# We are IntechOpen, the world's leading publisher of Open Access books
# Built by scientists, for scientists

**6,900**
Open access books available

**186,000**
International authors and editors

**200M**
Downloads

**154**
Countries delivered to

Our authors are among the
**TOP 1%**
most cited scientists

**12.2%**
Contributors from top 500 universities

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

## Interested in publishing with us? Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

1

# Hardware Implementation of Fuzzy Controllers

Victor Varshavsky, Viacheslav Marakhovsky[1],
Ilya Levin[2] and Hiroshi Saito[3]
[1]*St. Petersburg State Politechnical University*
[2]*Tel Aviv University*
[3]*The University of Aizu*
[1]*Russian Federation*
[2]*Israel*
[3]*Japan*

## 1. Introduction

*Fuzzy logic control* is a methodology bridging *artificial intelligence* and traditional *control theory*. This methodology is usually applied in the only cases when accuracy is not of high necessity or importance. On the other hand, as it is stated in (TI SPRA028, Jan.1993), "Fuzzy Logic can address complex control problems, such as robotic arm movement, chemical or manufacturing process control, antiskids braking systems or automobile transmission control with more precision and accuracy, in many cases, than traditional control techniques … . Fuzzy Logic is a methodology for expressing operational laws of a system in linguistic terms instead of mathematical equations."

Wide spread of the fuzzy control and high effectiveness of its applications in a great extend is determined by formalization opportunities of necessary behavior of a controller as a "fuzzy" (flexible) representation. This representation usually is formulated in the form of logical (fuzzy) rules under linguistic variables of a type "If A then B".

The Fuzzy Logic methodology (Yager & Zadeh, 1992; Klir & Yuan, 1996) comprises three phases:

1. The *fuzzification* is a transformation of analog (continuous) input variables to linguistic ones, e.g., transformation of temperature into the terms *cool*, *warm*, *hot* or transformation of speed into the terms *negative big* (*NB*), *negative small* (*NS*), *zero* (*Z*)", *positive small* (*PS*), *positive big* (*PB*). Such transformation is realized by introduction of so-called *membership functions*, which define both a range of value and a degree of membership. For linguistic variables it is important not only which membership function a variable belongs to, but also a relative degree (weight) to which it is a member. A variable can have a weighted membership in several membership functions at the same time.
2. The *fuzzy inference* maps input linguistic variables onto output linguistic variables on the base a system of fuzzy rules of the type "IF A THEN B" For instance: "IF the temperature is *worm* THEN the speed is *Positive Small* (*PS*)" or "IF the speed is *Negative Big* (*NB*) THEN force is *ZERO*". Since input linguistic variables are weighted, the output linguistic variables can be obtained weighted as well. Traditional fuzzy logic approach comprises Mamdani- type and Sugeno-type inference methods. The Mamdani-type

www.intechopen.com

method is more intuitive and assumes the output variables as a fuzzy set. Fuzzy rules in it contain a *fuzzy precondition* part (after IF) and a *fuzzy consequence* part (after THEN). The Sugeno-type method expects the output variables to be singletons or dealing with consequents that are equations. So it is better suited for mathematical analysis, nonlinear system modeling and interpolation.

3. In the *defuzzification* phase, the weighted values of output linguistic variables obtained as a result of fuzzy inference have to be transformed to analogue (continuous) variables. This procedure is also based on membership functions. Two major methods are used for defuzzification:

- The *maximum* defuzzification method, wherein an output value is determined by the linguistic variable with the maximum weight;
- The *centroid* calculation defuzzification method, wherein an output value is determined by the weighted influence of all the active output membership functions.

As a rule, or at least in a great part of applications, a fuzzy controller is a transformer of input analog signals into an analog output signal. A linguistic variable is a *subjective* characteristic of an input analog variable, values of which are transformed on bases of given membership functions into a set of weighted values of corresponding linguistic variables. This procedure is called a fuzzification and it contains as its composite part the analog-digital transformation.

A set of combinations of weighted linguistic variables corresponds to each value combination of input analog variables. On bases of a system of fuzzy inference rules it is possible to receive the set of weighted output linguistic variables. Using these variables and their membership functions, with help of one of well known defuzzification methods it is possible to form values of the analog output variable. The defuzzification procedure also includes digital-analog transformation.

At present the most wide-spread way of fuzzy logic control implementation is using the programmable fuzzy controllers, which are available on the market together with the means of computer aided programming (e.g. Motorola's 8-bit 68HC11 and 16-bit 68HC12 microcontrollers or specialized fuzzy processors of Siemens 80C517/80C535 families). However, in spite of the implementation evidence and fuzzy controllers' accessibility this approach to controller implementation possesses some disadvantages, e.g. such as high cost and low throughput (that is especially important when fuzzy control in the control contour is used) etc.

This work shows that for a sufficient wide set of applications, fuzzy controllers can be implemented as rather simple CMOS devices, which can be used in embedded systems or as an IP core. What is the basic idea of the proposal?

A fuzzy controller is a deterministic device, for which one and only one value of the output analog variable corresponds to each value combination of the input analog variables. It means that the fuzzy controller should realize an analog function $Y = f(x_1, x_2, ..., x_n)$. It should be noticed that in suppressing majority of publications on fuzzy controllers, this function is given as a response surface and practically without exception this surface has a piecewise linear form.

There are two important questions:

1. How to transit from a standard specification of a fuzzy logic function to the specification of corresponding analog function?
2. How to transit from an analog function specification and/or from a standard specification of a fuzzy logic function to corresponding CMOS implementation?

First of all, let us address to membership functions. In most cases (Yager & Zadeh, 1992; Marks II, 1994; Klir & Yuan, 1996) , membership functions have a triangle or trapeze form (see Fig. 1).

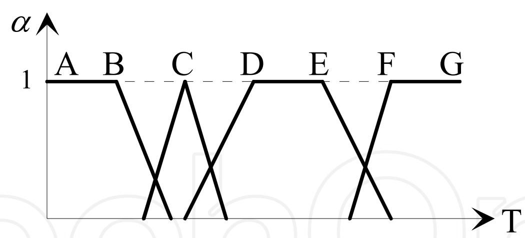



Fig. 1. Types of membership functions.

In Fig. 1 linguistic points (variables) A and B are *cold,* C is *fresh*, D and E are *worm*, F and G are *hot*. These points determine the connection of the linguistic variables with values of the analog variable T (T is *temperature*). Relatively to these points and similar points for other analog input variables we can compose a table of fuzzy rules connecting combinations of input linguistic variables with output linguistic variables.

On bases of membership functions we can put into accordance to the input and output linguistic variables a set of integer numbers splitting by appropriate way all diapason of changing of corresponding analog variables. Then the table of fuzzy rules will to determine by obvious way the function of multi-valued logic, values of which define the digit representation of the output linguistic variable on chosen value combinations of multi-valued input variables.

In other words, according to our concept, for a broad class of fuzzy controller specifications it is possible to construct corresponding tables connecting input and output membership functions. Frequently membership functions evenly divide the ranges of output variables' variations. If it is not so, the membership functions can be brought to even scale by increasing the number of gradations or, as it will be shown later, by introducing a certain equalization procedure for logical levels. Therefore, specification tables represent nothing but tables determining a specific multi-valued logical function. And what is more, for a number of implementations it is possible to neglect weighting and determining input linguistic variables and simply to use continuous-valued variables.

The above idea was in the focus of our research. We dealt with searching for simple basic multi-valued functions, which, from the one hand, would present a complete functional basis in the multi-valued logic, and from the other hand, could be efficiently implemented by CMOS technology.

## 2. Hardware implementation of fuzzy controllers

### 2.1 Summing amplifier as a multi-valued logical element

Summing amplifier's behavior, accurate to the members of the infinitesimal order that is determined by the amplifier's gain factor in disconnected condition (Fig. 2), is described as follows:

$$V_{out} = \begin{cases} V_{dd} & \text{if } \sum_{j=1}^{n} \frac{R_0}{R_j} (V_j - \frac{V_{dd}}{2}) \leq -\frac{V_{dd}}{2} \\ \frac{V_{dd}}{2} - \sum_{j=1}^{n} \frac{R_0}{R_j} (V_j - \frac{V_{dd}}{2}) & \text{if } \frac{V_{dd}}{2} > \sum_{j=1}^{n} \frac{R_0}{R_j} (V_j - \frac{V_{dd}}{2}) > -\frac{V_{dd}}{2} \\ 0 & \text{if } \frac{V_{dd}}{2} \leq \sum_{j=1}^{n} \frac{R_0}{R_j} (V_j - \frac{V_{dd}}{2}) \end{cases} \tag{1}$$

where $V_{dd}$ is the supply voltage, $V_j$ is the voltage on $j^{th}$ input, $R_j$ is the resistance of $j^{th}$ input, $R_0$ is the feedback resistance, and $V_{dd}/2$ is the midpoint of the supply voltage.

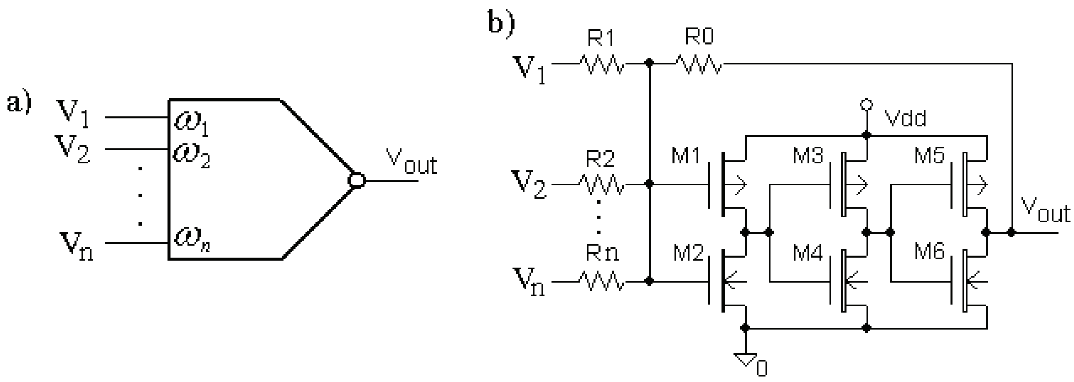



Fig. 2. Summing amplifier: a) general designation, b) CMOS implementation using symmetrical invertors.

Dependence of $V_{out}$ on $\sum_{j=1}^{n}\frac{R_0}{R_j}\cdot(V_j-\frac{V_{dd}}{2})$ is shown in Fig. 3 (a).

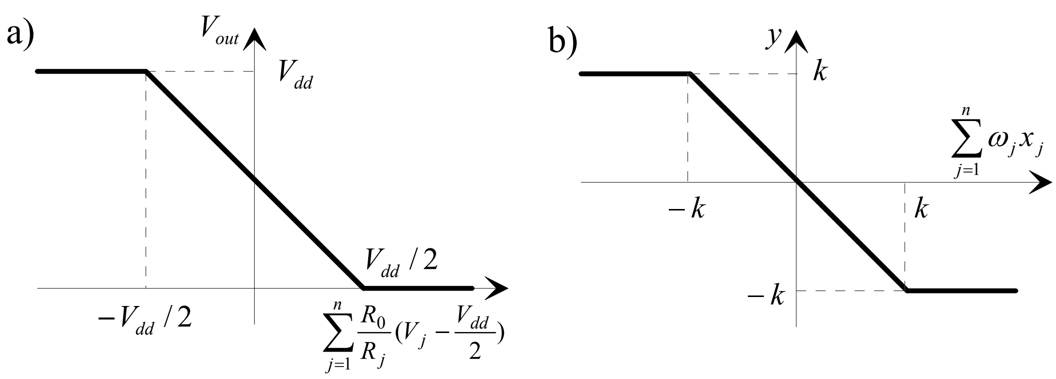



Fig. 3. Summing amplifier's behavior: a) within voltage coordinates; b) within multi-valued variable coordinates.

Let us split the source voltage $V_{dd}$ on $m$ = $2k$+1 voltage levels. Then replacing the input voltages $V_j - V_{dd}/2$ by $m$-valued logical variables $x_j = (2V_j - V_{dd})k/V_{dd}$ and the output voltage $V_{out}$ by $m$-valued variable $y$ and designating $R_0/R_j = \omega_j$ the system (1) can be represented as (2).

$$y(X)=S(\sum_{j=1}^{n}\omega_j\cdot x_j)=\begin{cases}+k & \text{if} \quad \sum_{j=1}^{n}\omega_j\cdot x_j\le -k\\ -\sum_{j=1}^{n}\omega_j\cdot x_j & \text{if } k>\sum_{j=1}^{n}\omega_j\cdot x_j>\text{-}k\\ -k & \text{if} \quad \sum_{j=1}^{n}\omega_j\cdot x_j\ge +k\end{cases} \tag{2}$$

Graphical view of (2) is shown in Fig.3 (b).

Later on, we will call the functional element, whose behavior is determined by the system (2), a multi-valued threshold element. When $\omega_j = 1$, $j = 1, 2, 3,$ , we will call it a majority element and designate as $maj(x_1, x_2, x_3)$.

### 2.2 Functional completeness of the threshold element

*The basic operation (or a set of basic operations) is called functionally completed in arbitrary-valued logic, if any function of this logic can be represented as superposition of the basic operations.*

There are some known functionally complete sets of functions. It is clear, that for proving the functional completeness of a certain new function it is sufficient to show that every function of the known functionally complete set can be represented as a superposition of the considered function. One of functionally complete functions in $m$-valued logic is the Webb's function (Post, 1921):

$$w(x,y) = [\max(x,y)+1]_{\bmod m}\,. \tag{3}$$

Therefore, for proving functional completeness of the threshold operation in multi-valued logic it is sufficient to show how the Webb's function can be represented through this operation (Varshavsky et al., 2003, 2004).

