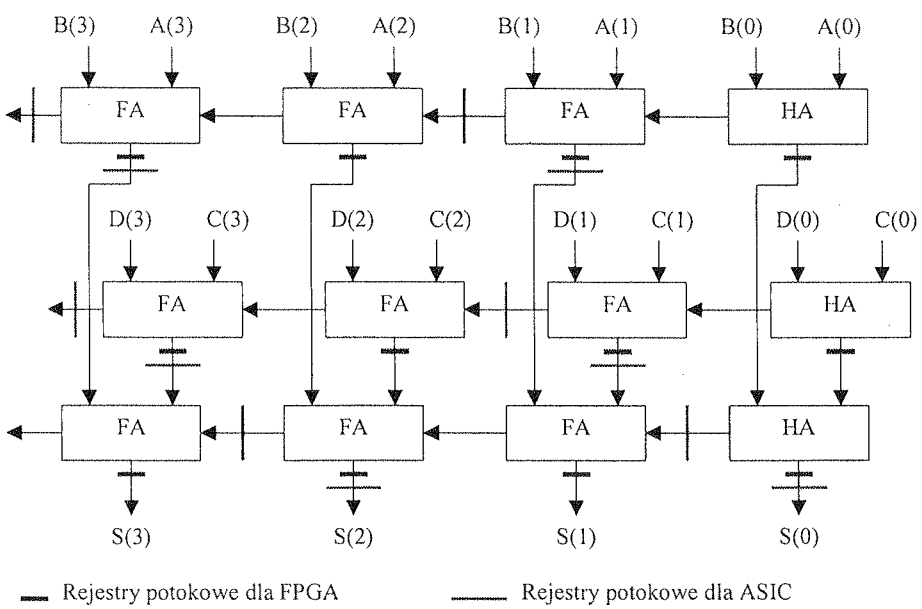
1.2. OPERACJA DODAWANIA W UKŁADACH FPGA

W układach ASIC (ang. *Application Specific Integrated Circuit*) klasyczny problem propagacji przeniesienia jest rozwiązywany w różnoraki sposób, np. z użyciem układów dodających z antycypacją przeniesienia CLA (ang. *Carry Look Ahead*) czy też z przełączaniem przeniesienia CS (ang. *Carry Select*) [4], w wyniku czego redukuje się czas propagacji przeniesienia kosztem większej powierzchni zajmowanej przez układ dodający. Innym podejściem jest całkowite wyeliminowanie problemu przeniesienia poprzez zastosowanie układów dodających przechowujących przeniesienie CSA (ang. *Carry-Save Adder*). W konsekwencji w układach ASIC w konwolwerach stosuje się głównie układy CSA [7].

Programowalne układy logiczne FPGA (ang. *Field Programmable Gate Array*) posiadają dedykowany układ propagacji przeniesienia skrośnego RCA (ang. *Ripple Carry Adder*) [8, 9], który jest tak szybki i efektywny, że konwencjonalne metody przyspieszenia propagacji przeniesienia są bezużyteczne dla układów dodających o szerokości 16 bitów i mają marginalne znaczenie dla układów dodających o szerokości 32 bity [8]. Co więcej, dedykowany układ propagacji przeniesienia jest implementowany poza standardową logiką pamięci LUT, co w rezultacie powoduje, że nie zajmuje on

dodatkowej logiki. Podsumowując, używanie układów dodających typu RCA jest najlepszym rozwiązaniem ze względu na czas propagacji i zajmowaną powierzchnię [10].

W rezultacie wbudowanie szybkiego układu generacji przeniesienia w układach FPGA, występuje fundamentalna różnica w architekturze potokowej układu dodającego RCA dla układów FPGA i ASIC. W układach ASIC rejestry potokowe powinny być wstawiane co N bloków logiki (gdzie N — liczba naturalna zależna od aplikacji), dzięki czemu łańcuch szeregowej propagacji przeniesień jest skrócony (rys. 2). Dla układów FPGA szybki układ propagacji przeniesienia w dużym stopniu redukuje czas propagacji przeniesienia przez co rejestry potokowe powinny być wstawiane co K dodawań (z pominięciem ścieżki propagacji przeniesienia). Niemniej dedykowany układ generacji przeniesienia nie jest w stanie całkowicie wyeliminować czasu szeregowej propagacji przeniesienia, który narasta wraz z szerokością układu dodającego. Dlatego ścieżka propagacji przeniesienia jest krytyczna i może wpływać na czas propagacji całego układu dodającego. Dla przykładu, dla układu FPGA firmy Xilinx serii XC4000, opóźnienie przez logikę LUT, np. powstałe w ramach generacji sumy w układzie pełnego sumatora FA (ang. *Full Adder*) jest około sześć razy większe niż dla układu dedykowanego przeniesienia szeregowego w FA. Po uwzględnieniu opóźnień w układzie FPGA spowodowanych przez programowalne połączenia, czas propagacji przeniesienia jest jeszcze mniej znaczący. Jest tak po pierwsze dlatego, że opóźnienie wnoszone przez programowalne połączenia jest często porównywalne (a niekiedy nawet dużo



Rys. 2. Przykład różnej architektury potokowej dla układów ASIC i FPGA, dla dodawania S = A+B+C+D. Parametry potokowości: N = 2 (ASIC), K = 1(FPGA). FA-Full Adder — sumator pełny, HA-Half Adder — półsumator

większe) z opóźnieniem spowodowanym przez logikę LUT i dodatkowe połączenia wewnątrz CLB (ang. *Configurable Logic Block*). Po drugie dedykowany układ propagacji przeniesienia używa dedykowanych, a przez to bardzo szybkich połączeń pomiędzy wyjściem, a wejściem szeregowego przeniesienia.

Dla układów FPGA i dla układów dodających o dużej szerokości celowe staje się podzielenie układu dodającego na parę części poprzez wstawienie dodatkowych rejestrów potokowych na ścieżce propagacji przeniesienia. W rezultacie otrzymuje się rozwiązanie hybrydowe będące złożeniem architektury potokowej dla układów ASIC i FPGA. To rozwiązanie jednakże komplikuje projektowanie układu i powoduje wstawianie dodatkowych rejestrów na wejściu i wyjściu układu zgodnie z regułą tnij–wstaw (ang. *cut-set*) [11]. Dlatego rozwiązanie hybrydowe nie zostało zastosowane w prezentowanym systemie, jednakże jest rozpatrywane w następnych etapach rozwoju projektu. Rozwiązanie hybrydowe wydaje się być lepsze od metody opóźnionego dodawania [12], dla którego wejście przeniesienia nie propaguje do wyjścia przeniesienia. Z drugiej strony, każdy 4–2 blok dodający ma standardowy blok przeniesienia i dwa wyjścia sumy co powoduje, że wymagane są dwa rejestry potokowe (dodatkowy jeden rejestr).

Podsumowując, dla układów FPGA najlepszym rozwiązaniem jest używanie dedykowanych układów dodających typu RCA oraz strategii potokowej dla której rejestry potokowe są wstawiane co K układów dodających jak to pokazuje rysunek 2.

2. PARAMETRY UKŁADU DODAJĄCEGO

2.1. OPIS PARAMETRÓW WEJŚCIOWYCH

W podrozdziale 1.1 opisano różne metody implementacji konwolucji. W dalszej części tego artykułu rozpatrywany będzie sam blok dodający i jego parametry, które są niezależne od powyższych metod. Poniżej zostaną więc określone parametry bloku dodającego.

Pierwszym parametrem jest liczba wejść do bloku dodającego oraz ich szerokość lub też zakres danych wejściowych. Dla dodawania: $y = a + b$, relacja pomiędzy zakresem danych wejściowych i wyjściowej jest następująca:

$$y_{min} = a_{min} + b_{min} \qquad y_{max} = a_{max} + b_{max} \tag{3a}$$

Dla odejmowania $y = a - b$ stosuje się poniższą zależność:

$$y_{min} = a_{min} - b_{max} \qquad y_{max} = a_{max} - b_{min} \tag{3b}$$

Używanie zakresu danych w porównaniu z używaniem tylko szerokości (liczby bitów) często pociąga za sobą zmniejszenie szerokości użytych układów dodających, jako że niektóre dane często nie używają pełnego zakresu bitowego. Dla przykładu dodanie trzech wejść o zakresie $0 \div 9$ powoduje, że wyjście jest w zakresie $0 \div 27$ (dana

wyjściowa o szerokości 5 bitów). W przypadku gdy rozpatrywana jest tylko szerokość wejść, wyjście ma szerokość 6 bitów. Poza tym rozpatrywanie zakresu danej wejściowej pozwala na łatwe operowanie na argumentach o różnych znakach i pozwala w niektórych przypadkach na pominięcie bitu znaku.

Dodatkowym parametrem każdego wejścia jest przesunięcie bitowe s argumentu wejściowego, kiedy to bity najmniej znaczące (LSB) są zawsze równe zero.

Korelacja pomiędzy wejściami

Dodatkowe zmniejszenie szerokości układu dodającego można niekiedy osiągnąć poprzez uwzględnienie korelacji pomiędzy wejściami. W ogólności dane $x(i-k)$ we wzorze (1) są nieskorelowane. Jednakże korelacja występuje w obrębie mnożenia $h(k) \cdot x(i-k)$, kiedy to występuje dodawanie cząstkowych wyników tego samego mnożenia, np. dodawanie przesuniętych danych $x(i-k)$ dla mnożenia MM. W konsekwencji korelacja jest rozpatrywana niezależnie dla każdego mnożenia, dlatego w tym rozdziale rozpatrywane jest tylko mnożenie.

Korelacja powinna być rozpatrywana niezależnie dla każdej sumy cząstkowej. Dla przykładu dla bloku dodającego $y = a+b+c$, najpierw wykonywana jest suma cząstkowa $y_{ab} = a+b$ i dlatego rozpatrywana powinna być tylko korelacja pomiędzy wejściami a i b. Co więcej korelacja powinna być rozpatrywana od samego początku dla każdej sumy cząstkowej, tj. rozpatrywane są tylko wejścia biorące udział w sumie cząstkowej, a nie historia połączeń drzewa dodającego.

2.1.1. Mnożenie bezmnożne (MM)

Dla architektury MM korelacja pomiędzy wejściami występuje tylko dla operacji odejmowania $y = a-b$, gdzie $a = 2^k \cdot b$. W tym wypadku we wzorze (3b) operacja dodawania powinna być zastąpiona przez operację odejmowania:

$$y_{min} = a_{min} - b_{min} \qquad y_{max} = a_{max} - b_{max} \tag{4}$$

2.1.2. Mnożenie LM

W przypadku mnożenia LM korelacja pomiędzy wejściami jest dużo bardziej skomplikowana. Niech zmienne $I_0, I_1, ..., I_k$ oznaczają wejścia adresowe pamięci LUT dla pojedynczego mnożenia, gdzie I_0 oznacza wejście do najmniej znaczącej LSB (ang. *Least Significant Bit*) pamięci LUT; zmienne $w_0, w_1, ..., w_k$ oznaczają szerokość magistrali adresowej pamięci LUT; $s_0, s_1, ..., s_k$ oznaczają przesunięcie bitowe w lewo każdej pamięci LUT, $s_j = s_0 + \sum_{l=0}^{j-1} w_l$ oraz s oznacza przesunięcie bitowe wyjścia.

Można zauważyć, że wszystkie wejścia za wyjątkiem najbardziej znaczącego MSB (ang. *Most Significant Bit*) operują na dodatnim zakresie binarnym:

$$I_{j\,max} = 2^{Wj} - 1 \qquad I_{j\,min} = 0 \qquad \text{dla} \qquad j = 0 \ldots k-1. \tag{5}$$

Pamięć MSB LUT jest wyjątkiem, dla którego zakres danej wejściowej można wyrazić jako:

$$I_{k\,max} = I_{max} >> s_k \qquad I_{k\,min} = I_{min} >> s_k \tag{6}$$

gdzie: I_{max}, I_{min} — maksymalne i minimalne wartości całej danej wejściowej $x(i-k)$, $>>s$ — przesunięcie bitowe w prawo o s bitów.
Zakres wyjścia pamięci LUT można zdefiniować jako:

$$\begin{aligned} O_{j\,max} &= h \cdot I_{j\,max} \qquad O_{j\,min} = h \cdot I_{j\,min} \qquad \text{dla} \qquad h \geq 0 \\ O_{j\,max} &= h \cdot I_{j\,min} \qquad O_{j\,min} = h \cdot I_{j\,max} \qquad \text{dla} \qquad h < 0 \end{aligned} \tag{7}$$

gdzie: h — współczynnik mnożenia.

Zakres całkowitej danej wyjściowej można otrzymać również używając wzoru (7) poprzez pominięcie indeksu j. Relacja pomiędzy zakresem całkowitej danej wyjściowej $O_{min} \div O_{max}$ i zakresem wyjść pamięci LUT można określić jako:

$$O_{max} \leq \left[\sum_{j=0}^{k} (O_{j\,max} << s_j) \right] >> s \qquad O_{min} \geq \left[\sum_{j=0}^{k} (O_{j\,min} << s_j) \right] >> s \tag{8}$$

Powyższa nierówność staje się równością w wypadku gdy nie obserwuje się korelacji pomiędzy wejściami lub też korelacja nie jest brana pod uwagę.

Algorytm znajdujący zakres danej wyjściowej cząstkowej sumy A (zbiór A zawiera indeksy wejść bloku dodającego, które wchodzą w skład sumy cząstkowej), dla której uwzględniana jest korelacja, jest oparty na określeniu zbioru korelacji C ($C \subseteq A$). Zbiór C zawiera indeks odpowiadający pamięci MSB LUT (indeks k) tylko wtedy gdy wejście k wchodzi w skład sumy cząstkowej A, tj. $k \in A$; w przeciwnym wypadku zbiór C jest pusty (nie obserwuje się korzyści związanej z uwzględnienia korelacji pomiędzy wejściami). Zbiór C jest dalej konstruowany poprzez iteracje, rozpoczynając od indeksu $j = k-1$. Indeks j należy do zbioru korelacji C wtedy gdy wejście j bierze udział w sumie cząstkowej A oraz zbiór C zawiera wejście $j+1$. W konsekwencji zbiór C zawiera kolejne wejścia: $j, j+1, \ldots, k$, gdzie $j-1$ jest indeksem najbardziej znaczącej pamięci LUT, która nie bierze udziału w sumie cząstkowej A. Zakres wejścia dla zbioru C jest obliczany w podobny sposób jak dla MSB LUT we wzorze (6) i jest wyrażony następująco:

$$I_{Cmax} = I_{max} >> s_C \qquad I_{Cmin} = I_{min} >> s_C \tag{8a}$$

gdzie: $s_C = w - \sum_{j \in C} w_j = s_{min(C)}$, $min(C)$ — najmniejszy indeks w zbiorze C.