First, let us represent the function $\max(x_1, x_2)$ by threshold functions. To do this let us consider the function $f_a(x)$, such as

$$f_a(x) = \max(x,a) = \begin{cases} a & \text{if} \quad a \ge x \\ x & \text{if} \quad x > a \end{cases}, \quad |x| \le k, \quad |a| \le k. \tag{4}$$

The diagram of this function is shown in Fig. 4(a). The $-maj(x,-a,-k)$ function diagram is shown in Fig. 4(b). Actually, as far as $x < a$, $x - a - k < -k$ and $-maj(x,-a,-k) = -k$. Note that for all values of $x$,

$$f_a(x) = -maj(x,-a,-k) + a + k$$

as it follows from Fig.4, hence

$$f_a(x) = -maj(-maj(x,-a,-k),a,k)\,. \tag{5}$$

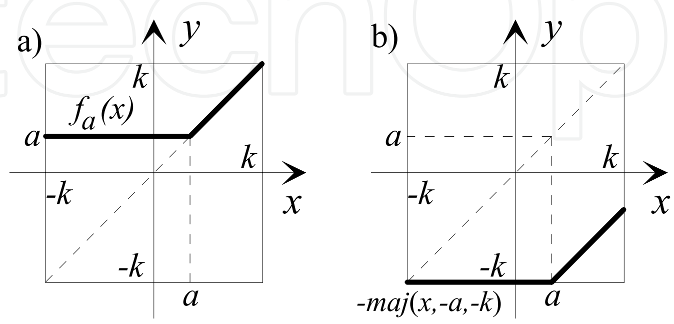



Fig. 4. Diagrams of the functions a) $f_a(x)$ and b) $-maj(x,-a,-k)$.

Taking into consideration

$$-maj(a,b,c) = maj(-a,-b,-c),$$

it follows from (5) that

$$\max(x_1, x_2) = maj(maj(x_1, -x_2, -k), -x_2, -k)\ . \tag{6}$$

Now let us consider the representation of the function $y = (x+1)_{\mathrm{mod}\,m}$, $x \geq 0$, $0 \leq y \leq m-1$ through threshold functions. First of all we designate $m = 2k+1$ and change the beginning of coordinates so that the function will have a form $y = (x+k+1)_{\mathrm{mod}(2k+1)} - k$, $x \geq -k$, $-k \leq y \leq +k$. To implement this function on threshold elements let us turn to the sequence of pictures in Fig. 5.

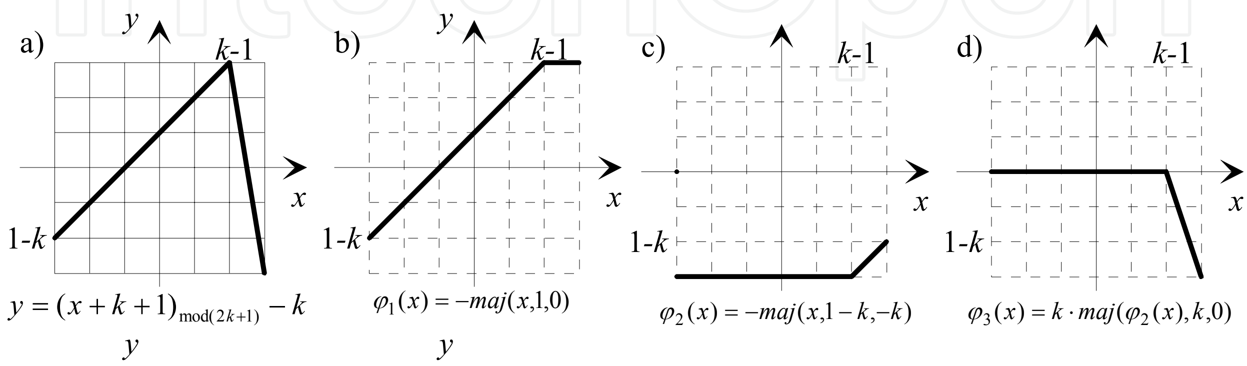



Fig. 5. Implementation of the function $y = (x+k+1)_{\mathrm{mod}(2k+1)} - k$.

It is easy to see that

$$(x+k+1)_{\mathrm{mod}(2k+1)} - k = \phi_1(x) + 2\varphi_3(x)$$

and obviously, this function can also be implemented on threshold elements as

$$y = maj(maj(x,1,0), k \cdot maj(maj(x,1-k,-k),-k,0), k \cdot maj(maj(x,1-k,-k),-k,0)).$$

Hence, the functional completeness of the summing amplifier in arbitrary-valued logic is shown. The proof procedure of functional completeness naturally does not give information about methods of effective synthesis. Some methods of a circuit design in the proposed basis will be developed later.

### 2.3 Fuzzy devices as multi-valued and analog circuits

Conventional implementation of fuzzy devices usually has the structure shown in Fig. 6. Analog variables $X = \{x_1, x_2, \ldots, x_n\}$ enter the fuzzy device input. Fuzzifier converts a set of analog variables $x_j$ into sets of weighted linguistic (digital) variables $A = \{a_1, a_2, \ldots, a_n\}$.

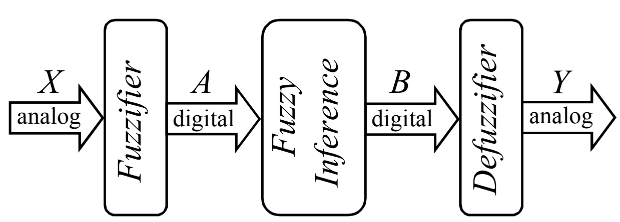



Fig. 6. Conventional structure of a fuzzy device implementation.

Fuzzy Inference block generates based on the fuzzy rules a set of weighted linguistic variables values $B = \{b_1, b_2, \ldots, b_k\}$.

Defuzzifier converts sets of weighted linguistic (digital) variables $B = \{b_1, b_2, ..., b_k\}$ into a set of output analog variables $Y = \{y_1, y_2, ..., y_k\}$.
As a rule, fuzzifier and defuzzifier include AD and DA (analog-digital and digital-analog) converters and are implemented on both levels (hardware and software). Fuzzy inference is usually implemented on the level of microprocessor software.
It is easy to see that each set of values of output analog variables unambiguously corresponds to some set of input analog variable values; hence a fuzzy device could be specified as a functional analog of a signal converter

$$Y(X) = \{y_1(X), y_2(X), ..., y_k(X)\}$$

and its output $Y$ determines a system of $n$-dimensional surfaces. In cases of sufficient simple membership functions (in known publications such functions are in majority), for fuzzy controller implementations as analog devices it is sufficient to provide a piecewise-linear approximation between a couples of points calculated as adjacent values of a multi-valued logic function.
Let $m = 2k+1$ linguistic variables $a_j$ ($a_j \in A$) correspond to values of analog variable $x_j$ ($xj \in X$). Then basing on a system of fuzzy rules, we can specify a system of $m$-valued logic functions, as follows:

$$B(A) = \{b_1(A), b_2(A), ..., b_k(A)\}\ . \tag{7}$$

Note that most publications describing fuzzy controllers contain tables specifying fuzzy controllers' behaviour as (7) and a plenty of publications contain piecewise-linear approximations of the corresponding surfaces.
The apparent conclusion can be made from the things mentioned above: if a fuzzy controller is represented as (7), it can be implemented as superposition of multi-valued threshold elements. In this case, owing linear behavior of the threshold element in the zone between the saturation levels ((2) and Fig. 3(b)), natural piecewise linear approximation appears between the discrete points of specification.
In the last subsection of this section some illustrations will be given to show that for a number of real applications the offered approach can provides simple and efficient circuits of controllers.

### 2.4 Fuzzy controller implementations as circuits from threshold elements

#### 2.4.1 Example 1

Let us consider the example, which is taken from (Kandel & Zedeh, 1993, pp. 81 – 86): "Design of a Rule-Based Fuzzy Controller for the Pitch Axis of an Unmanned Research Vehicle".
The fuzzy control rules for the considered device depend on the error value $e = ref - output$ and changing of error $ce = \dfrac{old\, e - new\, e}{sampling\ period}$. Fuzzifier gives seven linguistic variables for each of input analog variables (NB – negative big; NM – negative middle; NS – negative small; ZO – zero; PS – positive small; PM – positive middle; PB – positive big). The output has the same seven gradations. Corresponding 49 fuzzy rules are represented in Table 1.
Let us split evenly the source voltage (e.g. 3.5V) onto seven logical levels corresponding to linguistic levels and enumerate them with integer numbers from -3 to +3. Then Table 2 will represent Table 1 as the function of seven-valued logic.

| | e | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | NB | NM | NS | ZO | PS | PM | PB |
| ce | NB | ZO | PS | PM | PB | PB | PB | PB |
| | NM | NS | ZO | PS | PM | PB | PB | PB |
| | NS | NM | NS | ZO | PS | PM | PB | PB |
| | ZO | NB | NM | NS | ZO | PS | PM | PB |
| | PS | NB | NB | NM | NS | ZO | PS | PM |
| | PM | NB | NB | NB | NM | NS | ZO | PS |
| | PB | NB | NB | NB | NB | NM | NS | ZO |

Table 1. Table of Fuzzy Rules.

It is seen from Table 2 that the function is symmetric with respect to "North-West – South-East" diagonal and its values can be calculated as $e - ce$. This dependency is shown in Fig. 7.

| | e | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| ce | 0 | -3 | -2 | -1 | 0 | 1 | 2 | 3 |
| | -3 | 0 | 1 | 2 | 3 | 3 | 3 | 3 |
| | -2 | -1 | 0 | 1 | 2 | 3 | 3 | 3 |
| | -1 | -2 | -1 | 0 | 1 | 2 | 3 | 3 |
| | 0 | -3 | -2 | -1 | 0 | 1 | 2 | 3 |
| | 1 | -3 | -3 | -2 | -1 | 0 | 1 | 2 |
| | 2 | -3 | -3 | -3 | -2 | -1 | 0 | 1 |
| | 3 | -3 | -3 | -3 | -3 | -2 | -1 | 0 |

Table 2. The Seven-Valued Function.

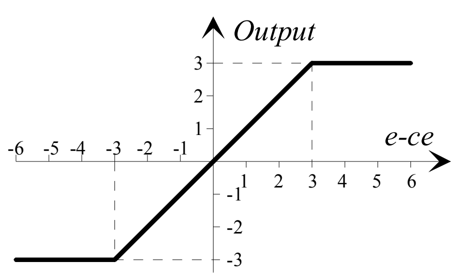



Fig. 7. Graphical representation of the function specified by Table 2.

It apparently follows from comparison of Fig. 3 (b) and Fig. 7 that in order to reproduce the function specified by Table 2 it is sufficient to have one two-input summing amplifier and one one-input amplifier that will be called inverter.
Note that inversion of logic variables lying within $-k \div +k$ interval is the operation of diametric negation $\overline{x} = -x$; the operation $V_{out} = V_{dd} - V_{in}$ corresponds to it in the terms of summing amplifier's input and output voltages. Thus CMOS circuit containing 12 transistors and 5 resistors, which implements our function, is shown in Fig. 8.

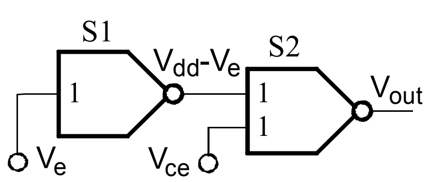



Fig. 8. Implementation of the fuzzy controller specified by Table 2.

### 2.4.2 Example 2

This example is taken from (Kandel & Zedeh, 1993, pp. 168 – 172): "Manipulator for Man-Robot Cooperation (Control Method of Manipulator/Vehicle System with Fuzzy Inference)".
In the considered example the experimental manipulator has two force/torque sensors. One of them is the operational force sensor $F_h$; the other is "the environmental force sensor" $\omega$. Each of input and output variables of the manipulator controller is represented with three linguistic variables – S (small), M (middle) and B (big). The controller has five fuzzy rules, as it follows:
If $\omega = S$ then $Output = B$;
If $\omega = B$ then $Output = S$;
If $\omega = M$ and $F_h = S$ then $Output$ = S;
If $\omega = M$ and $F_h = M$ then $Output = M$;
If $\omega = M$ and $F_h = B$ then $Output = B$.
The controller *Output* is three-valued logic function specified in Table 3.

| | | $F_h$ | | |
|---|---|---|---|---|
| | | –1 | 0 | +1 |
| $\omega$ | –1 | +1 | +1 | +1 |
| | 0 | –1 | 0 | +1 |
| | +1 | –1 | –1 | –1 |

Table 3. The ternary function.

It can be simply proved by trivial substitution that $Output = maj(2\omega, -F_h, 0)$ and CMOS implementation coincides with the circuit shown in Fig. 8, if make substitutions $V_e = V_{F_h}$, $V_{ce} = V_\omega$ and change the weight of the input $V_\omega$ to 2.

### 2.4.3 Example 3. Fuzzy controller for washing machine

This example is taken from Aptronix Incorporated (http://www.aptronix.com/fuzzynet).

*A. Controller specification*

*Input variables:*
*Dirtiness of clothes: Large (L), Medium (M), and Small (S);*
*Type of dirtiness: Greasy (G), Medium (M), and Not Greasy (NG).*
Output variable is *washing time* (minutes): *Very Long* (*VL*), *Long* (*L*), *Medium* (*M*), *Short* (*S*), and *Very Short* (*VS*). *Fuzzy rules* are represented in Table 4.

| *Wash. time* | | *Dirtiness of clothes* | | |
|---|---|---|---|---|
| | | *S* | *M* | *L* |
| *Type of dirt.* | *NG* | *VS* | *S* | *M* |
| | *M* | *M* | *M* | *L* |
| | *G* | *L* | *L* | *VL* |

Table 4. Matrix of linguistic variables.

According to our approach Table 4 can be transformed into the table of multi-valued logic variables (Table 5).

| Wash. time | | Dirtiness of clothes (Y) | | |
|---|---|---|---|---|
| | | –2 | 0 | +2 |
| Type of dirt. (X) | –2 | –2 | –1 | 0 |
| | 0 | 0 | 0 | +1 |
| | +2 | +1 | +1 | +2 |

Table 5. Matrix of multi-valued variables.

In this table the output variable *Wash. time* has 5 logical levels but input variables $X$ and $Y$ have only three. Because of change ranges of the output and input variables should be the same in the Table 5 logical levels of input variables $X$ and $Y$ are –2, 0, and +2.

*B. Functional decomposition*

Let us represent the washing time matrix as a sum of two matrices:

$$\begin{array}{ccccc} \text{Wash time} & & \varphi_1 & & \varphi_2 \\ \begin{vmatrix} -2 & -1 & 0 \\ 0 & 0 & +1 \\ +1 & +1 & +2 \end{vmatrix} & = & \begin{vmatrix} -1 & -1 & 0 \\ 0 & 0 & +1 \\ +1 & +1 & +2 \end{vmatrix} & + & \begin{vmatrix} -1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{vmatrix} \end{array} \tag{8}$$

or (*Wash. time*) = $S(-\varphi_1-\varphi_2)$ were $S$ is the function of summing amplifier with saturation.
Let us take into consideration a function of one variable

$$\varphi_3(Y) = S(0.5 \cdot Y - 2) = \begin{vmatrix} +2 & +2 & +1 \end{vmatrix} . \tag{9}$$

In (9) $Y$ corresponds to the *dirtiness of clothes* and varies from –2 to +2 as follows

$$Y = \begin{vmatrix} -2 & 0 & +2 \end{vmatrix} .$$

Now the following intermediate sum is introduced:

$$\varphi_3(Y) - 0.5 \cdot X - 2 = \begin{vmatrix} +1 & +1 & 0 \\ 0 & 0 & -1 \\ -1 & -1 & -2 \end{vmatrix} = -\varphi_1 . \tag{10}$$

Here $X$ corresponds to the *type of dirtiness* and varies also from –2 to 2 as follows

$$X = \begin{vmatrix} -2 \\ 0 \\ +2 \end{vmatrix} .$$

From (8) and (10) it is easy to see that (10) is $-\varphi_1$ and

$$(\textit{Wash. time}) = S(\varphi_3(Y) - 0.5 \cdot X - 2 - \varphi_2) .$$

Now let us introduce the function:

$$\varphi_4(X,Y) = S(X+Y+4) = \begin{vmatrix} 0-2-2 \\ -2-2-2 \\ -2-2-2 \end{vmatrix} \tag{11}$$

and form the second intermediate sum:

$$0.5 \cdot \varphi_4(X,Y) + 1 = \begin{vmatrix} +1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{vmatrix} = -\varphi_2 \,. \tag{12}$$

Finally

$$\begin{aligned}(Wash\ time) &= S[\varphi_3(Y) - 0.5 \cdot X - 1 + 0.5\varphi_4(X,Y)] = \\ &S_4[S_1(0.5Y-2) + 0.5S_2(X+Y+4) + 0.5S_3(X) - 1]\,.\end{aligned} \tag{13}$$

In Fig. 9 the CMOS implementation of the expression (13) is presented. The circuit is implemented as the superposition of four multi-valued threshold elements.

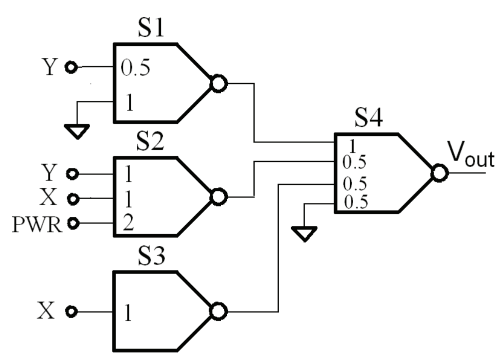



Fig. 9. CMOS implementation of fuzzy controller for washing machine.

The result of the SPICE simulation of the circuit in Fig. 9 is shown in Fig. 10 in the form of response surface.

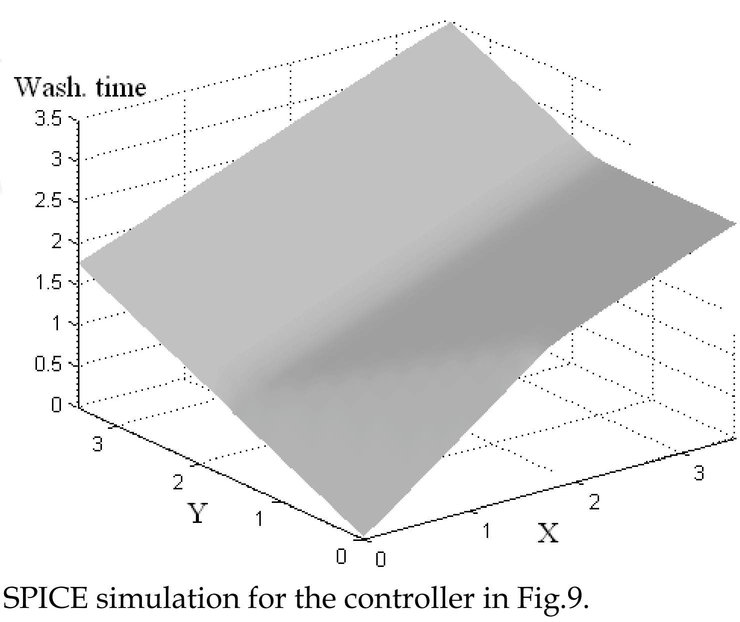



Fig. 10. Results of SPICE simulation for the controller in Fig.9.

In Fig. 10 all variables are represented in voltages. The correspondence of logical values to voltages is shown in Table 6. It is easy to see that the controller output signal represented by the surface in Fig. 10 has linear approximation between adjacent logical levels.

| -2 | -1 | 0 | 1 | 2 |
|---|---|---|---|---|
| 0V | 0.875V | 1.75V | 2.625V | 3.5V |

Table 6. Correspondence voltages to logical levels.

## 3. Universal method of implementing fuzzy inference rules

It was shown in 2.2 and (Varshavsky et al., 2003) that a summing amplifier with saturation is a functionally complete element in any multi-valued logic (of an arbitrary value). Thus it may serve as a basis for hardware implementation of fuzzy devices.

The study subject is design techniques for analog CMOS circuits implementing fuzzy controller multi-valued functions.

Without departing from the general character of the study, let us suppose that the logic has odd value $m = 2k+1$. Let's also assume that $X = \{x_1, x_2, \ldots, x_n\}$, $-k \leq x_j \leq +k$, , is a set of input multi-valued variables and $y = F(X)$ is the output variable. Then for a function of multi-valued logic it is possible to build an analog of the Shannon's decomposition in the binary logic:

$$y_i = F_i(X) = \bigcup_{\alpha=-k}^{+k} [\text{if } x_j = \alpha \text{ then } y = F(x_j = \alpha, X \setminus x_j)]. \tag{14}$$

Equation (14) can be further expanded so that it would be possible to build an realizing circuit using the variables exclusion method. To this effect, we need a sub-circuit implementing the function:

$$\text{if} \quad Z = A \quad \text{then } y = F(Z = A, X \setminus Z) \tag{15}$$

where $Z \subset X$ and $A$ is a value combination of the variable set $Z$.[1]

Having a basic element (sub-circuit realizing (15)), we can implement a fuzzy device directly according to the system of fuzzy rules. However, note that equations (14) and (15) represent multi-valued functions in a piecewise-constant manner. An example of a 7-valued function is given in Fig. 11(a).

Taking into account fuzzification and defuzzification procedures in fuzzy logic, corresponding multi-valued logic function should has at least piecewise-linear approximation between adjacent logical levels. Fig. 11(b) gives an example of such a representation of the function with evenly distributed logical values of the input and the output in the range of corresponding voltages.

[1] It is possible to add **else** in (15) that can be defined by circuit requirements.

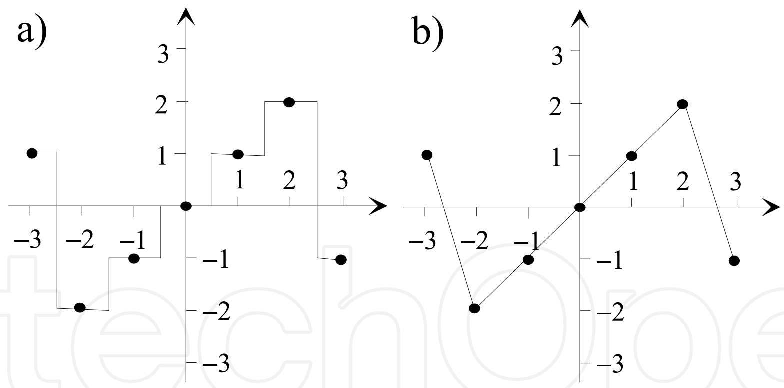



Fig. 11. Example of a seven-valued function: a) piecewise-constant representation; b) piecewise-linear representation.

### 3.1 Masking inputs of summing amplifiers

Let us rewrite the definition (2) of the inverting summing amplifier with saturation in the following form:

$$S(A\cdot X;\beta)=\begin{cases} +k & \text{if } \sum_{i=1}^{n}\alpha_i\cdot x_i+\beta\le -k \\ -\sum_{i=1}^{n}\alpha_i\cdot x_i-\beta & \text{if } +k>\sum_{i=1}^{n}\alpha_i\cdot x_i+\beta>-k \\ -k & \text{if } +k\le\sum_{i=1}^{n}\alpha_i\cdot x_i+\beta \end{cases} \qquad (16)$$

where $A = \{\alpha_1,\alpha_2,...,\alpha_n\}$ is a set of weight coefficients, $X = \{x_1,x_2,...,x_n\}$ is a set of analog or multi-valued variables, $\beta$ is a constant symbolizing a threshold, and $\pm k$ is a saturation value (in the case of $m$-valued logic, $m = 2k+1$).

Let us introduce a masking function $M_\alpha(x)$ by the next way:

$$M_\alpha(x)=\begin{cases} +k & \text{if } x\le\alpha-1 \\ k(\alpha-x) & \text{if } \alpha-1<x<\alpha+1 \\ -k & \text{if } \alpha+1\le x \end{cases} \qquad (17)$$

where $\alpha$ ($-k \le \alpha \le +k$) is a fixed value of the variable $x$. It can be easily seen that when $x = \alpha$, $M_\alpha(x) = 0$. Fig. 12 illustrates an example of the function $M_{-1}(x)$ for $m = 7$.

Taking into account that the source voltage $V_{dd}$ has the logical value equal to $+k$ and the ground potential $V_{gnd}$ has the logical value equal to $-k$, the mask-function can be easily implemented on bases of summing amplifier as

$$M_\alpha(x)=\begin{cases} S(kx-\alpha\cdot V_{dd}) & \text{if } \alpha<0 \\ S(kx) & \text{if } \alpha=0 \\ S(kx+\alpha\cdot V_{gnd}) & \text{if } \alpha>0 \end{cases} \qquad (18)$$

where $x$ ($V_{gnd} \le x \le V_{dd}$) is measured in voltages.

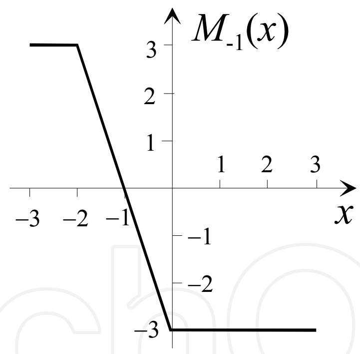



Fig. 12. $M_{-1}$ $(x)$ diagram for $m$ = 7.

Using the mask-function $M_\alpha(x)$ it is possible to implement the rule

$$\text{if } x = \alpha \text{ then } y = F(x = \alpha, Y) \text{ else } y = 0\,, \tag{19}$$

which extracts the value of the function $F(x = \alpha, Y)$ in the point $x = \alpha,$ using the circuit from summing amplifiers shown in Fig. 13.

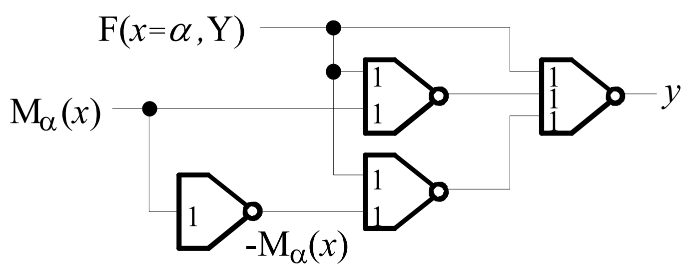



Fig. 13. Implementation of the rule (19).

This implementation can be written in analytical form as

$$y = S\{S[M_\alpha(x) + F(x = \alpha, Y)] + S[S(M_\alpha(x)) + F(x = \alpha, Y)] + F(x = \alpha, Y)\}. \tag{20}$$

For example, in the case when $\alpha$ = –1, $F(x$ = –1, $Y)$ = 2, and $m$ = 7, the behavior of the circuit in Fig.13 can be represented by Fig.14.

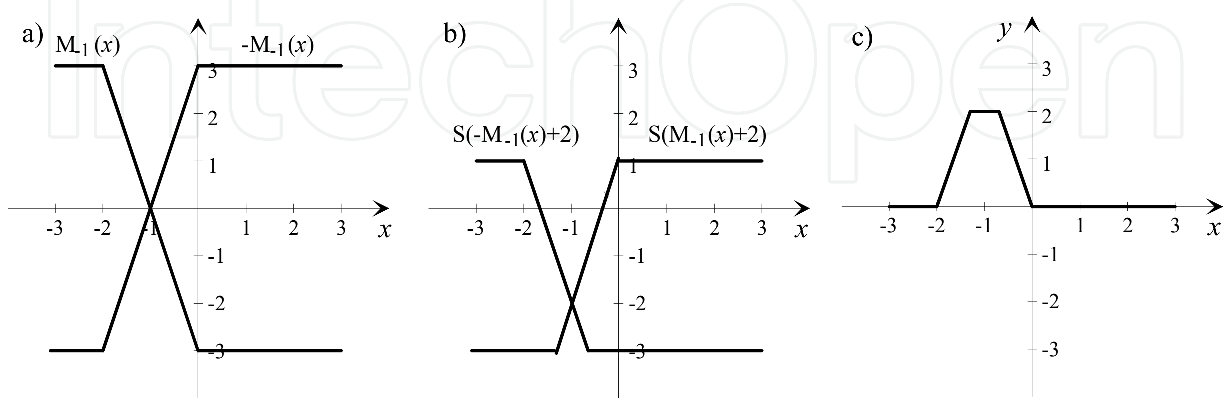



Fig. 14. Implementation example of the rule (19).

Analyzing the implementation of the rule (19) it is possible to see that in it the condition $x = \alpha$ is realized as the condition $M_\alpha(x)$ = 0.