Zakres wyjścia dla zbioru C jest obliczany podobnie jak dla wzoru (7):

$$\begin{aligned} C_{A\,max} &= h \cdot I_{C\,max} \quad C_{A\,min} = h \cdot I_{C\,min} \quad \text{dla} \quad h \geq 0 \\ C_{A\,max} &= h \cdot I_{C\,min} \quad C_{A\,min} = h \cdot I_{C\,max} \quad \text{dla} \quad h < 0 \end{aligned} \tag{9}$$

Ostatecznie zakres danej wyjściowej sumy cząstkowej A jest obliczany następująco:

$$\begin{aligned} O_{A\,min} &= \left[(C_{A\,min} << s_C) + \sum_{i \in A-C} (O_{i\,min} << s_i) \right] >> s_A \\ O_{A\,max} &= \left[(C_{A\,max} << s_C) + \sum_{i \in A-C} (O_{i\,max} << s_i) \right] >> s_A \end{aligned} \tag{10}$$

gdzie: s_A — przesunięcie bitowe sumy cząstkowej A, $s_A = min(s_i)$ dla $i \in A$.

Warto zwrócić uwagę, że zbiór korelacji C jest pusty wtedy, gdy MSB LUT nie należy do zbioru A. W tym wypadku $C_{A\,min} = 0$, $C_{A\,max} = 0$. Ponadto uwzględnienie korelacji nie daje żadnego efektu dla danych wejściowych wykorzystujących pełny zakres bitowy, np. $0 \div 255$ lub $-128 \div 127$.

Przykład 1

Rozważmy przykład z rysunku 1 dla danej wejściowej o zakresie $-99 \div 99$ (dana wejściowa o szerokości 8 bitów) i współczynniku mnożenia $h = 100$. Z równań 5 i 6, zakres danych wejściowych wynosi $I_0 = 0 \div 15$, $I_1 = -7 \div 6$. Wykorzystując wzór (7) otrzymujemy następujące zakresy wyjść pamięci LUT: $O_{0\,min} = 0$, $O_{0\,max} = 1500$ i $O_{1\,min} = -700$, $O_{1\,max} = 600$. W przypadku, gdy nie uwzględnia się korelacji zakres danej wyjściowej bloku dodającego otrzymany ze wzoru (8) wynosi $O_A = -11\,200 \div 11\,100$. W przeciwnym wypadku wykorzystując wzór (10) otrzymujemy $O_A = -9900 \div 9900$.

Zysk z uwzględnienia korelacji jest bardziej znaczący dla MSB pamięci LUT o mniejszej szerokości. Dla przykładu, dla danej wejściowej o zakresie $-9 \div 9$ (5 bitowa dana) $I_0 = 0 \div 15$ i $I_1 = -1 \div 0$. Zakres danej wyjściowej bez uwzględnienia korelacji jest $O_A = -1600 \div 1500$, w porównaniu z $O_A = -900 \div 900$ kiedy korelacja jest uwzględniona.

Zysk z uwzględnienia korelacji jest również bardziej widoczny w przypadku układu mnożącego z dynamiczną zmianą współczynnika mnożenia w literaturze określanego jako SCM (ang. *Self-Configurable Multiplier*) [13] lub DKCM (ang. *Dynamic Constant Coefficient Multiplier*) [14]. Dla układu DKCM w porównaniu ze standardowym układem LM o stałym współczynniku mnożenia używa się pamięci LUT typu RAM zamiast ROM, a zmiana współczynnika następuje poprzez odpowiednie przekodowanie pamięci LUT. W tym przypadku zamiast współczynnika mnożenia h we wzorze (7) powinno się użyć zakresu współczynnika $h_{min} \div h_{max}$. W konsekwencji wzór (7) powinien być zastąpiony przez następujące równania:

$$O_{j\,max} = Max(h_{max} \cdot I_{j\,max},\ h_{min} \cdot I_{j\,min})$$
$$O_{j\,min} = Min(h_{max} \cdot I_{j\,min},\ h_{min} \cdot I_{j\,max}) \qquad (11)$$

Warto zwrócić uwagę, że dla układu mnożącego DKCM występuje korelacja pomiędzy wejściami nawet w przypadku danej wejściowej wykorzystującej pełny zakres binarny.

Przykład 2

Rozważmy przykład z rysunku 1 dla zakresu wejściowego $-99 \div 99$ i współczynników mnożenia w zakresie $h = -100 \div 100$. W rezultacie otrzymujemy $I_0 = 0 \div 15$, $I_1 = -7 \div 6$. Zakres wyjść pamięci LUT otrzymany ze wzoru (11) jest równy $O_0 = -1500 \div 1500$ oraz $O_1 = -700 \div 700$. Zakres wyjścia bloku dodającego bez korelacji wynosi: $O_A = -12\,700 \div 12\,700$, a przy uwzględnieniu korelacji $O_A = -9000 \div 9900$.

2.1.3. Arytmetyka rozproszona

Dla arytmetyki rozproszonej DA (ang. *Distributed Arithmetic*) korelacja jest rozpatrywana w podobny sposób jak dla układu LM. Jednakże wejścia do pamięci LUT dla układu DA pochodzą od różnych wejść $x_j(i-k)$ (wzory 1 i 2). Dlatego każde wejście do pamięci LUT powinno mieć niezależnie zapis k_j, $C_{j\,min}$, $C_{j\,max}$ (podrozdział 2.1.4) oraz powinno być traktowane jak wejście jednowejściowej pamięci LUT. To jednak komplikuje zapis korelacji pomiędzy wejściami jako, że jedno wejście ma parę zapisów korelacji. Co więcej uwzględnienie korelacji wewnątrz samego bloku dodającego jest dużo trudniejsze, co zwiększa czas potrzebny na określenie zakresu danych we wnętrzu bloku dodającego.

W konsekwencji został przyjęty uproszczony model określający korelację pomiędzy wejściami, dla którego korelacja jest uwzględniana równocześnie dla wszystkich wejść $x(i-k)$ biorących udział w układzie DA. W konsekwencji zapis k_j, $C_{j\,min}$, $C_{j\,max}$ jest obliczany nie dla pojedynczego mnożenia, ale dla sumy mnożeń. To z kolei powoduje, że jeżeli pojedyncze mnożenie cząstkowe jest pominięte (pojedynczy bit DA-LUT dla dowolnego mnożenia nie jest włączony do sumy cząstkowej), to korelacja nie jest uwzględniana dla wszystkich bitów mniej znaczących niż niewłączony bit. Dlatego uproszczona wersja korelacji może spowodować użycie dłuższych układów dodających, jednakże przypadek ten można wyeliminować w większości przypadków poprzez odpowiednie posortowanie indeksów korelacji – bardziej znaczące bity wejściowe powinny mieć wyższy indeks korelacji. Takie posortowanie oraz to, że w układzie dodającym wejścia o równych wagach są z reguły grupowane razem ze sobą podczas sum cząstkowych powoduje, że rzadziej występuje przypadek pominięcia pojedynczego bitu w zbiorze korelacji.

2.1.4. Parametry sygnałów wejściowych

W przyjętym w tym artykule założeniu każde wejście j do bloku dodającego ma następujący zapis:

- przesunięcie bitowe s_j,
- szerokość wejścia w_j,
- zakres wejścia (np. dla LM $O_{j\,min}$, $O_{j\,max}$),
- rodzaj operacji: dodawanie lub odejmowanie.

Parametry korelacji są następujące:

Dla układu MM:

- Indeks mnożenia k. W przypadku gdy dwa lub więcej wejścia mają ten sam indeks k wtedy dla odejmowania powinno się użyć wzór (4) zamiast wzoru (3).

Dla układu LM i DA:

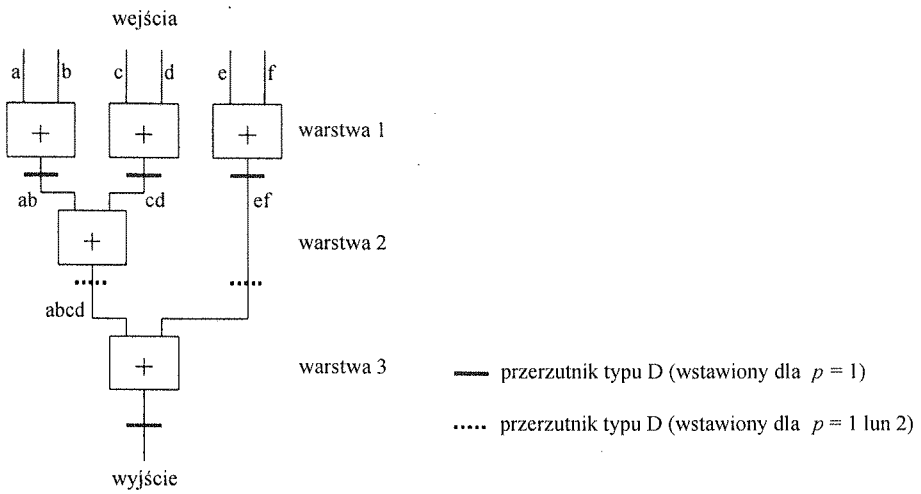
- Indeks korelacji k który wskazuje najbardziej znaczącą pamięć LUT: a) dla mnożenia w przypadku układu LM; b) dla sumy mnożeń w przypadku układu DA.
- Zakres danych po uwzględnieniu korelacji $C_{i\,min}$, $C_{i\,max}$, które są obliczane używając wzoru (9) dla zbioru korelacji $C = \{j, j+1, ..., k\}$. Dane $C_{i\,min}$, $C_{i\,max}$ są wykorzystywane zamiast danych $O_{j\,max}$, $O_{j\,min}$ oraz wzoru (3) w przypadku, gdy wszystkie wejścia o indeksie od j do k biorą udział w sumie cząstkowej A.

Korelacja dla układów LM i DA jest obliczana prawidłowo według powyższego schematu pod warunkiem, że indeksy są w prawidłowym porządku. Dlatego dla układu LM indeksy powinny być przyporządkowane niezależnie dla każdego mnożenia, rosnąco zgodnie z wzrastającą wagą (przesunięciem) pamięci LUT. Podobnie jest dla układu DA, w tym jednak przypadku wszystkie mnożenia biorące udział w układzie DA muszą być traktowane wspólnie (poprzez liniowe złożenie mnożeń i korelacji podobne jak dla układu LM).

2.2. STRUKTURA DRZEWA DODAJĄCEGO

W niniejszym podrozdziale opisane zostaną dodatkowe założenie i reguły budowania drzewa dodającego. Pierwszym założeniem jest użycie drzewa binarnego, dla którego liczba wejść do następnej warstwy drzewa dodającego jest równa połowie liczby wejść w poprzedniej warstwie. Przykład drzewa dla liczby wejść równej 6 jest pokazany na rysunku 3. Założenie drzewa binarnego wydaje się być oczywiste i powoduje, że opóźnienie pomiędzy wejściem a wyjściem jest minimalne. Warto jednak podkreślić, że założenie to może wykluczyć najlepsze rozwiązanie w przypadku, gdy opóźnienie pomiędzy wejściem a wyjściem nie ma znaczenia. Ten przypadek jest jednak rzadki i dlatego założenie użycia drzewa binarnego wydaje się być usprawiedliwione.

Najbardziej skomplikowaną częścią projektu jest łączenie wejść w pary w celu ich dodania. Łączenie to musi odbywać się w taki sposób, aby zminimalizować szerokość układów dodających, lub wyrażając to bardziej ogólnie, aby zminimalizować koszt



Rys. 3. Przykład bloku dodającego dla 6 wejść

bloku dodającego. W konsekwencji określony powinien być koszt komórki sumatora (FA/HA) oraz przerzutników typu D użytych w celu uzyskania architektury potokowej. Alternatywnym, bardziej uniwersalnym rozwiązaniem jest określenie kosztu układu dodającego dla wszystkich możliwych szerokości układu dodającego, np. dla szerokości od 1 do 32 bitów, jako że koszt układu dodającego w układzie FPGA może być różny od liczby użytych bloków FA/HA. Ponadto rozwiązanie to pozwala na powiązanie kosztu układu dodającego nie tylko z zajmowaną powierzchnią układu FPGA, ale również z czasem propagacji. W tym celu koszt układu dodającego powinien wzrastać gwałtownie wraz ze wzrostem szerokości układu dodającego, jako że opóźnienie układu dodającego wzrasta wraz z jego szerokością. W konsekwencji został przyjęty zapis, dla którego koszt układu dodającego jest określony niezależnie dla wszystkich szerokości.

Warto podkreślić, że szerokość układu dodającego jest mniejsza niż szerokość rezultatu dodawania w przypadku, gdy jeden argument dodawania jest przesunięty w stosunku do drugiego. Przypadek ten występuje dla przykładu z rysunku 1, dla którego 4 LSB są bezpośrednio kopiowane na wyjście. Dla odejmowania jednakże nie można bezpośrednio kopiować bitów LSB odjemnika, ponieważ operacja odejmowania wymaga oprócz operacji dodawania dokonania konwersji odjemnika do kodu uzupełnień do dwóch. Konwersja ta wymaga negacji każdego bitu odjemnika oraz dodanie 1 do LSB odjemnika. Operacja dodawania i konwersji do kodu uzupełnień do dwóch w układach FPGA może być z reguły wykonywana poprzez użycie standardowych komórek układu dodającego, dlatego operacja odejmowania jest teoretycznie tak samo kosztowna jak operacja odejmowania. Niemniej w przypadku gdy odjemnik jest przesunięty w prawo w stosunku do odjemnej, bity LSB odjemnika muszą być poddane konwersji do kodu uzupełnień do dwóch nawet jeżeli odejmowanie nie jest wykonywane dla tych bitów, co powoduje że wymagana jest dodatkowa logika, która nie występuje

dla układu dodającego. W konsekwencji dodawanie i odejmowanie muszą być traktowane w inny sposób [1] i dlatego różne reguły optymalizacji powinny być użyte dla dodawania i odejmowania.

W celu zwiększenia przepustowości układu dodającego użyto architektury potokowej. W konsekwencji wprowadzono dodatkowy parametr p, który określa co ile warstw drzewa dodającego należy wstawiać rejestry potokowe. Przykład architektury potokowej jest podany na rysunku 3. Dla $p = 1$ przerzutniki typu D są wstawiane po każdej warstwie drzewa dodającego, dla $p = 2$ przerzutniki są wstawiane po warstwie 2, 4, 6, itd. Warto podkreślić, że w układach FPGA przerzutnik typu D jest stowarzyszony z każdym blokiem logiki (pamięci LUT), dlatego teoretycznie wprowadzenie architektury potokowej nie powoduje zwiększenia kosztu układu dodającego. Jednakże niektóre bity rezultatu dodawania są kopiowane bezpośrednio na wyjście bez żadnej logiki, co ma miejsce w wypadku przesunięcia argumentów dodawania (przypadek opisany w poprzednim akapicie). Ponadto w przypadku nieparzystej liczby sygnałów w danej warstwie jeden z sygnałów musi być bezpośrednio kopiowany do następnej warstwy drzewa dodającego (nie można znaleźć pary) — przypadek ten jest pokazany np. na rysunku 3 (sygnał *ef*). Wymienione przypadki nie wymagają użycia logiki (pamięci LUT) co może powodować, że rejestry potokowe mogą nie być stowarzyszone z pamięciami LUT i dlatego obszar zajmowany przez blok dodający z architekturą potokową będzie zdefiniowany przez liczbę użytych przerzutników typu D, a nie liczbę sumatorów. W konsekwencji dla parametru potokowego $p = 1$ obszar zajmowany przez układ dodający jest określony raczej przez liczbę użytych przerzutników, a dla $p \geq 2$ przez liczbę sumatorów. Jednakże zwiększenie parametru p powoduje zmniejszenie przepustowości układu dodającego, dlatego parametr p powinien być wybrany w drodze kompromisu pomiędzy przepustowością a powierzchnią zajmowaną przez układ dodający.

2.3. PRZYKŁADY FILTRÓW

Rezultaty implementacji przykładowych filtrów i ich podstawowe parametry są podane w tabeli 1.

Tabela 1

Parametry przykładowych filtrów

Filtr	#wejść	zakres wejścia	rozmiar filtru	Koszt [# FA/HA] dla GrA
a	16	0–15	5×5	111
b	11	0–15	5×5	74
c	13	−99–99	8	128
d	39	0–199	41	413
e	85	0–9999	41	1358
f	290	0–9999	41	3730

3. ALGORYTM ZACHŁANNY (GRA)

Algorytm zachłanny GrA (ang. *Greedy Algorithm*) [15] wybiera kolejne najlepsze cząstkowe rozwiązanie przy przyjętej regule wyboru. W ten sposób złożony problem jest rozwiązywany poprzez stopniowe znajdywanie rozwiązań cząstkowych, dzięki czemu złożoność problemu całkowitego jest mniejsza. Wadą tej metody jest to, że wybieranie kolejnych optymalnych rozwiązań lokalnych często nie prowadzi do znalezienia najlepszego globalnego rozwiązania. Algorytm zachłanny jest najszybszym algorytmem z rozważanych w tym artykule i często daje zadawalający rezultat. Najważniejszą częścią algorytmu, która decyduje o jakości otrzymanego rezultatu jest określenie kryteriów wyboru, które określą priorytety, według których wybierane są optymalne lokalne rozwiązania. W opisanym projekcie głównym zadaniem algorytmu GrA jest znalezienie kolejnych optymalnych par, które będą dodawane. Różne kryteria są określone przy wyborze pierwszego i drugiego wejścia do dwuwejściowego układu dodającego. Zostały wybrane następujące zasady wyboru:

1. Wejście pierwsze.

1.1. Wybierz wejście o najmniejszym przesunięciu s_1. Jeżeli dwa lub więcej wejścia mają to samo najmniejsze przesunięcie s_1 rozważ następny punkt (dla tych wejść).

1.2. Wybierz wejście o najmniejszej szerokości w_1.

2. Wejście drugie.

2.1. Wybierz wejście, dla którego waga bitu MSB $m_2 = s_2 + w_2$ jest najmniejsza. Pomiń tą regułę w przypadku, gdy waga bitu MSB m_2 jest większa lub równa wadze bitu MSB wejścia pierwszego $m_1 (m_1 \geq m_2)$. W przypadku, gdy dwa lub więcej wejść ma taki sam m_2 lub $m_1 \geq m_2$ przejdź do następnego punktu.

2.2. Wybierz wejście z najmniejszym przesunięciem s_2. W przypadku, gdy dwa lub więcej wejść ma taki sam s_2 przejdź do następnego punktu.

2.3. Wybierz wejście, które nie generuje przeniesienia (szerokość rezultatu dodawania nie wzrasta). W przypadku gdy dwa lub więcej wejść spełnia powyższy warunek przejdź do następnego punktu.

2.4. Wybierz wejście z największą maksymalną wartością $I_{2\,max}$.

Reguła 1.1 powoduje, że pierwsze wejście jest wybierane w celu sortowania wejść ze względu na ich przesunięcie, dzięki czemu algorytm nie skupia się na znajdowaniu optymalnego lokalnego rozwiązania, a raczej dąży do znajdowania optymalnego globalnego rozwiązania. Jest tak dlatego, że wejścia niesparowane mają większe przesunięcia co powoduje, że w następnych krokach optymalizacji wejścia mają podobne przesunięcia, dzięki czemu otrzymuje się układy dodające o mniejszej szerokości. Reguła 1.2 dąży do znalezienia najlepszego lokalnego rozwiązania poprzez wybranie wejścia o najmniejszej szerokości. Ta reguła również dąży do znalezienia dobrego globalnego rozwiązania, ponieważ szersze wejścia powinny być grupowane z wejściami o większym przesunięciu, które są odłożone na następne iteracje algorytmu dzięki regule 1.1 i 2.2 oraz częściowo regule 2.1.

Wejście drugie jest wybierane, aby najpierw zoptymalizować rezultat lokalnie, a dopiero potem globalnie. Reguła 2.1 znajduje wejście, które generuje najwęższy rezultat dodawania. Reguła 2.2 wybiera wejście o najmniejszym przesunięciu i dlatego dąży do osiągnięcia najlepszego globalnego rozwiązania podobnie jak reguła 1.1. Warto podkreślić, że jeżeli przesunięcia s_1 i s_2 są różne, to $(s_2 - s_1)$ LSB bitów wejścia drugiego jest kopiowanych bezpośrednio na wyjście układu dodającego, co redukuje szerokość użytego układu dodającego. Bezpośrednie kopiowanie wejścia drugiego w pewien sposób usprawiedliwia to, że reguła 2.1 ma wyższy priorytet niż reguła 2.2. Reguła 2.3 wybiera wejście, które nie powoduje przeniesienia, dzięki czemu otrzymuje się węższy rezultat dodawania. W celu znalezienia najlepszego globalnego rozwiązania reguła 2.4 wybiera wejście o największej wartości maksymalnej, która nie generuje przeniesienia (reguła 2.3).

Powyższe reguły algorytmu zachłannego są oparte na dogłębnych badaniach, niemniej autorzy nie mogą zapewnić, że nie mogą być znalezione lepsze reguły optymalizacji. Co więcej, reguły mogą być różne dla różnych parametrów wejściowych, np. dla odejmowania bezpośrednie kopiowanie LSB nie ma miejsca i dlatego podane powinny być inne reguły optymalizacji. Co więcej średni koszt układu dodającego może się różnić dla różnych szerokości układu dodającego np. w celu uwzględnienia relacji czasu propagacji oraz powierzchni zajmowanej przez układ dodający, w rezultacie powinny być podane inne reguły i metody konstruowania układu dodającego.

4. PRZESZUKANIE WYCZERPUJĄCE (ES)

4.1. WPROWADZENIE

Najlepszy rezultat można zawsze otrzymać sprawdzając wszystkie możliwe możliwości. Niestety problem znalezienia optymalnego drzewa układu dodającego jest NP trudny przez co tylko proste drzewa dodające mogą być optymalizowane poprzez użycie metody wyczerpującego przeszukiwania ES (ang. *Exhausted Search*). Na początek rozważmy przykład drzewa dodającego z 5 wejściami, dla którego następujące przypadki muszą być rozpatrzone (przeszukiwanie dotyczy tylko dolnej warstwy dodającej, przykład pokazuje jak wejścia od a do e mogą być łączone w pary):

$$
\begin{array}{lll}
(a+b)+(c+d)+e; & (b+c)+(a+d)+e; & (c+a)+(b+d)+e; \\
(b+c)+(d+e)+a; & (c+d)+(b+e)+a; & (d+b)+(c+e)+a; \\
(c+d)+(e+a)+b; & (d+e)+(c+a)+b; & (e+c)+(b+a)+b; \\
(d+e)+(a+b)+c; & (e+a)+(d+b)+c; & (a+d)+(e+b)+c; \\
(e+a)+(b+c)+d; & (a+b)+(e+c)+d; & (b+e)+(a+c)+d.
\end{array}
$$

W celu znalezienia liczby możliwych kombinacji, które muszą być rozpatrzone przez algorytm ES, zostanie teraz zdefiniowana funkcja $S_1(n)$, która określa liczbę kombinacji na pojedynczej warstwie drzewa dodającego dla określonej liczby wejść n:

$$S_1(n) = \begin{cases} n \cdot (n-2) \cdot (n-4) \cdot \mathrm{K} \cdot 3 \cdot 1 & \text{dla nieparzystych } n \\ S_1(n-1) & \text{dla parzystych } n \end{cases} \tag{12}$$

Całkowita liczba możliwych kombinacji $S(n)$ jest zdefiniowana w sposób iteracyjny i jest wynikiem iloczynu liczby kombinacji dla rozpatrywanej warstwy drzewa dodającego oraz całkowitej liczbie kombinacji na wyższej (bliżej wyjścia) warstwie drzewa dodającego, dla którego liczba wejść jest dzielona przez dwa.

$$S(n) = \mathrm{S}_1(n) \cdot S(\lceil n/2 \rceil) \tag{13}$$

gdzie: $\lceil\ \rceil$ — funkcja zaokrąglania w górę.

Tabela 2

Liczba warstw i liczba kombinacji dla równej liczby wejść do bloku dodającego

n	# warstw	# kombinacji
2	1	1
3	2	3
4	2	3
5	3	45
6	3	45
8	3	315
10	4	42 525
12	4	467 775
14	4	42 567 525
16	4	638 512 875
18	5	1 465 387 048 125

Z tabeli 2 można wysunąć wniosek, że liczba możliwych kombinacji wzrasta gwałtownie wraz ze wzrostem liczby wejść, co powoduje, że metoda ES może być stosowana tylko dla liczby wejść nie większej niż 11 ÷ 16.

4.2. PRZESZUKIWANIE OGRANICZONE (CS)

Liczba możliwych kombinacji wzrasta bardzo gwałtownie z liczbą wejść do bloku dodającego, dlatego w tym podrozdziale zaproponowana jest metoda, która ogranicza liczbę możliwych przeszukiwań CS (ang. *Constrain Search*). Ta metoda na samym początku znajduje koszt bloku dodającego dla każdej warstwy bloku dodającego

z wykorzystaniem metody GrA opisanej w rozdziale 3. W konsekwencji najpierw jest obliczony koszt $C(l)$ bloku dodającego połączonego aż do warstwy l. W dalszym ciągu algorytmu następuje przeszukiwanie możliwych rozwiązań podobnie jak dla metody ES, jednakże przeszukiwanie może być przerwane już na początkowym etapie (warstwy wyższe niż l nie są przeszukiwane), jeżeli koszt częściowo połączonego bloku dodającego $C(l)$ jest większy niż $C_b(l)+t$, gdzie: $C_b(l)$ — koszt $C(l)$ najlepszego rozwiązania, t — pewna liczba progowa. Procedura porównywania kosztów aktualnego $C(l)$ i najlepszego $C_b(l)$ jest wykonywana po całkowitym połączeniu całej warstwy drzewa dodającego.

Technika CS zmniejsza czas potrzebny na znalezienie struktury bloku dodającego ponieważ rozwiązania, które mają mniejsze prawdopodobieństwo uzyskania najlepszego wyniku są wykluczane już na samym początku i wyższe warstwy drzewa dodającego nie muszą być dalej rozważane. Z drugiej strony jest możliwe, że układ dodający ma wyższy koszt dla dolnych warstw drzewa, ale ma mały koszt dla górnych warstw co powoduje, że rozwiązanie to jest pomijane mimo tego, że może to być rozwiązanie optymalne. Dlatego ważną kwestią jest właściwy dobór wartości progowej t. Zwiększając wartość progową t zwiększa się liczbę możliwych rozwiązań, które będą analizowane, ale zmniejsza się prawdopodobieństwo ominięcia najlepszego rozwiązania i odwrotnie.

Tabela 3

Teoretyczna liczba rozpatrywanych rozwiązań dla różnej liczby wejść N oraz różnych algorytmów

N	ES	CS (warstwa 1)	CS (warstwa 2)
6	45	15	45
8	315	105	315
10	42 525	945	14 175
12	467 775	10 395	155 925
14	42 567 525	135 135	14 189 175
16	638 512 875	2 027 025	212 837 625
18	1 465 387 048 125	34 459 425	32 564 156 625

Tabela 3 pokazuje teoretyczną liczbę możliwych rozwiązań dla metody CS zakładając, że przeszukiwanie zostanie zakończone już na warstwie 1 lub na warstwie 2. Można zauważyć, że liczba przeszukiwań uległa zdecydowanej redukcji, jednakże jest ona ciągle nie do zaakceptowania dla liczby wejść N większej niż 16.

4.3. WYNIKI DOŚWIADCZALNE

W tym podrozdziale podane są wyniki doświadczalne dla algorytmów GrA, ES i CS. Tabela 4 pokazuje koszt najlepszego rozwiązania dla różnych technik i różnej

wartości progowej t dla algorytmu CS. Tabela 5 pokazuje złożoność algorytmu, tj. liczbę rozważanych rozwiązań.