### *3.2* Mask-functions of other types

To decrease the number of variables, which an $m$-valued logical function depends on, by one using the analog of Shannon's decomposition (14) we need to implement $m$ rules of the type (19) and to find $m$ components $F(x = \alpha, Y)$, $-k \le \alpha \le +k$, of the decomposition. Sometimes the number of rules can be reduced, if the function $F(x,Y)$ doesn't change on some interval of changing logical values of the variable $x$. A single rule can correspond to such interval of the variable $x$ and the conditional part of this rule can have one of three forms: $\alpha \le x \le \beta$, $x \le \beta$, $\alpha \le x$ where $-k \le \alpha < \beta \le +k$. For the condition $\alpha \le x \le \beta$ let us construct the following mask-function:

$$M_{\alpha,\beta}(x) = \begin{cases} +k, & \text{if } \ x \ge \beta + 1 \\ -M_{\alpha-1}(x) - M_{\beta+1}(x), & \text{if } \ \alpha - 1 < x < \beta + 1 \\ -k, & \text{if } \ x \le \alpha - 1 \end{cases} \tag{21}$$

It is easy to see (Fig. 15(b)) that on the interval $\alpha \le x \le \beta$ this function takes the value 0.
In the case when $\alpha = -k$ or $\beta = k$, this mask-function will have one of the forms:

$$M_{-k,\beta}(x) = \begin{cases} +k, & \text{if } \ x \ge \beta + 1 \\ +k - M_{\beta+1}(x), & \text{if } \ x < \beta + 1 \end{cases} \quad \text{or} \tag{22}$$

$$M_{\alpha,+k}(x) = \begin{cases} -\mathrm{M}_{\alpha\text{-}1}(x) - k, & \text{if } \ \alpha - 1 < x \\ -k, & \text{if } \ x \le \alpha - 1 \end{cases} \tag{23}$$

and represents conditions $x \le \beta$ or $\alpha \le x$ respectively.
Mask-functions (21), (22), and (23) can be implemented on bases of summing amplifiers as

$$M_{\alpha,\beta}(x) = S[M_{\alpha-1}(x) + M_{\beta+1}(x)]\,, \tag{24}$$

$$M_{-k,\beta}(x) = S[V_{gnd} + M_{\beta+1}(x)]\,, \tag{25}$$

$$M_{\alpha,k}(x) = S[M_{\alpha-1}(x) + V_{dd}]\,. \tag{26}$$

Let us look how the masking can be performed for a wider scope of the variable changes, such as:

$$\text{if } \alpha \le x \le \beta \text{ then } y = F(\alpha \le x \le \beta, Y) = \Phi(Y) \text{ else } y = 0. \tag{27}$$

Using the mask-function $M_{\alpha,\beta}(x)$ it is possible to transform the rule (27) into the following form:

$$\text{if } M_{\alpha,\beta}(x) = 0 \text{ then } y = F(\alpha \le x \le \beta, Y) = \ \Phi(Y) \ \text{ else } \ y = 0. \tag{28}$$

The rule (28) can be implemented with the circuit shown in Fig. 13, if to change in it the inputs $M_\alpha(x)$ and $F(x = \alpha, Y)$ with the inputs $M_{\alpha,\beta}(x)$ and $\Phi(Y)$ respectively. This implementation is represented analytically as

$$y = S\{S[M_{\lambda,\delta}(x) + \Phi(Y)] + S[S(M_{\lambda,\delta}(x)) + \Phi(Y)] + \Phi(Y)\}. \tag{29}$$

The sequence of pictures in Fig. 15 illustrates the implementation of the rule

$$\text{if } -2 \le x \le +1 \text{ then } y = -2 \text{ else } y = 0$$

for the case of ($m = 9$)-valued logic.

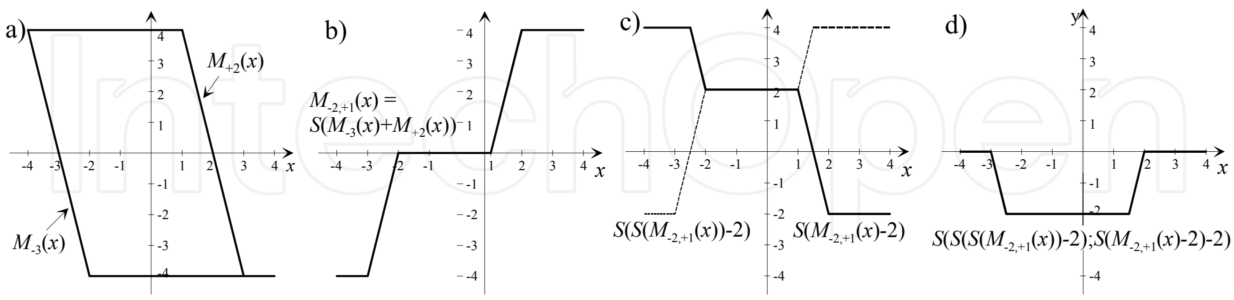



Fig. 15. Example of mask-functions application.

### 3.3 An application example of interval masking

For further explanation of the matter discussed in 3.2, let us recall an example from (Marks II, 1994, pp. 123 – 128) "*A Fuzzy Logic Force Controller for a Stepper Motor Robot*".
The fuzzy controller implements the function of two analog variables: *position error* and *force error*, which will be designate as $x$ and $y$ respectively. Each of the variables $x$ and $y$ is represented with 7 linguistic variables: NL, NM, NS, ZE, PS, PM, PL, and their membership functions are shown in Fig. 16.

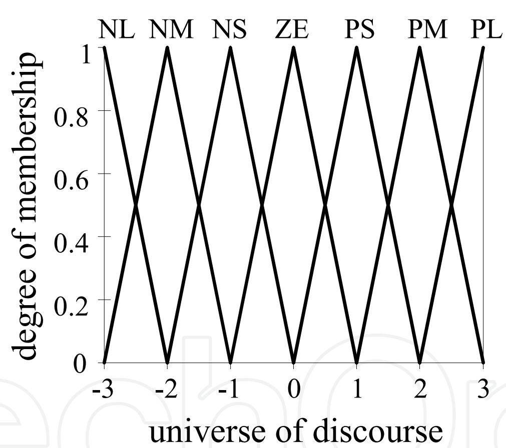



Fig. 16. Fuzzy sets for *force error* and *position error* inputs.

The Inference Engine Rule Matrix for the output linguistic variable from the cited work looks as it is shown in Table 7.
Let us transform the Table 7 into the Table 8 taking into account that we are going to produce fuzzy inference calculating values of the corresponding multi-valued logic function.
Table 8 comprises only two different columns defining two functions depending on the variable *force error* (Table 9).
Fig. 17 illustrates graphs of these functions. It is easy to see that the function $F_1(y)$ looks like mask-function $M_{-1,1}(y)$ but has different slops of the lines. By analogy with (17), (18), (21), (24), Fig. 15(a), and Fig. 15(b), it is possible to construct the function $F_1(y)$ in accordance with graphics in Fig. 18.

| | | position error ($x$) | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | NL | NM | NS | ZE | PS | PM | PL |
| | NL | NM | NL | NL | NL | NL | NL | NM |
| | NM | NS | NM | NM | NM | NM | NM | NS |
| | NS | ZE | NS | NS | NS | NS | NS | ZE |
| force error ($y$) | ZE | ZE | ZE | ZE | ZE | ZE | ZE | ZE |
| | PS | ZE | PS | PS | PS | PS | PS | ZE |
| | PM | PS | PM | PM | PM | PM | PM | PS |
| | PL | PM | PL | PL | PL | PL | PL | PM |

Table 7. Rule matrix of the inference engine.

| | | position error ($x$) | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | -3 | -2 | -1 | 0 | 1 | 2 | 3 |
| | -3 | -2 | -3 | -3 | -3 | -3 | -3 | -2 |
| | -2 | -1 | -2 | -2 | -2 | -2 | -2 | -1 |
| | -1 | 0 | -1 | -1 | -1 | -1 | -1 | 0 |
| force error ($y$) | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 |
| | 2 | 1 | 2 | 2 | 2 | 2 | 2 | 1 |
| | 3 | 2 | 3 | 3 | 3 | 3 | 3 | 2 |

Table 8. Matrix of the multi-valued logic function.

| | force error ($y$) | | | | | | |
|---|---|---|---|---|---|---|---|
| | -3 | -2 | -1 | 0 | 1 | 2 | 3 |
| $F_1(y)$ | -2 | -1 | 0 | 0 | 0 | 1 | 2 |
| $F_2(y)$ | -3 | -2 | -1 | 0 | 1 | 2 | 3 |

Table 9. Two different functions of the *force error.*

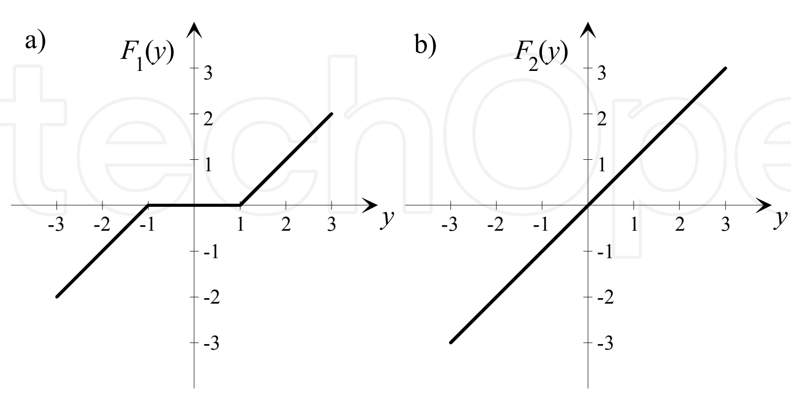



Fig. 17. Components of the function defined by Table 8 and decomposed relative to variable $x$.

As a result, the functions $F_1(y)$ and $F_2(y)$ can be implemented as

$$F_1(y) = S_3[\tfrac{2}{3}S_2(\tfrac{3}{2}y + \tfrac{3}{2}V_{gnd}) + \tfrac{2}{3}S_1(\tfrac{3}{2}y + \tfrac{3}{2}V_{dd})],\;\; F_2(y) = y. \tag{30}$$

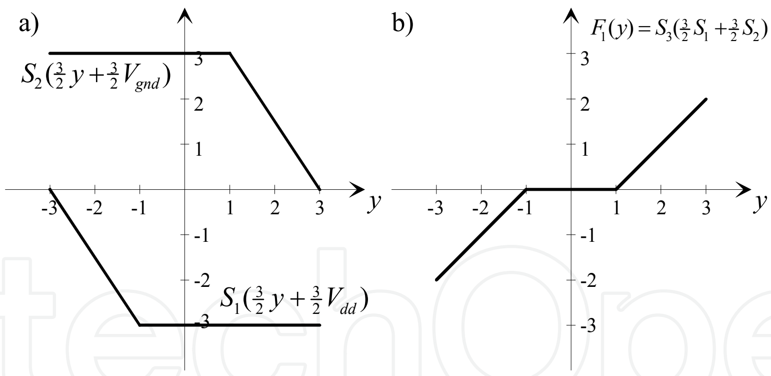



Fig. 18. Constructing of the function $F_1(y)$.

It is seen from the Table 7 and Table 8 that the behaviour of the controller's output in the decomposition by variable $x$ has the form:

$$\text{if } NM \le x \le PM \text{ then } Output = F_2(y) \text{ else } Output = F_1(y); \text{ or}$$

$$\text{if } -2 \le x \le +2 \text{ then } Output = F_2(y) \text{ else } Output = F_1(y). \tag{31}$$

It is possible to split the rule (31) into two rules and represent them as:

$$\text{if } M_{-2,+2}(x) = 0 \text{ then } Output = F_2(y) \text{ else } Output = 0; \tag{32}$$

$$\text{if } M_{-2,+2}(x) \ne 0 \text{ then } Output = 0 \text{ else } Output = F_1(y). \tag{33}$$

The rule (32) can be implemented in accordance with (29) and (30) and (24) as

$$\begin{cases} M_{-2,+2}(x) = S_4(M_{-3}(x) + M_{+3}(x)), \\ -M_{-2,+2}(x) = S_5(M_{-2,+2}(x)), \\ \Phi_1 = S_{11}\{S_6[M_{-2,+2}(x) + F_2(y)] + S_7[-M_{-2,+2}(x) + F_2(y)] + F_2(y)\}. \end{cases}$$

It is easy to check that the rule (33) can be implemented in accordance with the structural scheme shown in Fig. 13, in which the output amplifier has the weight equal to 2 of the input $F_1(y)$:

$$\Phi_2 = -S_{10}\{S_9[M_{-2,+2}(x) + F_1(y)] + S_8[-M_{-2,+2}(x) + F_1(y)] + 2F_1(y)\}.$$

Finally the output of the controller can be calculated as

$$Output = \Phi_1 + \Phi_2 .$$

For producing this summation it is possible to use summing amplifier $S_{11}$

$$Output = S_{11}(-\Phi_1; -\Phi_2).$$

Fig. 19 illustrates the structural scheme of the controller implementation with elements containing designations of input weights.

The controller circuit has been constructed from three-stage push-pull CMOS operational amplifiers with 1-MegOhm resistors in the feedback (Fig. 2(b)). It's functioning has been checked with SPICE simulation (MSIM 8). MOSIS BSIM3v3.1, level 7 model of 0.4μm transistors has been used. In this paper, all other SPICE simulation experiments with designed circuits of controllers have been executed under the same conditions.

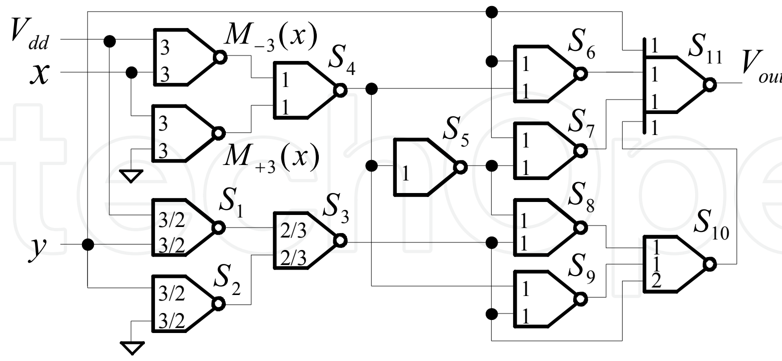



Fig. 19. Structural diagram of the controller.

In the experiments with the controller presented in Fig. 19, source voltage was 3.5V, input variable *x* changed linearly from 0V to 3.5V, input variable *y* changed discreetly in accordance with its logical values and kept constant value within one cycle of *x* changing. For the controller constructed from 3-stage elements results of SPICE simulation are shown in Fig. 20. It is possible to see that the functioning of the controller is correct (logical values of the circuit output depend on the logical values of the input variables in accordance with Table 7 and 8).

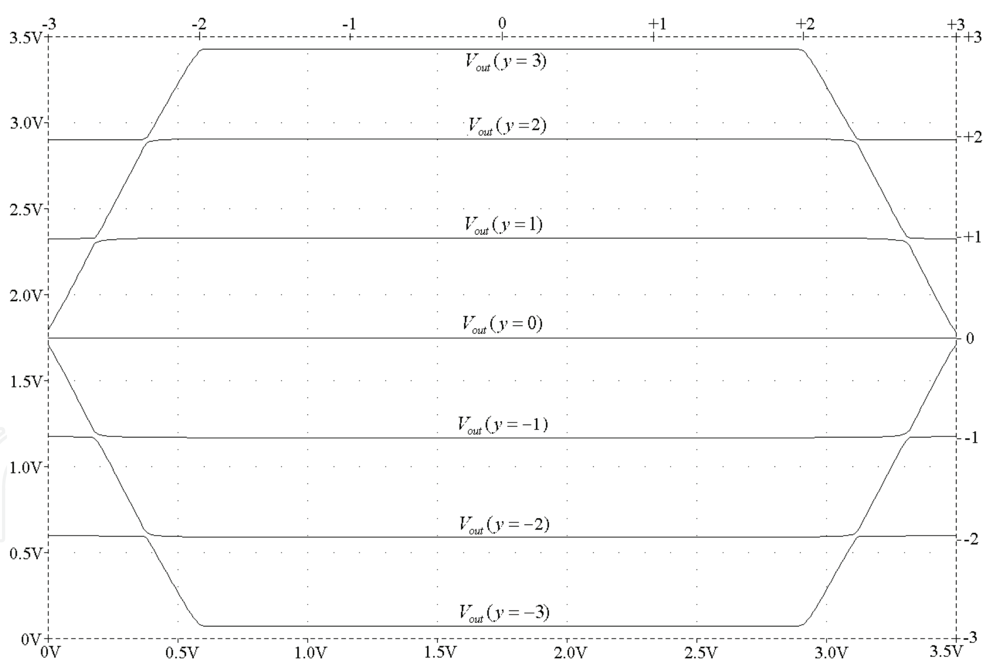



Fig. 20. SPICE simulation results for the controller constructed from 3-stage summing amplifiers.

## 4. Particular methods of fuzzy inference implementation

The universal method of implementing multi-valued logic functions proposed in the previous section can be always used but often can give inappropriate results due to its

universality. For this reason some particular design methods for fuzzy inference part of controllers were developed. These methods utilize specific properties of certain multi-valued logical function descriptions corresponding to sets of fuzzy inference rules.
According to the approach described above, an initial set of fuzzy rules is represented in the form of a matrix or matrices defining multi-valued logical functions. As a rule these matrices cannot be directly implemented. They must be decomposed into component matrices with relatively simple configuration of elements allocation, for which rather simple implementations can be find. The topologies of valuable elements inside of such component matrices can be specified as symmetrical, diagonal, matrixes with linear configurations of elements, with elements located along rows and columns, matrices containing single valuable element, and others.
The best way to introduce particular design methods is to show possible matrix decomposition into a set of implementable matrices on bases of a real design example.
Let us take the description of the rather complex fuzzy controller from the patent (Kimura & Kawawa, 1993) of Toyota Motors Corporation. The controller calculates a regeneration time decision coefficient $R$ on the base on a differential pressure coefficient $K_p$ and total fuel consumption $Q_f$. The set of fuzzy rules in terms of linguistic variables is represented in Table 10. Transformations of the input and output analog signals are performed in accordance with corresponding membership functions.

| | | $K_p$ | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | NB | NM | NS | ZO | PS | PM | PB |
| $Q_f$ | NB | NB | NB | NM | NS | ZO | PB | PB |
| | NM | NB | NM | NS | NS | ZO | PB | PB |
| | NS | NM | NM | NS | ZO | ZO | PS | PB |
| | ZO | NM | NS | ZO | ZO | PS | PM | PB |
| | PS | NS | ZO | ZO | PS | PS | PM | PB |
| | PM | ZO | ZO | PS | PM | PM | PM | PB |
| | PB | PS | PS | PM | PM | PM | PB | PB |

Table 10. Fuzzy rules for regeneration time decision coefficient $R$.

Analysis of the membership functions in (Kimura & Kawawa, 1993) of linguistic variables representing input and output analog signals shows that the linguistic variables having maximum weight are evenly distributed within the change ranges of corresponding analog signals. It means that without losing the accuracy of representation, these linguistic variables can be replaced with logical values as it is shown in Table 11.

| x\y | -3 | -2 | -1 | 0 | 1 | 2 | 3 |
|---|---|---|---|---|---|---|---|
| -3 | -3 | -3 | -1 | -1 | 0 | 3 | 3 |
| -2 | -3 | -2 | -1 | -1 | 0 | 3 | 3 |
| -1 | -2 | -2 | -1 | 0 | 0 | 1 | 3 |
| 0 | -2 | -1 | 0 | 0 | 1 | 2 | 3 |
| 1 | -1 | 0 | 0 | 1 | 1 | 2 | 3 |
| 2 | 0 | 0 | 1 | 2 | 2 | 2 | 3 |
| 3 | 1 | 1 | 2 | 2 | 2 | 3 | 3 |

Table 11. The 7-valued logical function

In this table, signals $Q_f$ and $K_p$ are changed with 7-valued logic variables $x$ and $y$ respectively.

### 4.1 Extracting a symmetrical component matrix

Let the Table 11 of the controller is represented as initial matrix $M$, which, in its turn, can be represented as sum of two component matrixes ($M_1$ and $M_2$).

$$\begin{array}{ccccc} M & & M_1 & & M_2 \\ \begin{vmatrix} -3-3-2-1\pm0+3+3 \\ -3-2-1-1\pm0+3+3 \\ -2-2-1\pm0\pm0+1+3 \\ -2-1\pm0\pm0+1+2+3 \\ -1\pm0\pm0+1+1+2+3 \\ \pm0\pm0+1+2+2+2+3 \\ +1+1+2+2+2+3+3 \end{vmatrix} & = & \begin{vmatrix} -3-3-2-2-1\pm0\pm0 \\ -3-2-2-1\pm0\pm0+1 \\ -2-2-1\pm0\pm0+1+1 \\ -2-1\pm0\pm0+1+1+2 \\ -1\pm0\pm0+1+1+2+2 \\ \pm0\pm0+1+1+2+2+3 \\ \pm0+1+1+2+2+3+3 \end{vmatrix} & + & \begin{vmatrix} \le0\le0\pm0+1+1+3+3 \\ \le0\pm0+1\pm0\pm0+3\ge2 \\ \pm0\pm0\pm0\pm0\pm0\pm0\ge2 \\ \pm0\pm0\pm0\pm0\pm0+1\ge1 \\ \pm0\pm0\pm0\pm0\pm0\pm0\ge1 \\ \pm0\pm0\pm0+1\pm0\pm0\ge0 \\ +1\pm0+1\pm0\pm0\ge0\ge0 \end{vmatrix} \end{array}$$

Matrix $M_1$ corresponds to a symmetrical component and $M_2$ corresponds to nonsymmetrical residual component.

Matrix $M_1$ is symmetrical relative to the side diagonal. Its components can be represented as a function $f_1(z)$ of one variable $z = (x+y)/2$. After performing the linear approximation between adjacent logical levels this function will has the form shown in Fig. 21.

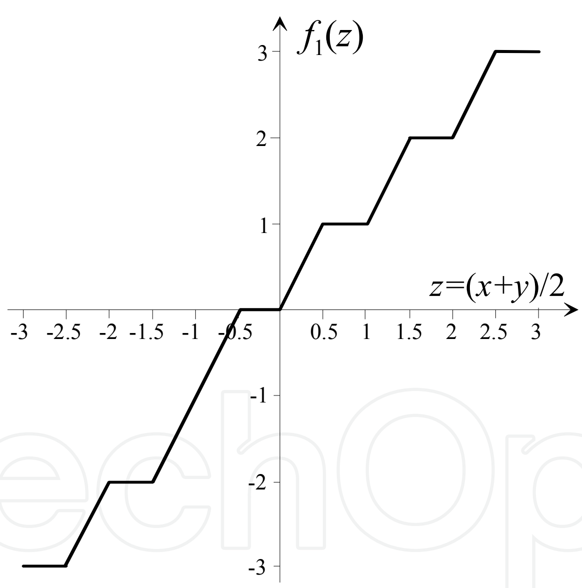



Fig. 21. Graph of the function $f_1(z)$.

To implement the function $f_1(z)$ let us represent it as a sum of 5 subfunctions $\alpha_j(z)$ shown in Fig. 22(a). It is easy to see, that

$$f_1(z) = \sum_{j=1}^{5} \alpha_j(z)\,. \tag{34}$$

Let us consider formation of $\alpha_j(z)$ using summing amplifiers on the example of $\alpha_1(z)$. For this let us address to Fig. 22(b).

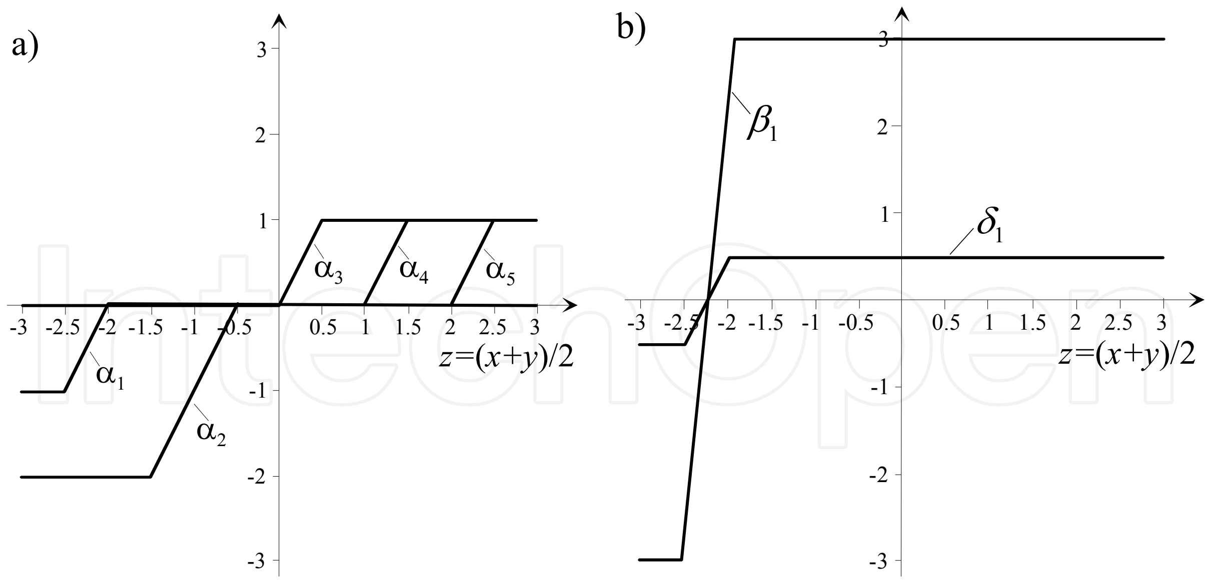



Fig. 22. a) Five components of the function $f_1(z)$; b) Representation of the function $\alpha_1(z)$ .

The function $\beta_1(z)$ in Fig. 22(b) can be implemented as $\beta_1(z) = -S(k_1 \cdot z + a_1)$. For $z = -2.25$, $\beta_1 = 0$ then $a_1 = 2.25k_1$. Taken into account that $k_1 = 6/0.5 = 12$, we receive $a_1 = 27$ and

$$\beta_1(x,y) = -S(6x+6y+27)\ ,\ \delta_1(x,y) = \tfrac{1}{6}\beta_1(x,y)\ .$$

Finally

$$\alpha_1(x,y) = \delta_1(x,y) - 0.5 = \tfrac{1}{6}\cdot\beta_1(x,y) - 0.5\ . \tag{35}$$

In the same way it is possible to find

$$\begin{aligned}
\alpha_2(x,y) &= \tfrac{1}{3}\beta_2(x,y) - 1 = \ -\tfrac{1}{3}S(3x+3y+6) - 1;\\
\alpha_3(x,y) &= \tfrac{1}{6}\beta_3(x,y) + 0.5 = -\tfrac{1}{6}S(6x+6y-3) + 0.5;\\
\alpha_4(x,y) &= \tfrac{1}{6}\beta_4(x,y) + 0.5 = \ -\tfrac{1}{6}S(6x+6y-15) + 0.5;\\
\alpha_5(x,y) &= \tfrac{1}{6}\beta_5(x,y) + 0.5 = \ -\tfrac{1}{6}S(6x+\cdot y-27) + 0.5.
\end{aligned} \tag{36}$$

The function $f_1(z)$ can be calculated in accordance with (34) on one summing amplifier. Finally taking into account mutual compensation of constants, we have

$$f_1(x,y) = S\{-\tfrac{1}{6}\cdot[\beta_1(x,y) + 2\cdot\beta_2(x,y) + \beta_3(x,y) + \beta_4(x,y) + \beta_5(x,y)]\}. \tag{37}$$

Thus, the implementation of the function $f_1(x, y)$, which represents the matrix $M_1$, consists of six summing amplifiers.

### 4.2 Extracting a matrix with elements separated by a line

This method is applicable for realization of matrices composed from two types of elements, which can be separated with a line. After extracting the symmetrical component the residual matrix is $M_2$.

$$\overset{M_2}{\begin{vmatrix} \le 0 & \le 0 & 0 & +1 & +1 & +3 & +3 \\ \le 0 & 0 & +1 & 0 & 0 & +3 & \ge 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & \ge 2 \\ 0 & 0 & 0 & 0 & 0 & +1 & \ge 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & \ge 1 \\ 0 & 0 & 0 & +1 & 0 & 0 & \ge 0 \\ +1 & 0 & +1 & 0 & 0 & \ge 0 & \ge 0 \end{vmatrix}} = \overset{M_3}{\begin{vmatrix} 0 & 0 & 0 & 0 & 0 & +3 & +3 \\ 0 & 0 & 0 & 0 & 0 & +3 & +3 \\ 0 & 0 & 0 & 0 & 0 & 0 & +3 \\ 0 & 0 & 0 & 0 & 0 & 0 & +3 \\ 0 & 0 & 0 & 0 & 0 & 0 & +3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{vmatrix}} + \overset{M_4}{\begin{vmatrix} \le 0 & \le 0 & 0 & +1 & +1 & \ge 0 & \ge 0 \\ \le 0 & 0 & +1 & 0 & 0 & \ge 0 & \ge 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \ge 0 \\ 0 & 0 & 0 & 0 & 0 & +1 & \ge 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \ge 0 \\ 0 & 0 & 0 & +1 & 0 & 0 & \ge 0 \\ +1 & 0 & +1 & 0 & 0 & \ge 0 & \ge 0 \end{vmatrix}}$$

This matrix has some elements of types $\le a$ and $\ge b$. This means that instead of values $a$ and $b$ of the elements it is possible to substitute any logical value less than $a$ and more than $b$ respectively. Let us split the matrix $M_2$ in two matrices ($M_3$ and $M_4$) and try to implement the matrix $M_3$. The matrix $M_4$ is a new residual matrix.