Tabela 4

Koszt implementacji (liczba FA/HA) dla różnych przykładów filtrów i różnych technik

Filtr	# wejść	ES	CS ($t = 5$)	CS ($t = 2$)	CS ($t = 0$)	CS ($t = -1$)	GrA
a	16	93	93	93	93	93	111
b	11	72	72	72	73	73	74
c	13	123	126	126	126	126	128

Tabela 5

Liczba rozważanych rozwiązań dla różnych technik i filtrów

Filtr	warstwa 1	warstwa 1, 2	ES	CS ($t = 5$)	CS ($t = 2$)	CS ($t = 0$)	CS ($t = -1$)
a	2 027 025	212 837 625	638 512 875	9 556 259	4 881 543	2 963 651	2 327 927
b	945	14 175	467 775	444 927	278 735	80 051	13 173
c	135 135	14 189 175	42 567 525	3 079 789	915 375	369 357	193 057

Można zauważyć z tabel 4 i 5, że zadowalający wynik można osiągnąć używając tylko algorytmu GrA, lepszy rezultat o około 2–7% można jednak uzyskać używając algorytmu ES. Główną wadą algorytmu ES jest jego długi czas implementacji, dlatego sensowne wydaje się użycie algorytmu CS dla liczby wejść nie większej niż 16. Dla wartości progowej $t = -1$, rozpatrywane jest tylko rozwiązanie częściowe, które jest lepsze niż najlepsze znalezione do tej pory. To powoduje, że metoda CS ($t = -1$) jest podobna do algorytmu GrA, dla którego rozwiązanie cząstkowe nie jest rozpatrywane dla pojedynczego układu dodającego, ale dla całej warstwy drzewa dodającego. Ponadto dla metody CS możliwe jest anulowanie wyboru, jeżeli koszt na wyższych warstwach jest większy od poprzedniego najlepszego rozwiązania.

Zwiększając wartość progową t liczba rozważanych rozwiązań wzrasta. Dla $t = 0$, rozpatrywane są rozwiązania nie tylko lepsze, ale również takie same (na tym etapie) jak rozwiązanie najlepsze. Dalsze zwiększanie wartości progowej t powoduje wzrost liczby rozpatrywanych rozwiązań i tylko nieznacznie poprawia osiągnięty rezultat.

Warto zwrócić uwagę, że algorytm GrA daje gorsze rezultaty dla filtrów, dla których używa się odejmowania (filtry A i C mają ujemne współczynniki). Powodem tego jest to, że algorytm ten traktuje odejmowanie i dodawanie w podobny sposób.

5. PODSUMOWANIE

W artykule tym podano gruntownej analizie blok dodający, który może być częścią filtrów typu FIR, jednakże istnieje szereg innych rozwiązań, gdzie napotyka się na podobny problem związany z implementacją tego typu obliczeń. Rozpatrzono

również skomplikowane parametry wejściowe takie jak zakres danych wejściowych (a nie tylko szerokość jak to ma zwykle miejsce), a nawet korelacje pomiędzy wejściami. W konsekwencji tego znalezienie optymalnej struktury drzewa dodającego jest skomplikowanym zadaniem, jako że wejścia mogą mieć różne szerokości, przesunięcia (wagę bitową) i mogą być różnie powiązane ze sobą. Dlatego zostały podane różne algorytmy poszukujące najlepszego rozwiązania. Algorytm zachłanny GrA jest algorytmem najprostszym i najszybszym, jednakże daje najgorszy rezultat z rozważanych algorytmów. Z drugiej strony optymalne rozwiązanie można uzyskać sprawdzając wszystkie możliwe rozwiązania, tj. używając algorytmu wyczerpującego przeszukiwania ES. Jednakże liczba możliwych rozwiązań wzrasta gwałtownie wraz z liczbą wejść i dla liczby wejść $N > 10 \div 14$ algorytm ten staje się nieakceptowalny. Dlatego została zaproponowana metoda ograniczonego przeszukiwania (CS), która jest podobna do metody ES, jednakże metoda CS odrzuca już na początku rozwiązania, dla których koszt częściowo połączonego bloku dodającego jest wysoki. Metoda ta może pominąć najlepsze rozwiązanie jednakże liczba przeszukiwań uległa gwałtownej redukcji. Wadą tej metody jest to, że liczba możliwych kombinacji wzrasta gwałtownie wraz z liczbą wejść N do bloku dodającego, dlatego metoda ta może być stosowana tylko dla $N \leq 16$. Jest to zatem nieznaczna poprawa w porównaniu z metodą ES.

Podsumowując, algorytm zachłanny jest zalecany dla liczby wejść większej niż 16 pomimo tego, że lepszy wynik można otrzymać uzyskując bardziej zaawansowane algorytmy. Dalszym etapem prac będzie użycie innych algorytmów optymalizacji takich jak np. Simulated Annealing czy też algorytmu genetycznego.

Otrzymane wyniki są częścią złożonego systemu AuToCon, którego zadaniem jest automatyczna generacji procesora konwolucji 2D w układach FPGA dla dowolnie podanych parametrów procesora, takich jak np. rozmiar konwolucji, wartości współczynników konwolucji czy też zakresu danej wejściowej.

6. BIBLIOGRAFIA

1. K. Wiatr, E. Jamro: *Constant Coefficient Multiplication in FPGA Structures.* Proceedings of the 26th Euromicro Conference, Maastricht, The Netherlands, Sep. 5–7 2000, Vol. I, pp. 252–259.
2. H. Garner: *Number Systems and Arithmetic.* Advances in Computing, 1965, vol. 6, pp. 131–194.
3. K. Chapman: *Fast Integer Multiplier fit in FPGA's.* EDN 1993 Design Idea Winner, EDN May 12th 1994.
4. A. R. Omondi: *Computer Arithmetic Systems. Algorithms Architecture and Implementations.* Prentice Hall, UK, 1994.
5. T. T. Do, C. Reuter, P. Pirsch: *Alternative approaches implementing high-performance FIR filters on lookup table-based FPGAs: A comparison.* SPIE Conference on Configurable Computing and Applications. Boston, Massachusetts, 2–3 Nov. 1998, pp. 248–254.
6. C. S. Burrus: *Digital filter structure described by arithmetic.* IEEE transaction on Circuits and systems, 1977, pp. 674–680.
7. R. A. Hawley, et. al.: *Design Techniques for Silicon Compiler Implementation of High-Speed FIR Digital Filters.* IEEE Journal of Solid-State Circuits, vol. 31, no 5, May 1996, pp. 656–667.
8. Xilinx Co.: *The Programmable Logic.* Data Book 1999.

9. Altera Co.: *Apex 20K Programmable Logic Device Family, Data Sheet.* ver. 2.05, Nov. 1999.
10. S. Xing, W. W. H. Yu: *FPGA Adders: Performance Evaluation and Optimal Design.* IEEE Design & Test of Computers, Jan.–Mar. 1998, pp. 24–29.
11. P. Pirsch: *Architectures for Digital Signal Processing.* Chichester UK, Wiley 1998.
12. Z. Luo, M. Martonosi: *Using Delayed Addition Techniqus to Accelerate Integer and Floating-Point Calculation in Configurable Hardware.* SPIE Conference on Configurable Computing: Technology and Applications, Boston, Massachusetts, Nov. 1998, Vol. 3526, pp. 202–211.
13. M. Wojko, H. ElGindy: *Configuration Sequencing with Self Configurable Multipliers.* 13th International Parallel Processing Symposium and 10th Symposium on Parallel and Distributed Processing, San Juan, Puerto Rico, USA, April 1999, pp. 643–651.
14. K. Wiatr, E. Jamro: *Implementation of Multipliers in FPGA Structures.* Proc. of the IEEE Intern. Symposium on Quality Electronic Design, San Jose, California, 26–28 March 2001, pp. 415–420.
15. T. H. Cormen, C. E. Leiserson, R. L. Rivest: *Intoduction to Algorithms.* Massachusetts Institute of Technology, 1994.

K. WIATR, E. JAMRO

EFFICIENT FPGA IMPLEMENTATION OF ADDERS AS A PART OF CONVOLVERS

Summary

Addition is a fundamental operation for the constant coefficient convolutions (FIR filters). In FPGAs, addition should be carried out employing ripple-carry adders rather than carry-save adders as it is the case for ASIC designs. Therefore different adder optimisation techniques are required as a result Exhaustive Search and Greedy Algorithm have been implemented. Different convolver architectures and consequently different input parameters, e.g. input width, correlation between different inputs, are described.

Keywords: specialised processors, multiprocessors systems architecture, programmable logic device, distributed arithmetic, real-time systems

KWARTALNIK ELEKTRONIKI I TELEKOMUNIKACJI, 2002, 48, z. 3–4, ss. 591–606

Optymalizacja drzewa dodającego implementowanego w układach FPGA z wykorzystaniem programowania genetycznego i *Simulated Annealing*

KAZIMIERZ WIATR, ERNEST JAMRO

Katedra Elektroniki, Akademia Górniczo-Hutnicza
Al. Mickiewicza 30, 30-059 Kraków

Otrzymano 2001.09.21
Autoryzowano 2002.02.02

Operacja dodawania jest podstawową operacją w realizacji wielu algorytmów przetwarzania danych (np. podczas obliczania operacji konwolucji — filtracji typu FIR o stałych współczynnikach). W układach FPGA (ang. *Field Programmable Gate Arrays*) operacja dodawania powinna być implementowana z wykorzystaniem układu dodającego z przeniesieniem skrośnym RCA (ang. *Ripple Carry Adder*), w porównaniu z układami ASIC, dla których optymalną architekturą jest układ dodający z przechowaniem przeniesienia CSA (ang. *Carry Save Adder*). W konsekwencji dla układów FPGA powinno się użyć innych metod optymalizacji drzewa dodającego niż dla układów ASIC. W artykule tym zostały przedstawione dwa takie algorytmy: programowanie genetyczne GP (ang. *Genetic Programming*) i *Simulated Annealing* SA (symulowane wyżarzanie). Algorytmy te zostały porównane z uprzednio użytymi metodami przeszukiwania wyczerpującego ES (ang. *Exhaustive Search*) oraz algorytmu zachłannego GrA (ang. *Greedy Algorithm*). W rezultacie wyniki otrzymane przez SA są lepsze niż dla GP oraz SA daje około $10 \div 20\%$ oszczędności w porównaniu z GrA. Dlatego optymalnym rozwiązaniem jest użycie algorytmu ES dla liczby wejść do bloku dodającego $N \leq 8$ oraz SA dla $N > 8$. W przypadku gdy decydującym czynnikiem jest czas znalezienia optymalnego drzewa zalecany jest algorytm GrA.

Słowa kluczowe: procesory specjalizowane, architektury systemów procesorowych, programowalne układy logiczne, arytmetyka rozproszona, systemy czasu rzeczywistego

1. WSTĘP

Układy FPGA zyskują coraz większą popularność i stają się ważną alternatywą dla układów mikroprocesorowych i układów dedykowanych ASIC (ang. *Application Specific Integrated Circuit*). Dlatego obecnie coraz częściej implementuje się w układach

FPGA cyfrowe przetwarzanie sygnałów DSP (ang. *Digital Signal Processing*), dla którego dodawanie jest podstawową operacją. Przykładem może być implementacja procesora konwolucji (filtrów FIR) z użyciem układu bezmnożnego MM (ang. *Multiplerless Multiplication*) [1], dla którego mnożenie wykonywane jest w postaci dodawań odpowiednio przesuniętego argumentu wejściowego. W przypadku równoległego układu procesora konwolucji, dla którego wynik operacji konwolucji otrzymywany jest co jeden takt zegara, otrzymywany jest skomplikowany blok dodający o N wejściach, które mogą być różnej szerokości (zapisane na różnej liczbie bitów) oraz różnie przesunięte względem siebie. Oprócz architektury MM istnieje szereg innych architektur takich jak mnożenie z użyciem pamięci LUT (*Look-Up Table*) [1] czy też arytmetyki rozproszonej DA (ang. *Distributed Arithmetic*) [2]. Dlatego w celu lepszej optymalizacji drzewa dodającego wydzielono niezależny od architektury układu blok dodający, który jest określony następującymi parametrami [3]:

- N — liczba wejść,
- w_i — szerokość wejścia i w bitach,
- V_{i_min}, V_{i_max} — zakres danej wejściowej i. W rozważaniach rozpatrywany jest raczej zakres danej wejściowej niż jej szerokość dzięki czemu można otrzymać dodatkową redukcję szerokości użytego układu dodającego,
- s_i — przesunięcie bitowe w lewo (waga) wejścia i,
- rodzaj operacji — dodawanie albo odejmowanie,
- korelacja pomiędzy wejściami. Często dodanie zakresów dwóch skorelowanych ze sobą wejść daje w wyniku zakres wyniku, który jest większy niż faktyczny zakres sumy wejść. Dlatego, aby uwzględnić tę zależność, stosuje się dodatkowo korelacje pomiędzy wejściami, która uwzględnia relacje pomiędzy zakresami wejść.

W układach ASIC klasyczny problem propagacji przeniesienia jest rozwiązywany w różnoraki sposób, np. z użyciem sumatora z antycypacją przeniesienia (ang. *Carry Look Ahead*) czy też z sumatora multipleksowanego sterowanego przeniesieniem (ang. *Carry Select Adder*) [4], w wyniku czego redukuje się czas propagacji przeniesienia kosztem większej powierzchni zajmowanej przez układ dodający. Innym podejściem jest całkowite wyeliminowanie problemu przeniesienia poprzez zastosowanie układów dodających przechowujących przeniesienie CSA (ang. *Carry-Save Adder*). W konsekwencji w układach ASIC w procesorach konwolucji stosuje się głównie układy CSA [5].

Układy FPGA posiadają dedykowany układ propagacji przeniesienia skrośnego RCA (ang. *Ripple Carry Adder*) [6, 7], który jest tak szybki i efektywny, że konwencjonalne metody przyspieszania propagacji przeniesienia są bezużyteczne dla układów dodających o szerokości 16 bitów i mają marginalne znaczenie dla układów dodających o szerokości 32 bity. Co więcej, dedykowany układ propagacji przeniesienia jest implementowany poza standardową logiką pamięci LUT czyli nie zajmuje on dodatkowej logiki. Podsumowując, używanie układów dodających typu RCA w układach FPGA jest najlepszym rozwiązaniem ze względu na czas propagacji i zajmowaną powierzchnię [8].