Let us address to Fig. 23. It is easy to see that the matrix $M_3$ consists of elements with two different values, which can be separated with help of two parallel lines: $x - 3y + 8 = 0$ and $x - 3y + 8 = 1$. A new variable is introduced

$$w = x - 3y + 8 \,. \tag{38}$$

Value of the variable $w$ in the point with coordinates $(x, y)$ is proportional to the distance of this point from the line. In all points lying on and up of the line, $w \le 0$, and in all points lying on and down of the dashed line ($x - 3y + 7 = 0$), $w \ge 1$.

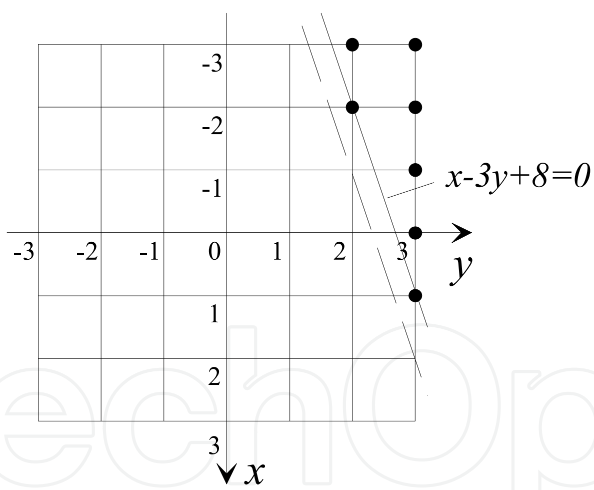



Fig. 23. Separating valuable elements of the matrix $M_3$.

It is easy to see, that the matrix $M_3$ representing the function $f_2(x, y)$ can be implemented as

$$f_2(x, y) = S\{S[3(-x + 3y - 8)] - 3\} \,. \tag{39}$$

In this implementation all valuable matrix elements are equal to "3".

### 4.3 Extracting a matrix with rectangular configuration of valuable elements

Let us introduce a *Pyramid Function* that is the function, which corresponds to a matrix with a single valuable element and represents a rule of the type

$$\text{if } (x=a)\,\&\,(y=b) \text{ then } f(x,y)=c \text{ else } f(x,y)=0. \tag{40}$$

This function is shown in Fig. 24.

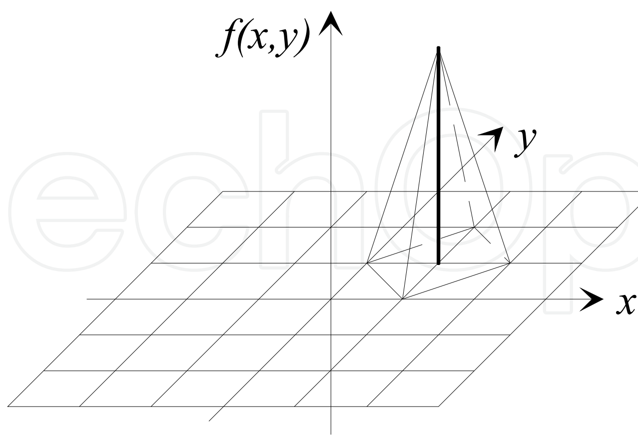



Fig. 24. A pyramid function.

The *Pyramid Function* has some fixed value $c$ ($-k \le c \ne 0 \le +k$) at the point ($a$, $b$) and at the rest of the space bordered by points neighboring to ($a$, $b$) this function is zero. The transition from $c$ to zero is linear. Neighborhood is defined by coordinate increments $\Delta_x = \pm 1$ and $\Delta_y = \pm 1$.
Let's turn to Fig. 25 to construct the pyramid function.

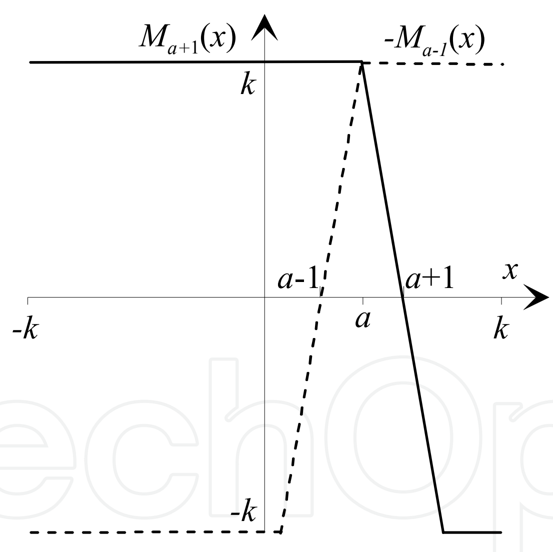



Fig. 25. Component functions of the pyramid projection onto the flat $y = b$; $c = k$.

The figure shows two component mask-functions $M_{a+1}(x)$ and $-M_{a-1}(x)$ those are implemented for ($2k$+1)-valued logic ($-k \le x \le +k$) as:

$$\begin{aligned} M_{a+1}(x) &= S[k \cdot (x - a - 1)], \\ -M_{a-1}(x) &= S[k \cdot (-x + a - 1)]. \end{aligned} \tag{41}$$

Similarly component functions of the pyramid projection onto the flat $x = a$ for the case $c = k$ can be constructed as:

$$\begin{aligned} M_{b+1}(y) &= S[k \cdot (y - b - 1)], \\ -M_{b-1}(y) &= S[k \cdot (-y + b - 1)]. \end{aligned} \tag{42}$$

It is easy to check that the function

$$\gamma(x) = S[M_{a+1}(x) - M_{a-1}(x) - 2k] \tag{43}$$

has the form shown in Fig. 26; $\gamma(x)$ equals to "0" when $x = a$ and equals to "+$k$" for all other integer argument values.

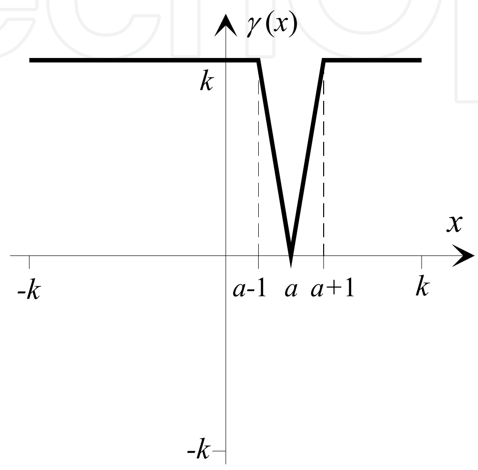



Fig. 26. Graph of the function $\gamma(x)$.

In a similar manner, we can construct the following function of 2 variables:

$$\gamma(x, y) = S[M_{a+1}(x) - M_{a-1}(x) + M_{b+1}(y) - M_{b-1}(y) - 4k]. \tag{44}$$

The function $\gamma(x, y)$ equals to "0" when $(x = a)\&(y = b)$ and equals to "+$k$ " for all other matrix points.
Now it is easy to construct the pyramid function with height "c" presented in Fig. 25:

$$f(x, y) = \frac{c}{k} S[\gamma(x, y) - k]. \tag{45}$$

A pyramid of an arbitrary height is obtained by simple input gain factor scaling of the next amplifier. The pyramid sign can be elementarily changed at the stage of component mask-function constructions.
We anticipate some complications in the case when it is needed to receive good "sewing" pyramids with already implemented functions. The pyramid function $f(x, y)$ (45) of Fig. 24 type has intersections with the flats $y = \{b, b \pm 5, b \pm 1\}$ and with the flats $x = \{a, a \pm 0.5, a \pm 1\}$ shown in Fig. 27.
When "sewing" a pyramid function with other functions to get monotonous piece-linear approximation between adjacent logical values the view of the pyramid function by each of its coordinate can be changed to one of variants shown in Fig. 29.
To construct a pyramid function with graphs by coordinates $x$ and $y$ of the Fig. 28(a) type (center trapeze), it is sufficient to substitute in the functions (41) – (45) instead of original variables $x, y$ new variables $z = x + y$, $w = x - y$ and instead of points $a, b$ points $c = a + b$, $d = a - b$ respectively.

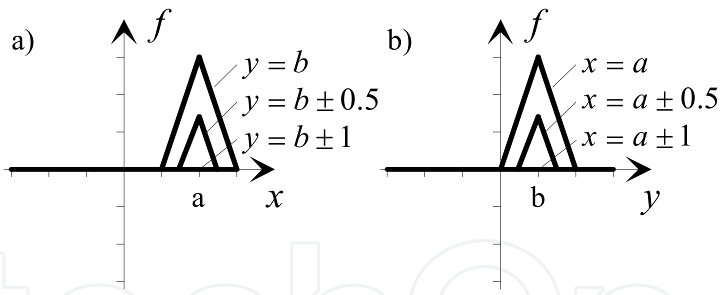



Fig. 27. Graph of the pyramid function a) by coordinate $x$ and b) by coordinate $y$.

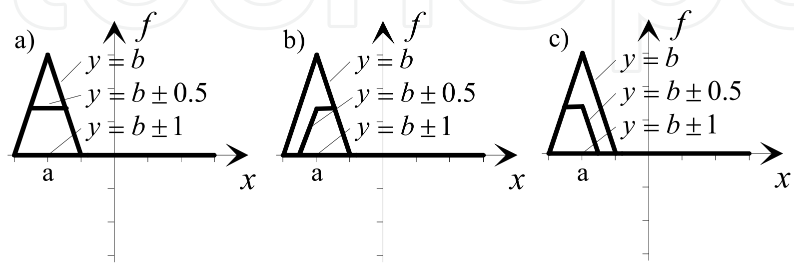



Fig. 28. Possible graphs of a pyramid function by one of coordinates: a) center trapeze, b) right trapeze, c) left trapeze.

It means the transition to the pyramid function shown in Fig. 29, which is implemented by analogy with (41) – (45) as

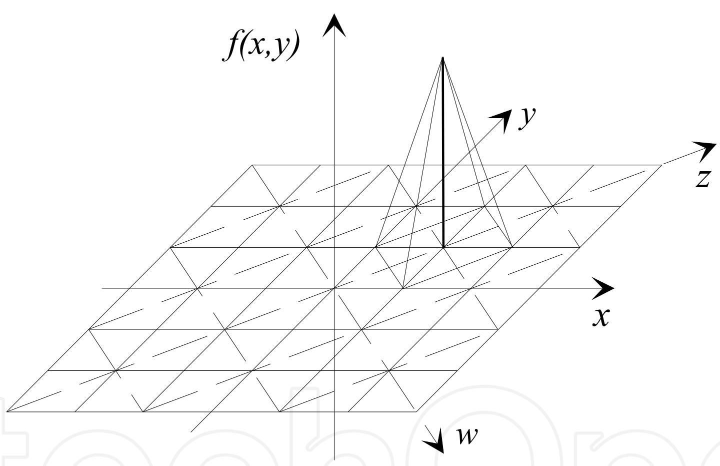



Fig. 29. A pyramid in coordinates $z = x + y$ and $w = x - y$.

$$f(x,y) = S\{S[S(k(x+y-a-b-1)) + S(k(-x-y+a+b-1)) + S(k(x-y-a+b-1)) + S(k(-x+y+a-b-1)) - 4k] - k\}. \tag{46}$$

For implementing a function that has graphs along one of its variables (e.g. $x$) of the right trapeze type (Fig. 28(b)), let us introduce two intermediate functions $\phi\,[x,\xi(y)]$ and $\psi\,[x,\xi(y)]$ shown in Fig.30.

It is easy to check that these intermediate functions can be implemented as

$$\begin{aligned} \varphi[x,\xi(y)] &= S[M_{a-1}(x) + 2k - \xi(y)], \\ \psi[x,\xi(y)] &= S[-M_a(x) - 2k - \xi(y)]. \end{aligned} \tag{47}$$

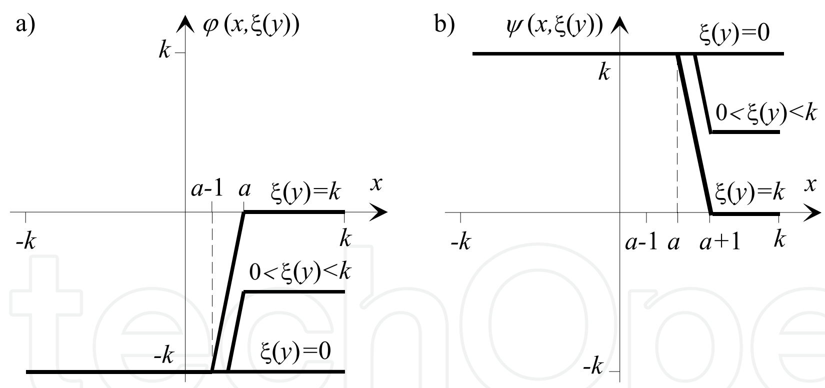



Fig. 30. Intermediate functions a) $\varphi\ [x,\xi(y)]$, b) $\psi\ [x,\xi(y)]$.

The function $f(x,y)=\varphi[x,\xi(y)]+\psi[x,\xi(y)]$ has the right-hand trapeze form along the axis $x$ (Fig. 28(b)). If the function $\xi(y)$ has a triangle form, $\xi(b)=\pm k$, and $\xi(y<b-1)=\xi(y>b+1)=0$, the function $f(x, y)$ is a pyramid function.

For implementing a function that has graphs along the axis $x$ of the left trapeze type (Fig. 28(c)), two intermediate functions $\varphi[x,\xi(y)]$ and $\psi\ [x,\xi(y)]$ shown in Fig. 31 are introduced.