W konsekwencji optymalizacja bloku dodającego w układach FPGA przebiega w diametralnie inny sposób niż dla układów ASIC, dlatego przedmiotem niniejszego

artykułu są nowe metody konstrukcji optymalnego drzewa dodającego składającego się tylko z układów dodających typu RCA. Optymalizacja w głównej mierze polega na odpowiednim łączeniu wejść w pary w celu ich dodania (w konsekwencji otrzymuje się drzewo binarne) w taki sposób, aby użyte układy dodające miały jak najmniejszą szerokość lub też wyrażając to bardziej ogólnie, aby koszt bloku dodającego był jak najmniejszy. Koszt bloku dodającego nie musi być prostą sumą liczby użytych sumatorów bitowych, ale może on uwzględniać czas propagacji w układzie dodającym, np. poprzez zwiększanie relatywnego kosztu sumatorów w przypadku użycia długich układów dodających, itd.

Niniejsza praca jest kontynuacją prac opublikowanych w [3], gdzie przedstawiono szczegółowo parametry bloku dodającego oraz ich powiązanie z architekturą procesora konwolucji. Ponadto wprowadzono dodatkowy parametr potokowości p, dzięki któremu poprawia się przepustowość układu dodającego. Niestety dla $p = 1$ (rejestry potokowe są wstawiane po każdej operacji dodawania) obszar zajmowany przez blok dodający jest określony przez liczbę użytych rejestrów, dlatego koszt rejestrów musi być również uwzględniony podczas konstruowania zoptymalizowanego drzewa dodającego. W pracy [3] przedstawiono również podstawowe metody optymalizacji układu dodającego przy użyciu algorytmu zachłannego GrA (ang. *Greedy Algorithm*) oraz przeszukiwania wyczerpującego ES (ang. *Exhaustive Search*). Najlepszy rezultat można otrzymać używając algorytmu ES, dla którego przeszukiwane są wszystkie możliwe kombinacje drzewa dodającego. Niestety liczba możliwych rozwiązań gwałtownie wzrasta wraz ze wzrostem liczby wejść N. Dla przykładu dla $N = 8$ i $N = 16$ liczba możliwych kombinacji wynosi odpowiednio 315 oraz 638 512 875. W konsekwencji algorytm ES jest zalecany tylko dla liczby wejść nie większej niż około $N \leq 12$. Dlatego wprowadzono dodatkowy algorytm przeszukiwania ograniczonego CS (ang. *Constrain Search*), który jest podobny do algorytmu ES, jednakże z góry odrzucane są rozwiązania, które mają wysoki koszt już na dolnych warstwach drzewa dodającego (warstwach bliższych wejścia), czyli mają małe prawdopodobieństwo uzyskania najlepszego wyniku. Wadą tej metody jest to, że liczba kombinacji wzrasta gwałtownie wraz z liczbą wejść N i dlatego metoda ta może być stosowana tylko dla liczby wejść nie większej niż $N \leq 16$. Alternatywnym algorytmem przedstawionym w pracy [3] jest algorytm zachłanny GrA, który jednak daje rezultat o $10 \div 20\%$ gorszy od algorytmu ES. Co więcej, algorytm GrA wymaga zmiany reguł optymalizacji w przypadku uzależnienia kosztu układu dodającego od jego szerokości (uwzględnienie kosztu układu dodającego od czasu propagacji) czy też uwzględnienia dodatkowego kosztu rejestrów potokowych, co utrudnia konstruowanie tego algorytmu, tym bardziej że relacje nie są stałe i różnią się dla różnych układów FPGA czy też nawet zajętości układu FPGA.

W konsekwencji w niniejszym artykule omówiono inne techniki optymalizacji drzewa dodającego takie jak *Simulated Annealing SA* (symulowane wyżarzanie) oraz programowanie genetyczne GP (ang. *Genetic Programming*). Algorytmy te są rozwiązaniem pośrednim pomiędzy algorytmami ES i GrA ponieważ mają dłuższy czas obliczeniowy niż GrA, ale z reguły dają lepszy rezultat. W porównaniu z algorytmem ES, liczba iteracji algorytmów GP i SA jest znacznie mniejsza i co więcej można ustalić

wybraną liczbę iteracji. Zwiększenie liczby iteracji powoduje wydłużenie czasu obliczenia, ale otrzymywany jest lepszy wynik końcowy i na odwrót. Warto zauważyć, że zalecane jest zwiększenie liczby iteracji dla większych układów dodających.

2. SIMULATED ANNEALING (SA)

2.1. WPROWADZENIE

Działanie algorytmu SA [9, 10] jest analogiczne do tego, co dzieje się podczas kontrolowanego (powolnego) schładzania metalu, którego cząsteczki podczas schładzania układają się w taki sposób, aby uformować strukturę krystaliczną, która charakteryzuje się wysoką gęstością i niską energią. W algorytmie SA wartość, którą chcemy optymalizować jest analogiczna do energii w systemie termodynamicznym. W wysokiej temperaturze, kiedy to cząstki mają dużą ruchliwość występują duże zmiany w strukturze i wysoce prawdopodobne jest zaakceptowanie układu o wyższej energii. W niskiej temperaturze, występują tylko małe zmiany dotyczące tylko sąsiednich cząstek i prawdopodobieństwo zaakceptowania układu o wyższej energii jest małe.

Algorytm SA, użyty w celu optymalizacji drzewa dodającego, składa się z następujących kroków:

Funkcji kosztu, która zwraca koszt danego drzewa dodającego.

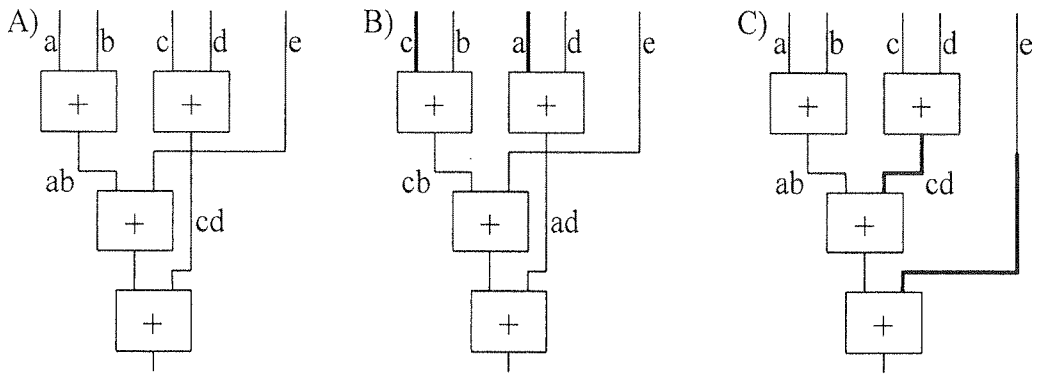
Schematu schładzania, który określa jak szybko spada temperatura podczas kolejnych iteracji algorytmu. W optymalizowanym bloku dodającym temperatura początkowa T_1 równa się kosztowi dwubitowego układu dodającego, a temperatura końcowa T_S wynosi $^1/_4$ kosztu układu sumatora bitowego. W każdym kroku iteracji temperatura T_i jest zmniejszana według poniższego wzoru:

$$T_{i+1} = \eta \cdot T_i \tag{1}$$

gdzie: $\eta = \left(\frac{T_1}{T_S}\right)^{1/S}$, S — liczba iteracji.

Generowanie nowej struktury drzewa dodającego polega na losowym wybraniu dwóch dwu-wejściowych układów dodających (lub dwóch wejść do bloku dodającego) na tej samej warstwie drzewa, tj. losowym wybraniu dowolnego układu dodającego, a następnie losowym wybraniu innego układu dodającego ograniczając wybór tylko do tej samej warstwy co układ pierwszy. Następnie zamienia się wejścia wybranych układów dodających, jak to pokazuje przykład z rysunku 1.

Zmiany w układzie są w pewien sposób ograniczone od temperatury T_i. W standardowym algorytmie SA, znanym również jako układ Boltzmann'a, funkcją generującą, która określa modyfikacje wektora wejściowego jest funkcja Gaussowska rozkładu prawdopodobieństwa. W przypadku drzewa dodającego, liczba możliwych rozwiązań jest dyskretna i dlatego funkcja Gaussa jest bezużyteczna. Dlatego losowo wybiera się



Rys. 1. Przykład możliwych zmian w układzie podczas jednego kroku iteracji: A) układ początkowy, B), C) układ A po przykładowej modyfikacji

dwa układy dodające i oblicza się lokalną funkcję akceptacji LAF (ang. *Local Acceptance Function*), która określa czy dana zmiana jest akceptowana lokalnie. Funkcja LAF różni się tym od globalnej funkcji akceptacji, która jest opisana w następnym akapicie, że pod uwagę brane są tylko dwa układy dodające losowo wybrane podczas modyfikacji drzewa dodającego. Jeżeli zmiana nie jest zaakceptowana lokalnie wtedy następuje jej odrzucenie i wygenerowanie następnej modyfikacji, itd. Licznik iteracji i temperatura nie ulegają zmianie podczas tych czynności.

Funkcja akceptacji. Po określeniu nowego drzewa dodającego i obliczeniu jego kosztu C algorytm SA decyduje, czy zaakceptować nowy układ czy też przywrócić poprzedni układ na podstawie funkcji akceptacji $h(\quad)$, która jest funkcją Boltzmann'a dystrybucji prawdopodobieństwa:

$$h(\Delta C, T) = \frac{1}{1+\exp(\Delta C/T)} \tag{2}$$

gdzie: $\Delta C = C_{i+1} - C_i$ — różnica pomiędzy kosztem nowego i poprzedniego drzewa dodającego.

Nowy układ jest akceptowany z prawdopodobieństwem równym wartości otrzymanej przez obliczenie funkcji akceptacji.

2.2. WYNIKI DOŚWIADCZALNE

Wynik doświadczalny dla różnych przykładów filtrów jest podany w tabeli 1, na podstawie której można zauważyć, że dla SA i liczby iteracji 1000, koszt filtru *a* wynosi 103 czyli tyle samo jak dla algorytmu ES (filtry od $a \div f$, są dla lepszego porównania takie same jak przykłady w [3]). Dla filtru *c* koszt równy 123 jest uzyskany dla liczby iteracji równej 1 000 000, dla algorytmu CS koszt 126 był osiągnięty dla liczby iteracji sięgającej 3 000 000, co pozwala na wysunięcie wniosku, że algorytm SA góruje nad

algorytmem CS i często zwraca porównywalny wynik jak ES. Jednakże dla algorytmu ES (i częściowo CS) zmiana w układzie jest dobrze określona i z reguły występuje na najwyższej warstwie drzewa, dlatego w celu obliczenia kosztu układu dodającego (najbardziej czasochłonna część algorytmu) można posłużyć się częściowo kosztem obliczonym w poprzednich etapach. W przypadku algorytmu CS, najczęściej obliczany jest koszt tylko najniższych warstw, a warstwy najwyższe nie są rozważane. W konsekwencji czasochłonność pojedynczego kroku algorytmu ES i CS jest mniejsza niż w przypadku algorytmu SA, gdzie zmiana występuje przypadkowo na dowolnej warstwie co powoduje, że obliczany jest każdorazowo koszt całego układu dodającego. Jednakże mniejsza czasochłonność pojedynczej iteracji algorytmu CS i ES nie równoważy tak znacząco mniejszej liczby potrzebnych iteracji dla algorytmu SA.

Tabela 1

Koszt (wyrażony w liczbie użytych sumatorów bitowych) najlepszego układu dodającego dla algorytmu GrA, ES i różnej liczby iteracji algorytmu SA

Filtr	GrA	SA 1k	SA 30k	SA 1M	ES
a	111	93±0	93±0	93±0	93
c	128	126.9±0.3	125±1.4	123±0	123
d	413	394±3	385±1	382±1	–
d (bez LAF)	413	398±4	382±1	380±1	–
e	1358	1346±10	1299±3	1292±4	–
e (bez LAF)	1358	1341±9	1293±4	1283±4	–
f	3730	3702±20	3338±14	3245±6	–
f (bez LAF)	3730	3706±13	3419±13	3296±8	–

Układ końcowy (otrzymany w ostatniej iteracji algorytmu SA) bardzo często nie jest najlepszym algorytmem i dlatego najlepszy otrzymany układ jest za każdym razem zachowywany jako wynik ostateczny. W rezultacie zwiększa to tylko nieznacznie czasochłonność algorytmu, ale pozwala na otrzymanie dużo lepszego wyniku.

Tabela 1 pokazuje wyniki dla dwóch różnych opcji: z i bez lokalnej funkcji akceptacji LAF. Obliczenie kosztu dwóch układów dodających przed i po modyfikacji tylko nieznacznie zwiększa czasochłonność algorytmu, a funkcja LAF odrzuca rozwiązania, które mają małe prawdopodobieństwo wygenerowania dobrego globalnego wyniku. Z drugiej strony LAF ogranicza w pewien sposób przeszukiwanie i powoduje, że część dobrych rozwiązań jest odrzucona, co ma szczególnie miejsce dla małych układów dodających oraz dużej liczby iteracji. Dla przykładu dla filtru *d* można otrzymać lepszy wynik z pominięciem funkcji LAF. Jednakże funkcja LAF zdecydowanie poprawia otrzymany wynik dla bardziej skomplikowanych układów i mniejszej liczby iteracji, dlatego ostatecznie została ona zastosowana.

3. PROGRAMOWANIE GENETYCZNE (GP)

Programowanie genetyczne GP (ang. *Genetic Programming*) [11, 12] jest techniką optymalizacji opartą na koncepcji naturalnej selekcji i procesie ewolucji. Ważniejsze etapy GP obejmują: schemat kodowania, ocena przystosowania w populacji, wybór rodzicieli, operacje krzyżowania i mutacji. Etapy te zostaną omówione poniżej.

3.1. SCHEMAT KODOWANIA

Schemat kodowania określa w jaki sposób jest reprezentowany rozważany problem i w naszym przypadku określa jak jest reprezentowane drzewo dodające. W przypadku algorytmów genetycznych struktura problemu powinna być zapisana w postaci ciągu zer i jedynek lub też liczb całkowitych, a wszelkie operacje powinny być wykonywane bezpośrednio na tej reprezentacji. W przypadku GP wszystkie operacje są wykonywane z uwzględnieniem rzeczywistej struktury, którą opisuje dana reprezentacja. W rozważanym przypadku drzewo dodające jest zapisane w postaci dwóch wektorów liczb naturalnych. Każdy układ dodający ma przyporządkowany jeden wpis do każdego z tych wektorów. Wpis ten określa indeks wejścia do rozważanego układu dodającego. Dla przykładu, rodzic 0 na rysunku 2 jest reprezentowany przez następujące dwa wektory:

rozważany układ dodający	*24, 23, 22, 21, 20, 19, 18, 17, 16, 15, 14, 13*
wektor 0	*22, 19, 18, 13, 14, 16, 0, 3, 2, 4, 9, 7*
wektor 1	*23, 20, 21, 17, 15, 11, 6, 8, 1, 5, 12, 10*

Wstępnie można odnieść wrażenie, że strukturę drzewa dodającego można opisać określając tylko kolejność w dolnej warstwie drzewa dodającego, a wyższe warstwy drzewa można określić łącząc sąsiednie wejścia na tej samej warstwie. Niestety jest to tylko szczególny przypadek, kiedy liczba wejść N do bloku dodającego jest potęgą liczby 2. W przeciwnym wypadku pojawia się sygnał *samotny*, który nie może zostać sparowany i dlatego musi być przeprowadzony przez rozważaną warstwę bez wykonania operacji dodawania. W przytoczonym przypadku rodzica 0 na rysunku 2 sygnał numer 11 jest samotny w warstwie pierwszej, sygnał numer 18 jest samotny w warstwie drugiej. W konsekwencji sygnały samotne komplikują strukturę drzewa dodającego i powodują, że kodowana musi być każda warstwa drzewa dodającego.

3.2. OCENA PRZYSTOSOWANIA W POPULACJI I OPERACJA SELEKCJI

Ocena przystosowania w populacji jest oparta na obliczeniu kosztu rozważanego drzewa dodającego.

Selekcja polega na wybraniu osobników, których właściwości będą kopiowane do następnej populacji. Operacja selekcji określa, na podstawie oceny przystosowania

w populacji, które osobniki mają przetrwać; czym mniejszy koszt bloku dodającego tym większe prawdopodobieństwo przetrwania. W rozważanym przypadku zastosowano algorytm modGA [12] jako, że przewyższa on standardowy algorytm genetyczny [12]. Dla algorytmu modGA, w każdej populacji wybieranych jest $(p–r)$ osobników, które mają być skopiowane do następnej populacji w niezmienionej formie z prawdopodobieństwem proporcjonalnym do przeskalowanej funkcji przystosowania f'_i, która jest otrzymywana przez liniowe skalowanie kosztu rozważanego bloku dodającego f_i:

$$f'_i = a \cdot f_i + b \tag{3}$$

Współczynniki a i b są obliczane niezależnie dla każdej populacji tak aby spełniać poniższy układ równań:

$$a \cdot f_{min} + b = 1 \tag{4}$$

$$\sum_{i=1}^{p} (a \cdot f_i + b) = p - r \tag{5}$$

gdzie: f_{min} — drzewo dodające o najmniejszym koszcie, p — rozmiar populacji, r — liczba osobników przeznaczonych do zagłady $r < p/2$.

Równanie 4 zachowuje najlepszego osobnika z prawdopodobieństwem równym 1. Równanie 5 powoduje, że średnio $(p–r)$ osobników jest zachowywanych podczas jednego obrotu ruletki; podczas jednego obrotu ruletki każdy osobnik jest wybierany do następnej populacji tylko raz z prawdopodobieństwem f'_i. W rozważanym przypadku, ruletka obraca się, aż zostanie wybranych $(p–r)$ osobników, tak że średnio obraca się ona tylko jeden raz.

Osobniki przeznaczone do zagłady, których jest r, są zastępowane przez nowe osobniki, które otrzymuje się w drodze operacji krzyżowania i mutacji:

$$r = c + m \tag{6}$$

gdzie: c — liczba nowych osobników otrzymanych przez operacje krzyżowania, m — liczba osobników otrzymanych przez operacje mutacji nienadpisującej (zob. operacja mutacji).

W rozważanym przypadku $p = 12$, $r = 5$, $c = 4$, $m = 1$.

3.3. OPERACJA KRZYŻOWANIA

Operacja krzyżowania jest wykonywana na losowo wybranych parach rodziców. Struktura drzewa dodającego wydaje się być bardzo podobna do powszechnie używanego grafu drzewa używanego do planowania i podziału [12], czy też znajdowania drzewa optymalnych działań [13]. Jednakże w przeciwieństwie do tych przykładów, dla drzewa

dodającego struktura drzewa jest bardzo ograniczona. Konsekwentnie operacja krzyżowania generuje tylko poprawne drzewo dodające — nie jest potrzebna dodatkowa procedura naprawcza, często konieczna dla algorytmów genetycznych.

Zostały zaimplementowane dwie różne opcje operacji krzyżowania. Pierwsza z nich (opcja A) dąży do skopiowania jak największej porcji informacji od rodziców na podstawie rzeczywistej struktury rodziców a nie tylko numerów indeksów, a następnie używa algorytmu zachłannego (uproszczona wersja algorytmu opisanego w [3]) w celu sparowania układów dodających, które nie można połączyć na podstawie rzeczywistej struktury rodziców. Druga operacja krzyżowania (opcja B) polega na kopiowaniu przez potomka całej struktury jednego z rodziców, a następnie implementacji zmian podobnie jąk dla algorytmu SA jednakże nie w sposób losowy a na podstawie drugiego rodzica. W tej opcji rozważa się indeksy układu dodającego a nie faktyczne drzewo połączeń. Poniżej znajduje się bardziej szczegółowy opis procedur krzyżowania.

Opcja A

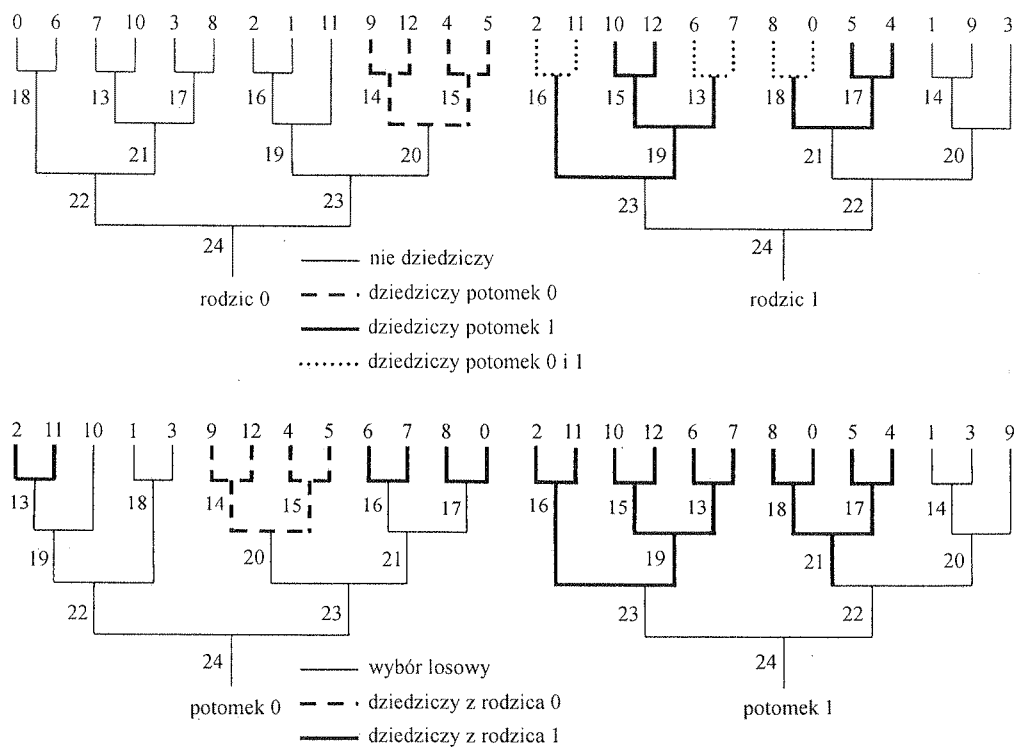
W opcji A potomek dziedziczy jedną, lub też wszystkie oprócz jednej, gałęzi drzewa dodającego od pierwszego rodzica. Dla przykładu podanego na rysunku 2, potomek 0 dziedziczy od rodzica 0 strukturę następujących układów dodających: 20, 14, 15, 9, 12, 4, 5. Następnie potomek dziedziczy jak najwięcej od drugiego rodzica. W przedstawionym przykładzie, dziedziczy strukturę układów dodających 13, 2, 11; 16, 6, 7 i 17, 8, 0 od rodzica 1. Niestety struktura potomka bardzo często nie może być całkowicie określona na podstawie takiego algorytmu dziedziczenia jako, że niektóre połączenia zrealizowane na podstawie struktury rodzica 0 wykluczają się wzajemnie z połączeniami występującymi w strukturze rodzica 1. I tak dla przytoczonego przykładu połączenia 14, 9, 12 odziedziczone od rodzica 0 wykluczają połączenia 14, 1, 9 realizowane dla rodzica 1 jako, że wejście 9 jest już połączone.

Warto podkreślić, że operacja krzyżowania rozważa tylko numery indeksu na dolnej (wejściowej) warstwie drzewa dodającego (w przytoczonym przykładzie indeksy $0 \div 12$). Dla wyższych warstw indeksy nie są rozpatrywane, a algorytm krzyżowania uwzględnia tylko faktyczne połączenia począwszy od warstw dolnych a skończywszy na warstwach górnych. W konsekwencji parowanie (przy dziedziczeniu od rodzica drugiego) układów dodających na wyższych warstwach jest realizowane tylko wtedy gdy zrealizowane zostało dla danego układu dodającego parowanie na warstwie niższej. W konsekwencji, pojedyncze połączenie, które nie mogło być zrealizowane na warstwie niższej propaguje się na warstwę wyższą, uniemożliwiając połączenie układu dodającego na warstwie wyższej. To powoduje, że struktura drzewa dodającego jest bardzo rzadko określona całkowicie dla warstw górnych. Z drugiej strony jednak potomek dziedziczy tylko rzeczywiste połączenia, które istniały dla jednego z rodziców.

Takie podejście powoduje, że efektywność algorytmu krzyżowania zależy mocno od punktu krzyżowania, tj. jak dużo struktury dodającej jest kopiowane od pierwszego rodzica. W konsekwencji dodatkowy parametr krzyżowania jest dołączony do schematu kodowania (genu), co pozwala na optymalizowanie tego parametru razem ze strukturą drzewa dodającego w procesie programowania genetycznego. Ten dodatkowy parametr

określa, od której warstwy drzewa dodającego począwszy, losowo wybrana gałąź drzewa jest kopiowana od pierwszego rodzica do potomka. Dla przykładu na rysunku 2, potomek 0 dziedziczy następującą strukturę od rodzica 0: 20, 14, 15, 9, 12, 4, 5, tj. od gałęzi numer 20 począwszy, gałąź ta została losowo wybrana z warstwy 2 (określonej przez ten dodatkowy parametr). Ponadto ten dodatkowy parametr określa czy ma być dziedziczona tylko jedna gałąź czy też wszystkie gałęzie oprócz jednej. Dla przytoczonego przypadku, potomek 0 dziedziczy tylko jedną gałąź od rodzica 0, natomiast potomek 1 dziedziczy wszystkie gałęzie oprócz jednej od rodzica 1, tj. za wyjątkiem układu dodającego numer 20, 14, 1, 9, 3.

W wyniku implementacji przytoczonego algorytmu dla starych populacji otrzymanych pod koniec optymalizacji występował tylko parametr krzyżowania, który określał, że należy kopiować tylko jedną gałąź począwszy od przedostatniej warstwy drzewa dodającego, tj. kopiuj połowę drzewa dodającego od pierwszego rodzica. W konsekwencji parametr krzyżowania został usunięty z procesu optymalizacji i punkt krzyżowania został na stałe określony jako — kopiuj połowę układu dodającego od pierwszego z rodziców.



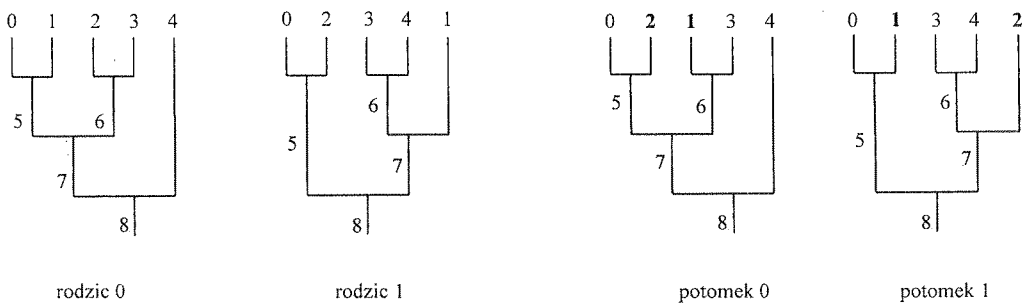
Rys. 2. Przykład operacji krzyżowania dla opcji A

Opcja B

W opcji B krzyżowanie jest bardzo podobne jak dla algorytmu SA, który został opisany w rozdziale 2. Na początku cała struktura potomka jest kopiowana od pierwszego rodzica, a krzyżowanie polega na dokonywaniu zmian w strukturze potomka na podstawie struktury połączeń drugiego rodzica. Krzyżowanie to różni się jednak od generowania nowej struktury w algorytmie SA tym, że dla SA drzewo dodające jest modyfikowane w sposób przypadkowy. Przykład operacji krzyżowania jest podany na rysunku 3.

Operacja modyfikowania potomka składa się z trzech kroków:

1. Losowym wybraniu wspólnego sygnału (w rozpatrywanym przypadku sygnał 0).
2. Określeniu sygnałów podlegających zamianie. Sygnały te są sparowane ze wspólnym sygnałem określonym w punkcie 1 (w rozpatrywanym przypadku sygnał 1 dla rodzica 0 i sygnał 2 dla rodzica 1). W przypadku gdy sygnał wspólny jest sygnałem samotnym (jest bez pary) sygnałem podlegającym zamianie jest sygnał wspólny.
3. Zamianie sygnałów znalezionych w poprzednim punkcie.