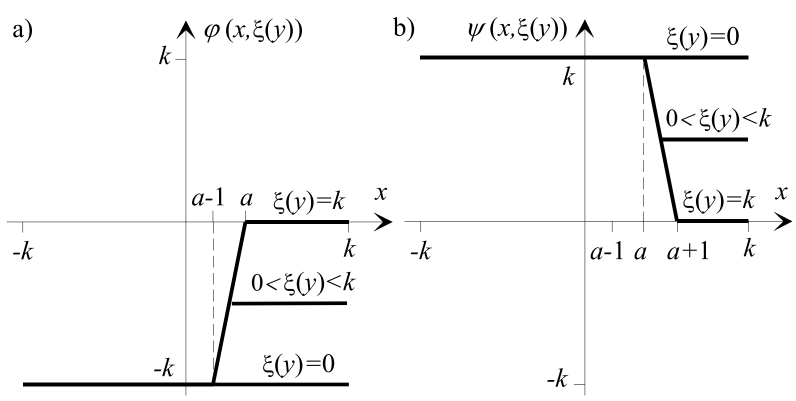



Fig. 31. Intermediate functions a) $\varphi[x,\xi(y)]$, b) $\psi\ [x,\xi(y)]$.

These functions are implemented as

$$\varphi[x,\xi(y)]=S[M_a(x)+2k-\xi(y)]-2k+\xi(y),$$
$$\psi[x,\xi(y)]=S[-M_{a+1}(x)-2k-\xi(y)]+2k-\xi(y).$$

Then the function

$$f(x,y)=\varphi[x,\xi(y)]+\psi[x,\xi(y)]=S[M_a(x)$$
$$+2k-\xi(y)]+S[-M_{a+1}(x)-2k-\xi(y)] \tag{48}$$

has the form of the left trapeze type along the axis $x$ shown in fig. 28(c).

The pyramid function approach is not limited to rules with point condition (40) and may be extended to rules with interval condition of the type

$$\text{if } (a_1 \le x \le a_2 \ \text{ and } \ b_1 \le y \le b_2) \ \text{ then } f(x,y)=k \ \text{ else } f(x,y)=0. \tag{49}$$

Interval conditions can be implemented by simple changing constants in mask-functions. The rule (49) represents matrices with rectangular configurations of valuable elements. Such matrices can be implemented as truncated pyramids.
Note that the function similar to (44) may be constructed for an arbitrary number of variables. Implementation of a pyramid function of two variables requires 6 amplifiers. Introducing each additional variable requires two additional amplifiers.

### 4.4 Extracting a matrix with valuable elements laying on a diagonal

Let us split the matrix $M_4$ on two matrices ($M_5$ and $M_6$) and try to implement the matrix $M_5$. Matrix $M_6$ is the next residual matrix.

$$\overset{M_4}{\begin{vmatrix} \le 0 & \le 0 & 0 & +1 & +1 & \ge 0 & \ge 0 \\ \le 0 & 0 & +1 & 0 & 0 & \ge 0 & \ge 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \ge 0 \\ 0 & 0 & 0 & 0 & 0 & +1 & \ge 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \ge 0 \\ 0 & 0 & 0 & +1 & 0 & 0 & \ge 0 \\ +1 & 0 & +1 & 0 & 0 & \ge 0 & \ge 0 \end{vmatrix}} = \overset{M_5}{\begin{vmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & +1 \\ 0 & 0 & 0 & 0 & 0 & +1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & +1 & 0 & 0 & 0 \\ 0 & 0 & +1 & 0 & 0 & 0 & 0 \end{vmatrix}} + \overset{M_6}{\begin{vmatrix} \le 0 & \le 0 & 0 & +1 & +1 & \ge 0 & \ge 0 \\ \le 0 & 0 & +1 & 0 & 0 & \ge 0 & \ge 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \ge 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \ge 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \ge 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \ge 0 \\ +1 & 0 & 0 & 0 & 0 & \ge 0 & \ge 0 \end{vmatrix}}$$

In its turn, the matrix $M_5$ can be composed from two matrices

$$\frac{1}{3}\overset{M_{51}}{\begin{vmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & +3 \\ 0 & 0 & 0 & 0 & 0 & +3 & 0 \\ 0 & 0 & 0 & 0 & +3 & 0 & 0 \\ 0 & 0 & 0 & +3 & 0 & 0 & 0 \\ 0 & 0 & +3 & 0 & 0 & 0 & 0 \end{vmatrix}} + \frac{1}{3}\overset{M_{52}}{\begin{vmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{vmatrix}}$$

and represented as the sum of corresponding functions

$$f_3(x,y) = \tfrac{1}{3}[f_{31}(x,y) + f_{32}(x,y)]\,.$$

In the matrix $M_{51}$, elements with the value "+3" lay on the line $x + y - 2 = 0$. This matrix can be described by the function of one variable $f_{31}(z = x + y)$, which is defined by the rule

$$\text{if } z = 2 \text{ then } f_{31}(z) = 3 \text{ else } f_{31}(z) = 0.$$

The function $f_{31}(z)$ can be constructed as it is shown in Fig. 32.
It is easy to see from this figure that

$$f_{31}(z) = \alpha_6(z) + 3,\quad \alpha_6(z) = S[\beta_6(z) + \delta_6(z) + 6],\quad \beta_6(z) = S(3z - 3),\quad \delta_6(z) = S(-3z + 9).$$

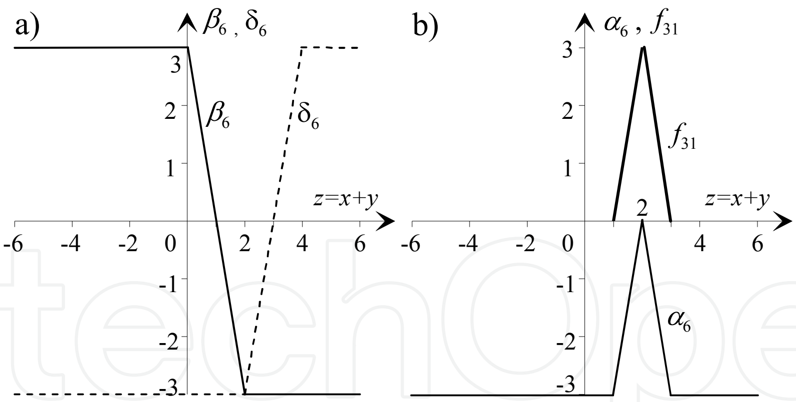



Fig. 32. Functional representations of the matrixes $M_5$.

For good "sewing" the function $f_{32}$ with $f_{31}$ and the function $f_3$ with $f_1$ the function $f_{32}$ has to be implemented as pyramid function of variables $z = (x + y)/2$ and $w = (x - y)/2$ as it is shown in Fig. 29. Substitution of the variables $z$ and $w$ in formula (46) instead of $x + y$ and $x - y$ respectively, $a = +1$, $b = +1$, $k = 3$, and changing the sign of the function gives

$$f_{32}(z,w) = S\{S[\beta_6(z) + \delta_6(z) + \beta_7(w) + \delta_7(w) + 12] + 3\}$$

where $\beta_6(z)$, $\delta_6(z)$ are already implemented and

$$\beta_7(w) = S(3w + 3), \quad \delta_7(w) = S(-3w + 3).$$

Finally, the function $f_3(x, y)$ corresponding to the matrix $M_5$ can be implemented as

$$f_3(x,y) = \tfrac{1}{3} S\{\beta_6(x,y) + \delta_6(x,y) + S[\beta_6(x,y) + \delta_6(x,y) + \beta_7(x,y) + \delta_7(x,y) + 12] + 9\} + 1 \quad (50)$$

where

$$\beta_6(x,y) = S(3x + 3y - 3), \quad \delta_6(x,y) = S(-3x - 3y + 9),$$
$$\beta_7(x,y) = S(3x - 3y + 3), \quad \delta_7(-3x + 3y + 3).$$

### 4.5 Implementation of the matrix *M*6

Let us split the matrix $M_6$ on two matrices ($M_7$ and $M_8$) and try to implement the matrix $M_7$. The new residual matrix is $M_8$, all elements of which are defined.
The matrix $M_7$

$$\begin{array}{ccccc} M_6 & & M_7 & & M_8 \\ \begin{vmatrix} \le 0 & \le 0 & 0 & +1 & +1 & \ge 0 & \ge 0 \\ \le 0 & 0 & +1 & 0 & 0 & \ge 0 & \ge 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \ge 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \ge 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \ge 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \ge 0 \\ +1 & 0 & 0 & 0 & 0 & \ge 0 & \ge 0 \end{vmatrix} & = & \begin{vmatrix} 0 & 0 & 0 & +1 & +1 & +1 & +1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{vmatrix} & + & \begin{vmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & +1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ +1 & 0 & 0 & 0 & 0 & 0 & 0 \end{vmatrix} \end{array}$$

has one rectangular component and can be represented as the function $f_4(x, y)$ that is defined by the rule

$$\text{if } (x = -3)\&(y \geq 0) \text{ then } f_4(x,y) = 1 \text{ else } f_4(x,y) = 0.$$

By analogy with constructing formulas (41) – (45) let us compose two auxiliary functions: $\gamma_1(x)$ for the condition $x = -3$ and $\gamma_1(y)$ for the condition $y \geq 0$. These functions are represented in Fig. 33 and can be constructed as

$$\gamma_1(x) = S[-M_{-3}(x)], \quad \gamma_1(y) = S[M_{-1}(y) + 3].$$

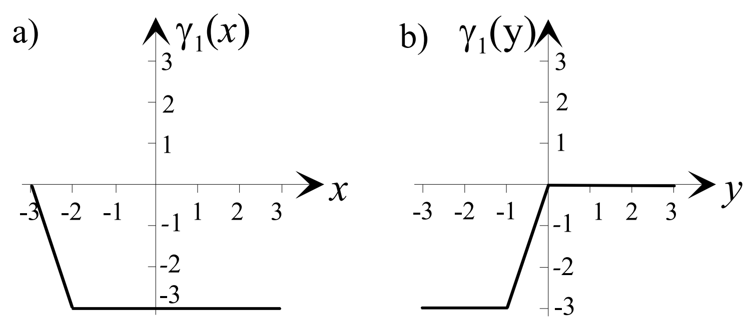



Fig. 33. Two auxiliary functions: a) $\gamma_1(x)$ for the condition $x = -3$ and
b) $\gamma_1(y)$ for the condition $y \geq 0$ .

Based on the functions $\gamma_1(x)$ and $\gamma_1(y)$ it is possible to construct the function of two variables

$$\gamma_1(x,y) = S[-M_{-3}(x) + M_{-1}(y) + 3].$$

Taking into account that for $k = 3$

$$-M_{-3}(x) = S(-3x - 9), \quad M_{-1}(y) = S(3y + 3)$$

it is not difficult to find

$$f_4(x,y) = \tfrac{1}{3}S[S(-3x - 9) + S(3y + 3) + 3] + 1\,. \tag{51}$$

The residual matrix $M_8$ has only two nonzero elements with coordinates ($x$ = +3, $y$ = –3) and ($x$ = –2, $y$ = –1). Let us designate the functions, with help of which the controller function can be corrected in these points, as $f_5$ and $f_6$ respectively. Depending on coordinates of a valuable element and conditions of good "sewing" its function with already implemented fragments different implementation methods can be used.

The function $f_5(x, y)$ is equal to 0 everywhere except the point ($x$ = +3, $y$ = –3), at which $f_5(x, y) = 1$. For monotonic piecewise-linear connection of this function with the function $f_1(x, y)$, graphs of the function $f_5(x, y)$ along each of its arguments must have the form of Fig. 28(a) or (b). The function $f_5(x, y)$ can be implemented as pyramid function in accordance with (46) but the following approach, which is shown in Fig. 34, gives better implementation.

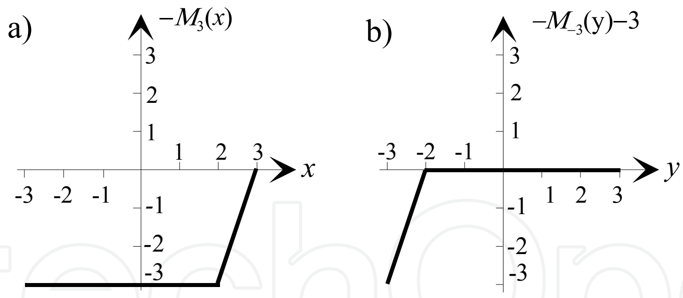



Fig. 34. Auxiliary functions: a) $-M_3(x)$ and b) $-M_3(y)-3$.

Now it is not difficult to construct the function

$$f_5(x,y) = \tfrac{1}{3}\{S[-M_{-3}(y) - 3 - M_3(x)] - M_3(x)\}$$

and finally the function $f_5$ is implemented as

$$f_5(x,y) = \tfrac{1}{3}S[S(-3y-9) + S(-3x+9) - 3] + \tfrac{1}{3}S(-3x+9). \quad (52)$$

As experiments showed, the monotonic piecewise-linear approximation between the logical level in the point (-2,-1) and logical levels in the adjacent points of the functions $f_1(x, y)$ and $f_4(x, y)$ will be obtained, if the pyramid function $f_6$ in the point (-2,-1) is implemented in accordance with the formula (46).

$$f_6(x,y) = \tfrac{1}{3}S\{S[S(3x+3y+6) + S(-3x-3y-12) + S(3x-3y) + S(-3x+3y-6) - 12] - 3\}.$$

After some transformations providing the possibility to save one summing amplifier this function looks as follows

$$f_6(x,y) = \tfrac{1}{3}S[S(-3x-3y-6) + S(3x+3y+12) + S(-3x+3y) + S(3x-3y+6) + 12] + 1. \quad (53)$$

**4.6 Implementation of the controller**

Correctness of the designed controller and its functioning has been checked with SPICE simulation (MSIM 8). In simulation experiments MOSIS BSIM3v3.1 level 7 models of 0.4μm transistors have been used.

As building blocks for the controller circuit two types of summing amplifiers were used (ordinary and powerful). They are built on the basis of three-stage push-pull CMOS operational amplifier[2] with 1.5-MegOhm resistor in the feedback. Examples of such summing amplifiers are shown in Fig. 35. Transistors in this figure are marked with two numbers, which designate transistor dimensions (length and width).

[2]The operational amplifier of this type ii the simplest and is chosen only with the purpose of simplifying SPUCE simulation.