Rys. 3. Przykład operacji krzyżowania dla opcji B

Warto podkreślić, że sygnał wspólny może być wybrany losowo na dowolnej warstwie (za wyjątkiem warstwy najwyższej, co prowadzi do przypadku trywialnego). Indeksy sygnałów na wyższych warstwach dla różnych rodziców mogą nie odpowiadać sobie nawzajem w sensie rzeczywistej struktury drzewa dodającego. Dla przykładu danego na rysunku 3 sygnał 5 dla rodzica 0 ($S_5 = S_0 + S_1$) jest różny od sygnału 5 dla rodzica 1 ($S_5 = S_0 + S_2$). To oznacza, że zamiana dokonywana na wyższych warstwach bardzo często nie rozważa rzeczywistej struktury drzewa dodającego jako, że indeksy są przyporządkowywane w mniej lub bardziej przypadkowy sposób. Dlatego aby poprawić pracę algorytmu, indeksy na wyższych warstwach drzewa są przyporządkowywane (sortowane) według narastającego indeksu wejścia (a dokładniej mniejszego indeksu z dwóch wejść). Dla przykładu dla rodzica 1 na rysunku 3 układ dodający numer 5 ma najmniejszy indeks w warstwie 1 ponieważ wejście 0 ma najniższy indeks w warstwie 0. Sortowanie indeksów poprawia korelacje pomiędzy rodzicami, niemniej indeksy sygnałów na wyższych warstwach często reprezentują różne sygnały dla różnych rodziców. To oznacza, że zamiana sygnałów jest często wykonywana w sposób przypadkowy,

szczególnie jeżeli struktura rodziców jest zdecydowanie różna. Warto zauważyć, że korelacja pomiędzy rodzicami spada wraz ze wzrostem liczny wejść do bloku dodającego, dlatego dla dużych bloków dodających ten sposób krzyżowania nie jest zalecany (tab. 3).

Krzyżowanie przez pojedyncze modyfikowanie jest z reguły niewystarczające, dlatego zazwyczaj dokonuje się $1 \div 3$ zamian w celu otrzymania potomka.

Główną ideą algorytmu modGA jest to, że algorytm unika posiadania kopii takich samych osobników w jednej populacji, co może się jednak zdarzyć przypadkowo ale jest mało prawdopodobne [12]. Jednakże wyniki doświadczalne dowiodły, że zarówno opcja A jak i B mogą wytworzyć identyczne potomstwo do swoich rodziców, szczególnie w przypadku gdy rodzice są bardzo podobni do siebie. To powoduje, że w jednej populacji znajdują się często wielokrotne kopie tego samego osobnika, co pogarsza działanie algorytmu modGA. Dlatego aby poprawić działanie algorytmu, w przypadku gdy potomek jest kopią jednego z rodziców (lub też różni się tylko nieznacznie) wprowadza się dodatkową operacje mutacji, która jest wykonywana na tym potomku. W konsekwencji dodatkowa operacja mutacji zapobiega powstawaniu dokładnej kopii rodzica podczas operacji krzyżowania i powstrzymuje najsilniejszego osobnika od dominacji w populacji. Podobnie jest zresztą w naturze, gdzie osobniki unikają krzyżowania się ze swoimi krewnymi w celu zapobieżenia powstawania podobnych genetycznie potomków. Podobne rozwiązanie zostało zaproponowane przez Maudlin'a [14], gdzie częstość mutacji jest uzależniona od zróżnicowania genetycznego populacji. Wadą algorytmu zaproponowanego przez Maudlin'a jest to, że potrzebne są dodatkowe obliczenia potrzebne do określenia stopnia zróżnicowania populacji. W podejściu zaproponowanym w tej pracy dzięki skojarzeniu operacji określania stopnia zróżnicowania z operacją krzyżowania wymagana jest tylko niewielka modyfikacja algorytmu krzyżowania oraz niewiele dodatkowych obliczeń.

3.4. OPERACJA MUTACJI

Operacja krzyżowania może wykorzystać tylko potencjał aktualnych osobników, dlatego wprowadza się dodatkowo operację mutacji, która w sposób przypadkowy generuje nowe chromosomy. W opisywanym algorytmie mutacja jest dokonywana w podobny sposób jak generacja nowego drzewa w algorytmie SA, tj. poprzez zamianę dwóch losowo wybranych sygnałów na tej samej warstwie drzewa dodającego.

Zostały zaimplementowane dwie różne opcje operacji mutacji:

- mutacja nienadpisująca rodzica (MNN),
- mutacja nadpisująca rodzica (MN).

MNN jest stowarzyszona z algorytmem modGA. W każdej populacji generowanych jest r nowych osobników, z których c osobników powstaje w operacji krzyżowania, a m osobników podczas MNN. Dlatego losowo wybrany osobnik, który przetrwał operacje selekcji, jest kopiowany a następnie na tej kopii jest przeprowadzona operacja MNN.

Operacja MN jest przeprowadzona w sposób standardowy, czyli każdy osobnik który przetrwał operacje selekcji jest mutowany z prawdopodobieństwem p_m.

W niniejszej pracy zostały zaimplementowane dwie różne opcje mutacji w celu uzyskania poprawnego rozwoju populacji. W przypadku gdy użyta jest tylko MN, wysokie prawdopodobieństwo mutacji p_m uniemożliwia rozwój najlepszych osobników jako, że prawdopodobieństwo wygenerowania potomka o podobnym przystosowaniu w populacji jest bardzo małe — mniejsze niż prawdopodobieństwo mutacji. W konsekwencji najlepsze rozwiązanie jest często wygenerowane bardziej w sposób przypadkowy niż na podstawie właściwości algorytmu genetycznego i przystosowanie osobników w końcowych populacjach jest często dalekie od przystosowania najlepszego osobnika w historii. Z drugiej strony, niskie prawdopodobieństwo mutacji powoduje, że mutacja ma małoznaczący wpływ na wynik algorytmu genetycznego, co powoduje otrzymanie gorszych wyników końcowych.

Użycie tylko mutacji nienadpisującej rodzica (MNN) powoduje, że najlepsze osobniki są zawsze kopiowane do następnej populacji bez żadnej zmiany (najlepszy osobnik jest wybierany podczas operacji selekcji z prawdopodobieństwem równym 1), przez co populacja jest zdominowana przez najlepsze osobniki, które mogą być w lokalnym minimum. W konsekwencji najlepszy wynik można osiągnąć poprzez kombinacje tych dwóch mutacji. Wyniki doświadczalne przedstawione w tabeli 2 potwierdzają to założenie (za wyjątkiem małych bloków dodających z dużą liczbą iteracji).

Tabela 2

Wyniki doświadczalne dla różnych filtrów i liczby iteracji oraz różnych opcji mutacji: tylko MNN, kombinacja MNN i MN oraz tylko MN; wynik dla krzyżowania — opcja A

filtr	$p_m = 0$, $m = 1$	$p_m = 0.2\%$, $m = 1$	$p_m = 10\%$, $m = 0$
d) iter = 6 k	393 ± 2	392 ± 5	391 ± 2
d) iter = 200 k	392 ± 4	388 ± 5	381 ± 1
e) iter ± 6 k	1302 ± 3	1303 ± 2	1317 ± 4
e) iter = 200 k	1297 ± 3	1292 ± 2	1297 ± 2
f) iter = 6 k	3580 ± 7	3580 ± 6	3616 ± 8
f) iter = 200 k	3454 ± 5	3455 ± 4	3508 ± 16

3.5. WYNIKI DOŚWIADCZALNE

Tabela 3 pokazuje wyniki doświadczalne dla różnych algorytmów rozważanych w tym artykule. Liczba iteracji dla algorytmów GP i SA jest tak dobrana, aby czasy obliczeń były podobne. Można zauważyć, że algorytm SA z reguły daje lepszy rezultat niż algorytm GP oraz, że krzyżowanie z opcją A jest z reguły lepsze od krzyżowania z opcją B, szczególnie w przypadku bardziej skomplikowanych bloków dodających. Dla

przykładu dla opcji B i filtru f (najbardziej skomplikowany filtr) wynik algorytmu GP jest tak zły, że wynik początkowy otrzymany przez GrA jest najlepszym rozwiązaniem dla liczby iteracji 6000.

Tabela 3

Wyniki doświadczalne dla różnych opcji GP i SA

Filtr — algorytm	liczba iteracji (GP/SA)		
	200/1 k	6 k/30 k	200 k/1 M
d) Opcja A	399±3	392±5	388±5
d) Opcja B	412±2	392±4	387±5
d) SA	394±3	385±1	382±1
e) Opcja A	1341±6	1303±2	1292±2
e) Opcja B	1358±0	1357±1	1297±4
e) S.A.	1346±10	1299±3	1292±4
f) Opcja A	3713±7	3580±6	3455±4
f) Opcja B	3730±0	3730±0	3645±26
f) S.A.	3702±20	3338±14	3245±6

4. WNIOSKI

Niniejszy artykuł przedstawia analizę różnych metod optymalizacji bloku dodającego będącego np. częścią filtru FIR. Praca [3] zawiera sformułowanie problemu i przedstawia różne architektury układu procesora konwolucji oraz ich wpływ na parametry bloku dodającego, a w szczególności korelacje pomiędzy wejściami. Ponadto w [3] zaimplementowano trzy podstawowe metody optymalizacji: przeszukiwanie wyczerpujące (ES), przeszukiwanie ograniczone (CS) oraz algorytm zachłanny (GrA). Algorytm ES i CS może być użyty tylko dla małych bloków jako, że ilość wymaganych obliczeń szybko wzrasta wraz ze wzrostem ilości wejść do bloku dodającego. Prostszym obliczeniowo algorytmem jest algorytm GrA, który jednak charakteryzuje się gorszymi wynikami. Dlatego w niniejszym artykule zaimplementowano dwie inne metody optymalizacji drzewa dodającego: Simmulated Annealing (SA) i programowanie genetyczne (GP).

Dla małej liczby wejść do bloku dodającego, algorytm SA z reguły znajduje bardzo szybko najlepsze rozwiązanie przy znacznie mniejszej liczbie iteracji niż dla algorytmu ES. Niemniej dla liczby wejść do bloku dodającego $N \leq 8$, liczba iteracji algorytmu ES jest nie większa niż 315, dzięki czemu koszt znalezienia najlepszego rozwiązania jest bardzo niski, co powoduje że zalecany jest algorytm ES. Dla $N \geq 9$, ES przeszukuje co najmniej 42 525 kombinacji dlatego zalecana jest implementacja algorytmu SA.

Programowanie genetyczne jest inną metodą konkurencyjną w stosunku do SA. Struktura bloku dodającego jest jednak silnie ograniczona (zdefiniowana) i dlatego zostało wybrane programowanie genetyczne, w którym w operacji krzyżowania otrzymywany jest zawsze potomek spełniający te ograniczenia. Niemniej silne ograniczenia bloku dodającego powodują, że operacja krzyżowania jest silne zdeterminowana przez te ograniczenia, tj. struktura potomka jest określona głównie na podstawie jednego z rodziców, a pozostała część potomka często jest zdefiniowana bardziej aby spełnić założenia drzewa dodającego niż skopiować strukturę drugiego rodzica. Dlatego zostały zaimplementowane dwie różne metody krzyżowania, aby przeprowadzić krzyżowanie bardziej optymalnie. Niemniej przy tym samym czasie pracy algorytmu wynik jest lepszy dla algorytmu SA. Podobny wniosek został wysnuty również przez McMahon et. al. [15] dla problemu planowania zadań, co pokazuje, że dla niektórych zadań algorytm SA góruje nad GP.

Optymalizacja bloku dodającego jest częścią systemu automatycznej generacji procesora konwolucji w układach FPGA — systemu AuToCon (ang. *Automated Tool for Convolution processor design*). System AuToCon [1, 3] generuje zoptymalizowany kod w języku VHDL dla różnych parametrów procesora konwolucji, np. rozmiar konwolucji, wartości współczynników, etc. Warto zwrócić jednak uwagę, że otrzymane wyniki mogą posłużyć do optymalizacji drzewa dodającego dla innych operacji implementowanych w układach FPGA, np. dla mnożenia macierzy, sieci neuronowych itd.

5. BIBLIOGRAFIA

1. K. Wiatr, E. Jamro: *Constant Coefficient Multiplication in FPGA Structures.* Proceedings of the 26th Euromicro Conference, Maastricht, The Netherlands, Sep. 5–7 2000, Vol. I, pp. 252–259.
2. T. T. Do, C. Reuter, P. Pirsch: *Alternative Approaches Implementing High-Performance FIR Filters on Look-up-table Based FPGAs: A Comparison.* SPIE Conference on Configurable Computing and Applications, Boston, Massachusetts, 2–3 Nov. 1998, pp. 248–254.
3. K. Wiatr, E. Jamro: *Implementacja układów dodających wchodzących w skład konwolwera w układach programowalnych FPGA.* Kwartalnik Elektronika i Telekomunikacja PAN, Warszawa 2001, 48, z.3 –4, ss. 571–589.
4. A. R. Omondi: *Computer Arithmetic Systems. Algorithms Architecture and Implementations.* Prentice Hall, UK, 1994.
5. R. A. Hawley et, al.: *Design Techniques for Silicon Compiler Implementation of High-Speed FIR Digital Filters.* IEEE Journal of Solid-State Circuits, May 1996, vol. 31, no 5, pp. 656–667.
6. Xilinx Co. *The Programmable Logic* Data Book 1999.
7. Altera Co. *Apex 20K Programmable Logic Device Family, Data Sheet.* ver. 2.05, Nov. 1999.
8. S. Xing, W. W. H. Yu: *FPGA Adders: Performance Evaluation and Optimal Design.* IEEE Design & Test of Computers, Jan.–Mar. 1998, pp. 24–29.
9. E. H. Aarts, J. Korst: *Simulated Annealing and Boltzman Machines.* Chichester, UK, Wiley 1989.
10. S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi: *Optimisation by Simulated Annealing.* Science, 220 (4598), May 1983: pp. 671–680.
11. J. R. Koza: *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* Cambridge, MA: The MIT Press, 1992.
12. Z. Michalewicz: *Genetic Algorithm + Data Structure = Evolutions Programs.* Berlin, Spinger-Verlag 1992.

13. T. Aytekin, E. E. Korkmaz, H. A. Guvernir: *An Application of Genetic Programming to the 4-Op Problem using Map-Trees,* in Xin Yao *Progress in Evolutionary Computation*, Selected Papers on AI'93 nad AI'94 Workshops on Evolutionary Computation. Berlin, Springer, 1995, pp. 28–40.
14. M. L. Maudlin: *Maintaining Diversity in Genetic Search.* AAAI Proc. National Conference on Artificial Inteligence, 1984, pp. 247–250.
15. G. McMahon, D. Hadinoto: *Comparison of Heuristic Search Algorithms for Single Machine Scheduling Problems*, in Xin Yao *Progress in Evolutionary Computation*, Selected Papers on AI'93 nad AI'94 Workshops on Evolutionary Computation. Berlin, Springer, 1995, pp. 293–304.

K. WIATR, E. JAMRO

OPTIMISATION OF AN ADDER TREE IMPLEMENTED IN FPGAS EMPLOYING GENETIC PROGRAMMING AND SIMULATED ANNEALING

Summary

Addition is a very basic operation employed in numerous processes, e.g. constant coefficient FIR filters. In Field Programmable Gate Arrays (FPGAs), an addition should be carried out in the standard way employing ripple-carry adders, rather than carry-save adders as it is usually the case for ASICs. Consequently different adders optimisation techniques should be used in order to reduce area occupied by the adder tree. In this paper implementation of two different optimisation techniques: Genetic Programming (GP) and Simulated Annealing (SA) are described. The implementation results of these techniques are compared to the previously published results for the Exhaustive Search (ES) and Greedy Algorithm (GrA). As a result, the SA usually outperforms the GP, and the SA gives about $10 \div 20\%$ area reduction in comparison to the GrA. In conclusion, for the number of inputs to an adder tree $N \leq 8$, the ES is the recommended algorithm as the number of possible combinations is usually acceptable, otherwise the SA should be employed. In the case when the time of finding the optimal adder tree is a critical factor, the GrA is recommended.

Keywords: specialised processors, multiprocessors systems architecture, programmable logic device, distributed arithmetic, real-time systems.

KWARTALNIK ELEKTRONIKI I TELEKOMUNIKACJI

INFORMACJE DLA AUTORÓW

Redakcja przyjmuje do publikowania prace oryginalne, przeglądowe i monograficzne wchodzące w zakres szeroko pojętej elektroniki. Ponieważ KWARTALNIK ELEKTRONIKI I TELEKOMUNIKACJI jest czasopismem Komitetu Elektroniki i Telekomunikacji Polskiej Akademii Nauk, w związku z tym na jego łamach znajdują się prace naukowe dotyczące podstaw teoretycznych i zastosowań z zakresu elektroniki, telekomunikacji, mikroelektroniki, optoelektroniki, radiotechniki i elektroniki medycznej.
Artykuły powinno charakteryzować oryginalne ujęcie zagadnienia, własna klasyfikacja, krytyczna ocena (teorii lub metod), omówienie aktualnego stanu, lub postępu danej gałęzi techniki oraz omówienie perspektyw rozwojowych.
Artykuły publikowane w innych czasopismach nie mogą być kierowane do druku w Kwartalniku Elektroniki i Telekomunikacji w drugiej kolejności zgłoszenia.
Objętość artykułu nie powinna przekraczać 30 stron po około 1800 znaków na stronie, w tym rysunki i tabele.

Wymagania podstawowe.
Artykuły należy nadsyłać na wyraźnym, jednostronnym, czarno-białym wydruku komputerowym. Wydruk w formacie A4 powinien mieć znormalizowaną liczbę wierszy i znaków w wierszu (30 wierszy po 60 znaków w wierszu), w dwóch egzemplarzach, w języku polskim lub angielskim wybranym przez autora. Do wydruku powinna być dołączona dyskietka z elektronicznym tekstem artykułu. Preferowane edytory to WORD 6 lub 8.
Układ artykułu (w wersji podstawowej) musi być następujący:
- Tytuł.
- Autor (imię i nazwisko autora/ów).
- Miejsce pracy (nazwa instytucji, miejscowość, adres, + ew. adres elektroniczny (e-mail).
- Zwięzłe streszczenie powinno być w języku takim, w jakim jest pisany artykuł (wraz ze słowami kluczowymi).
- Tekst podstawowy powinien mieć następujący układ:
  1. WPROWADZENIE
  2. np. TEORIA
  3. np. WYNIKI NUMERYCZNE
  3.1. .....
  3.2. .....
  4. .....
  5. .....
  6. PODSUMOWANIE
  7. ew. PODZIĘKOWANIE
  8. BIBLIOGRAFIA
- Układ streszczenia w dodatkowej wersji językowej powinien być następujący;
  TYTUŁ (w języku angielskim – o ile artykuł pisany jest w języku polskim i na odwrót),
  AUTOR (inicjał imienia i nazwisko).

Obszerne do 3600 znaków streszczenie (wraz z słowami kluczowymi) w języku:
a. angielskim, gdy artykuł pisany jest w języku polskim,
b. polskim, gdy artykuł pisany jest w języku angielskim.

Streszczenie to powinno pozwolić czytelnikowi na uzyskanie istotnych informacji zawartych w pracy. Z tego względu w streszczeniu tym mogą być cytowane numery istotnych wzorów, rysunków i tabel zawartych w podstawowej wersji językowej.
- Wszystkie strony muszą mieć numerację ciągłą.

Sposób pisania tekstu.
Tekst powinien być pisany bez używania wyróżnień, a w szczególności nie dopuszcza się spacjowania, podkreślania i pisania tekstu dużymi literami z wyjątkiem wyrazów, które umownie pisze się dużymi literami

(np. FORTRAN). Proponowane wyróżnienia Autor może zaznaczyć w maszynopisie zwykłym ołówkiem za pomocą przyjętych znaków adiustacyjnych np. podkreślenie linią przerywaną oznacza spacjowanie (rozstrzelenie), podkreślenie linią ciągłą – pogrubienie, podkreślenie wężykiem – kursywa.
Tekst powinien być napisany z podwójnym odstępem między wierszami, tytuły i podtytuły małymi literami. Marginesy z każdej strony powinny mieć około 35 mm. Wielkość czcionki wydruku powinna być zbliżona co najmniej do wielkości czcionki maszyny do pisania (minimum 12 punktów). Przy podziale pracy na rozdziały i podrozdziały cyfrowe ich oznaczenia nie powinny być większe niż III stopnia (np. 4.1.1.).

Sposób pisania tabel.
Tabele powinny być pisane na oddzielnych stronach. Tytuły rubryk pionowych i poziomych powinny być napisane małymi literami z podwójnym odstępem między wierszami. Przypisy (notki) dotyczące tabel należy pisać bezpośrednio pod tabelami. Tabele należy numerować kolejno liczbami arabskimi, u góry każdej tabeli podać tytuł dwujęzyczny. W pierwszej kolejności w podstawowej wersji językowe, a później w dodatkowej wersji językowej. Tabele umieścić na końcu maszynopisu. Przyjmowane są tabele algorytmów i programy na wydrukach komputerowych. W tym przypadku zachowany jest ich oryginalny układ. Tabele powinny być cytowane w tekście.

Sposób pisania wzorów matematycznych.
Rozmieszczenie znaków, cyfr, liter i odstępów powinno być zbliżone do rozmieszczenia elementów druku. Wskaźniki i wykładniki potęg powinny być napisane wyraźnie i być prawidłowo obniżone lub podwyższone w stosunku do linii wiersza podstawowego. Znaki nad literami i cyframi, całkami i in. symbolami (strzałki, linie, kropki, daszki) powinny być umieszczone dokładnie nad tymi elementami, do których się odnoszą. Numery wzorów cyframi arabskimi powinny być kolejne i umieszczone w nawiasach okrągłych z prawej strony. Nazwy jednostek, symbole literowe i graficzne powinny być zgodne w wytycznymi IEE (Intemational Electronical Commision) oraz ISO (Intemational Organization of Standarization).

Powołania.
Powołania na publikacje powinny być umieszczone na ostatnich stronach tekstu pod tytułem „Bibliografia", opatrzone numeracją kolejną bez nawiasów. Numeracja ta powinna być zgodna z odnośnikami w tekście artykułu. Przykłady opisu publikacji:

- periodycznej — 1. F. Valdoni: A new milimetre wave satelitę. E.T.T. 1990, vol. 2, no 5, pp. 141–148
- nieperiodyczne — 2. K. Andersen: A resource allocation framework. XVI Intemational Symposium, Stockholm (Sweden), may 1991, paper A 2.4
- książki — 3. Y.P. Tvidis: Operation and modeling ofthe MOS transistors. New York, McGraw-Hill, 1987, p. 553

Materiały ilustracyjne.
Rysunki powinny być wykonane wyraźnie, na papierze gładkim lub milimetrowym w formacie nie mniejszym niż 9×12 cm. Mogą być także w postaci wydruku komputerowego (preferowany edytor Corel Draw). Fotografie lub diapozytywy przyjmowane są raczej czarno-białe w formacie nie przekraczającym 10×15 cm. Na marginesie każdego rysunku i na odwrocie fotografii powinno być napisane ołówkiem imię i nazwisko Autora oraz skrót tytułu artykułu, do którego są przeznaczone oraz numer rysunku. Spis podpisów pod rysunki i fotografie powinien być umieszczony na oddzielnej stronie. Podpisy pod rysunkami (fotografiami) powinny być dwujęzyczne: w pierwszej kolejności w podstawowej wersji językowej, a później w dodatkowej wersji językowej. Rysunki powinny być cytowane w tekście.

Uwagi dodatkowe.
Na odrębnej stronie powinny być podane następujące informacje:
- adres do korespondencji z kodem pocztowym (domowy lub do miejsca pracy),
- telefon domowy i/lub do miejsca pracy,
- adres e-mailowy (jeśli autor posiada).

Autorowi przysługuje bezpłatnie 20 odbitek artykułu. Dodatkowe egzemplarze odbitek, lub cały zeszyt Autor może zamówić u wydawcy na własny koszt.

Autora obowiązuje korekta autorska, którą powinien wykonać w ciągu 3 dni od daty otrzymania tekstu z Redakcji oraz zwrócić osobiście, lub listownie pod adres Redakcji. Korekta powinna być naniesiona na przekazanych Autorowi szpaltach na marginesach ew. na osobnym arkuszu w przypadku uzupełnień tekstu większych niż dwa wiersze. W przypadku nie zwrócenia korekty w terminie, korektę przeprowadza Redakcja Techniczna Wydawcy.

Redakcja prosi Autorów o powiadomienie ją o zmianie miejsca pracy i adresu prywatnego.

INFORMATION FOR AUTHORS OF K.E.T

The editorial stuff will accept for publishing only original monographic and survey papers concerning widely understood electronics. Because of the fact that KWATRALNIK ELEKTRONIKI I TELEKOMUNIKACJI is a journal of the Committee for Electronics and Telecommunications of Polish Academy of Science, it presents scientific works concerning theoretical bases and applications from the field of electronics, telecommunications, microelectronics, optoelectronics, radioelectronics and medical electronics.
Articles should be characterised by original depiction of a problem, its own classification, critical opinion (concerning theories or methods), discussion of an actual state or a progress of a given branch of a technique and discussion of development perspectives.
An article published in other magazines can not be submitted for publishing in K.E.T.
The size of an article can not exceed 30 pages, 1800 character each, including figures and tables.

Basic requirements
The article should be submitted to the editorial stuff as a one side, clear, black and white computer printout in two copies. The article should be prepared in English or Polish. Floppy disc with an electronic version of the article should be enclosed. Preferred wordprocessors: WORD 6 or 8.
Layout of the article.
- Title.
- Author (first name and surname of author/authors).
- Workplace (institution, address and e-mail).
- Concise summary in a language article is prepared in (with keywords).
- Main text with following layout:
  - Introduction
  - Theory (if applicable)
  - Numerical results (if applicable)
  - Paragraph 1
  - Paragraph 2
  - .....
  - .....
  - Conclusions
  - Acknowledgements (if applicable),
  - References
- Summary in additional language:
  - Title (in Polish, if article was prepared in English and vice versa)
  - Author (firs name initials and surname)
  - Extensive summary, hawever not exceeching 3600 characters (along with keywords) in Polish, if artide wos prepared in English and vice versa). The summary should be prepared in a way allowing a reader to obtaoin essential information contained in the artide. For that reason in the summary author can place numbers of essential formulas, figures and tables from the article.

Pages should have continues numbering.

Main text
Main text can not contain formatting such as spacing, underlining, words written in capital letters (except words that are commonly written in capital letters). Author can mark suggested formatting with pencil on the margin of the article using commonly accepted adjusting marks.
Text should be written with double line spacing with 35 mro left and right margin. Titles and subtitles should be written with small letters. Titles and subtitles should be numbered using no more than 3 levels (i.e. 4.1.1.).

Tables
Tables with their titles should be placed on separate page at the end of the article. Titles of rows and columns

should be written in small letters with double line spacing. Annotations concerning tables should be placed directly below the table. Tables should be numbered with Arabic numbers on the top of each table. Table can consist algorithm and program listings. In such cases original layout of the table will be preserved. Table should be cited in the text.

Mathematical formulas

Characters, numbers, letters and spacing of the formula should be adequate to layout of main text. Indexes should be properly lowered or raised above the basic line and clearly written. Special characters such as lines, arrows, dots should be placed exactly over symbols which they are attributed to. Formulas should be numbered with Arabic numbers place in brackets on the right side of the page. Units of measure, letter and graphic symbols should be printed according to requirements of IEE (International Electronical Commission) and ISO (International Organisation of Standardisation).

References

References should be placed at the and of the main text with the subtitle „References". References should be numbered (without brackets) adequately to references placed in the text. Examples of periodical [1], non-periodical [2] and book [3] references:

1. F. Valdoni: A new millimetre wave satelite. E.T.T. 1990, vol.2, no 5, pp. 141–148
2. K. Anderson: A recource allocation framework. XVI International Symposium, Stockholm (Sweden), May 1991. paper A 2.4
3. Y.P. Tvidis: Operation and modelling of the MOS transistors. New York, McGraw-Hill, 1987, p. 553

Figures

Figures should be clearly drawn on plain or millimetre paper in the format not smaller than 9×12 cm. Figures can be also printed (preferred editor – CorelDRAW). Photos or diapositives will be accepted in black and white format not grater than 10×15 cm. On the margin of each drawing and on the back side of each photo author name and abbreviation of the title of article should be placed. Figure's captures should be given in two longuaqes (first in the languaqe the article is writtes in and then in additional larguaqe). Figure's captures shoukl be also listed on separate page. Figures should be cited inthe text.

Additional information

On the separate page following information should be placed:

- mailing address (home or office),
- phone (home or/and office),
- e-mail.

Author is entitled to free of charge 20 copies of article. Additional copies or the whole magazine can be ordered at publisher at the one's expense.

Author is obliged to perform the author's correction, which should be accomplished within 3 days starting from the date of receiving of the text from the editorial stuff. Corrected text should be return to the editorial stuff personally or by mail. Correction marks should be placed on the margin of copies received from the editorial stuff or if needed on separate pages. In the case when the correction is not returned in said time limit, correction will be performed by technical editorial stuff of the publisher.

In case of changing of workplace or home address Authors are asked to inform the editorial stuff.