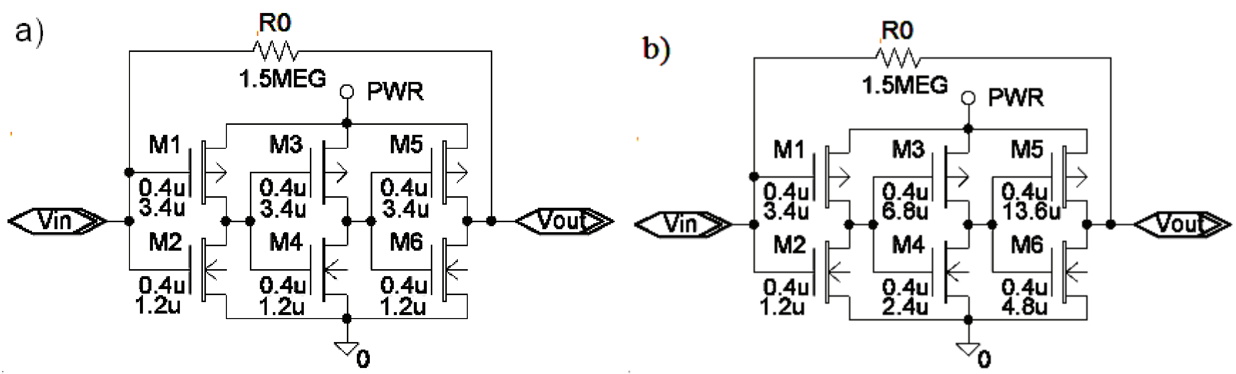



Fig. 35. Operational amplifier with feedback: a) ordinary, b) powerful.

The controller schematic for experiments is represented in Fig. 36. Two powerful elements PHS1 and PHS2 of the controller produce signals -*y* and -*x* respectively. Other elements are ordinary.

Analytical description of the controller schematic can be derived on bases of the function implementations (35) – (37), (39), (50) – (53) and has the following form

$$
\begin{aligned}
f_1(x,y) &= S_9[\tfrac{1}{6}S_3(6x+6y+27) + \tfrac{1}{3}S_4(3x+3y+6) + \tfrac{1}{6}S_5(6x+6y-3) + \\
&\qquad \tfrac{1}{6}S_6(6x+6y-15) + \tfrac{1}{6}S_7(6x+6y-27)]; \\
f_2(x,y) &= S_9(S_8(-3x+9y-24)-3); \\
f_3(x,y) &= \tfrac{1}{3}S_{12}\{S_{10}(3x+3y-3) + S_{11}(-3x-3y+9) + S_{15}[S_{10}(3x+3y-3) + \\
&\qquad S_{11}(-3x-3y+9) + S_{14}(3x-3y+3) + S_{13}(-3x+3y+3) + 12] + 9\} + 1; \\
f_4(x,y) &= \tfrac{1}{3}S_{18}[S_{16}(-3x-9) + S_{17}(3y+3) + 3] + 1; \\
f_5(x,y) &= \tfrac{1}{3}S_{21}[S_{19}(-3x+9) + S_{20}(-3y-9) - 3] + \tfrac{1}{3}S_{19}(-3x-9); \\
f_6(x,y) &= \tfrac{1}{3}S_{26}[S_{23}(-3x-3y-6) + S_{22}(3x+3y+12) + S_{25}(-3x+3y) + \\
&\qquad S_{24}(3x-3y+6) + 12] + 1; \\
F(x,y) &= \sum_{j=1}^{6} f_j(x,y) = S_{28}[S_{27}(S_9 + \tfrac{1}{3}S_{12} + \tfrac{1}{3}S_{18} + \tfrac{1}{3}S_{19} + \tfrac{1}{3}S_{21} + \tfrac{1}{3}S_{26} + 3)].
\end{aligned}
$$

Enumeration of summing amplifiers in this description corresponds to enumeration of elements in the controller circuit. The controller contains 28 amplifiers and 86 resistors. Resistor values have been calculated as $R_j = R_0 / w_j$ where $w_j$ is logical weight of the $j^{th}$ element input signal.

In experiments, source voltage was 3.5V. Input variable *x* changed linearly from 0V to 3.5V, input variable *y* changed discreetly and kept constant value within one cycle of *x* changing.

The voltage range was evenly divided onto seven logical levels so that the logical levels "-3" and "+3" corresponded to voltages $V_{gnd}$ and $V_{dd}$ respectively.

Results of SPICE simulation of the controller schematic are represented in Fig. 37. This figure has been constructed by using GNUplot and illustrates the response surface in the coordinates *X*, *Y*. Analyzing the surface it is possible to conclude that the functioning of the controller is correct because of logical values of the circuit output depend on the logical values of the input variables in accordance with the Table 11.

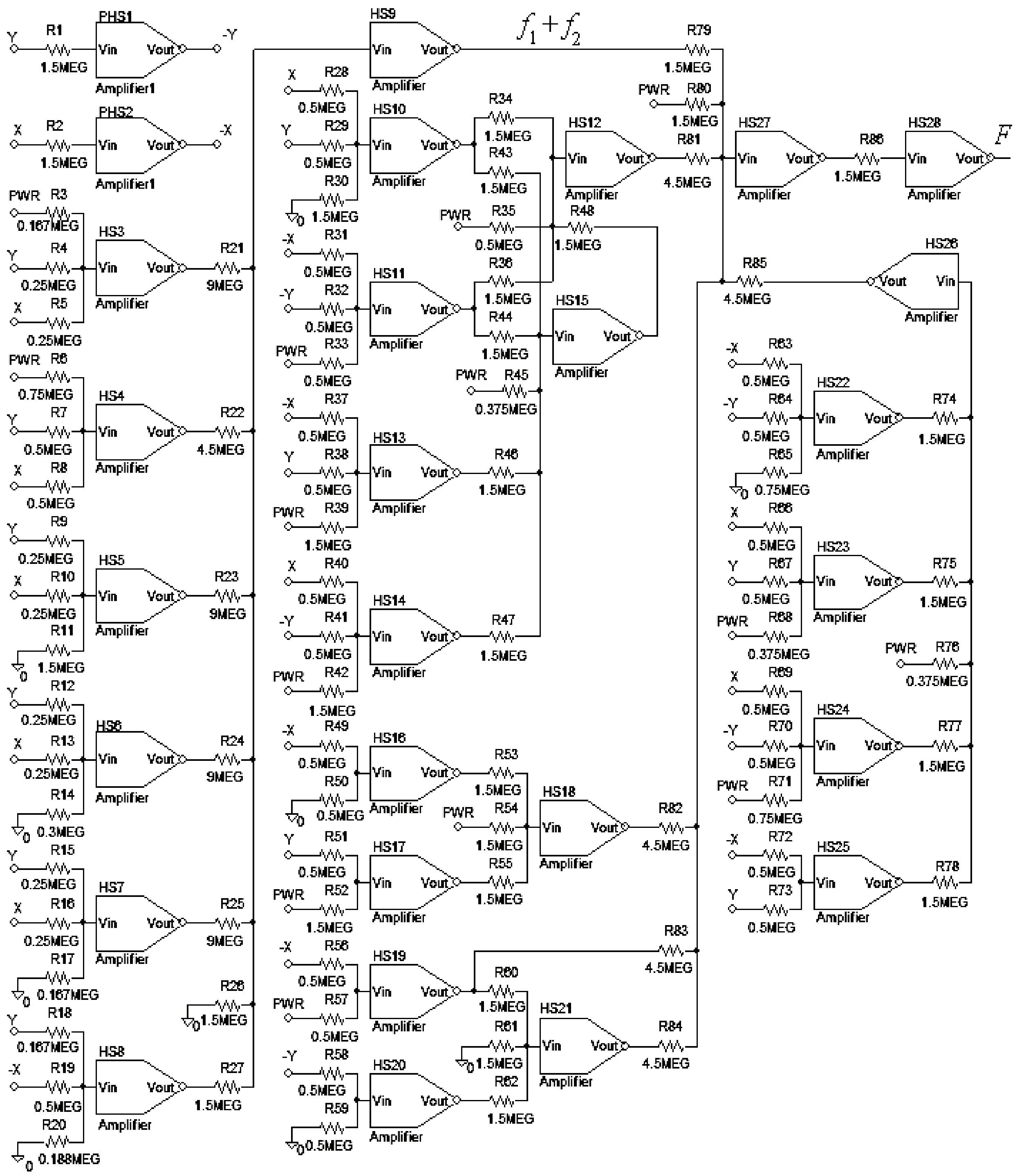



Fig. 36. The controller schematic for experiment.

Moreover, the controller output signal has monotonic piecewise-linear approximation between adjacent logical levels. Thus the designed controller can be used as an analog device, which has analog inputs and produces an analog output signal.

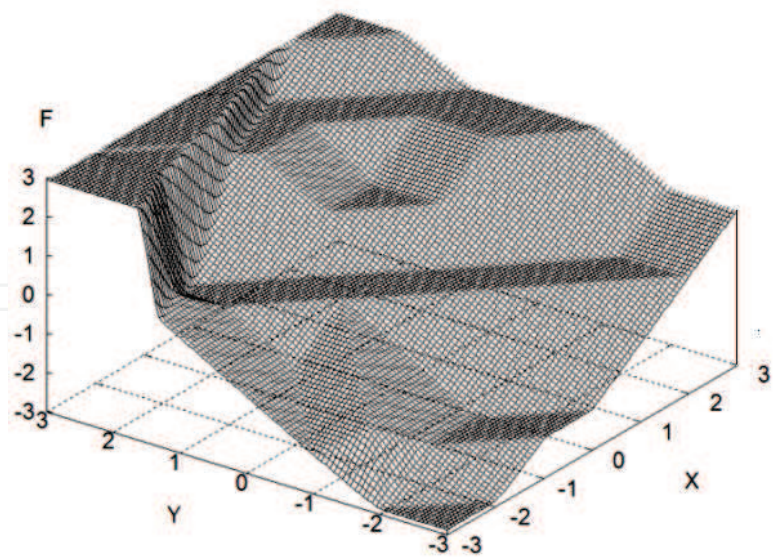



Fig. 37. The response surface of the controller.

## 5. Transformation of analog signals into multi-valued logic variables

In previous sections functions of multi-valued logic were specified by tables of fuzzy rules over linguistic variables. It was implicitly assumed that values of analog signals, which corresponded to linguistic variables, were evenly distributed in the range of voltages representing analog signals. Otherwise by artificial means the number of linguistic variables can be increased that leads to growing the implementation complexity.

In this section a procedure of transforming input analog variables into multiple-valued logic variables with evenly distributed logical levels is suggested. This procedure in some sense is analogous to the procedure of fuzzification in fuzzy controllers. Because of using a fuzzy control description for implementation of controllers as multi-valued logical functions the same term "fuzzification" for the suggested procedure of logical levels equalization for input variables will be used.

The same procedure is supposed to be used when output multi-valued variables with evenly distributed logical levels demand backward transformation to an analog form with not evenly distributed voltages corresponding to logical levels. In this case the term "defuzzification procedure" will be applied.

### 5.1 Fuzzification procedures

Let us examine more attentively the fuzzification procedure for the case of linear membership functions or membership functions, which sufficiently simply can be represented as piecewise-linear, and propose sufficiently simple universal method. Here the standard determination of a membership function is used. The membership function determines the weight of the corresponding linguistic variable $b$ for each value of an analog variable $X$:

$$w_b = F(b, X);\ \ 0 \le w_b \le 1\ .$$

The simplest example of membership functions is given in Fig. 38.

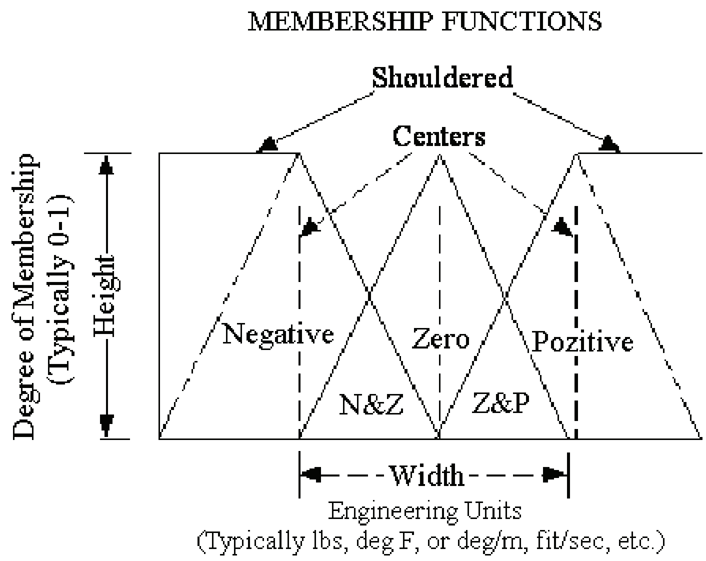



Fig. 38. The simplest type of membership functions.

"The membership function is a graphical representation of the magnitude of participation of each input. It associates a weighting with each of the inputs that are processed, define functional overlap between inputs, and ultimately determines an output response. The rules use the input membership values as weighting factors to determine their influence on the fuzzy output sets of the final output conclusion. Once the functions are inferred, scaled, and combined, they are defuzzified into a crisp output which drives the system. There are different memberships functions associated with each input and output response. Some features to note are:

SHAPE - triangular is common, but bell, trapezoidal, haversine and, exponential have been used (More complex functions are possible but require greater computing overhead to implement.);

HEIGHT or magnitude (usually normalized to 1);

WIDTH (of the base of function);

SHOULDERING (locks height at maximum if an outer function. Shouldered functions evaluate as 1.0 past their center);

CENTER points (center of the member function shape);

OVERLAP (N&Z, Z&P, typically about 50% of width but can be less)". [3]

Fig.38 illustrates the features of the triangular membership function, which is used in the following example.

The procedure of fuzzification and constructing corresponding diagram is examined on an example of the Container Crane fuzzy Controller, membership functions for which are given in Fig. 39.[4]

It is assumed, without disrupting the generality of reasoning, that with changing the *angle* within the limits $(-90^\circ \div +90^\circ)$ and the *distance* in the limits (-10 ÷ +30) *yards* the corresponding analog voltages vary within the range $(0 \div 3.5)V$. The source voltage of the controller circuit is also 3.5V.

[3] Citation is taken from "Fuzzy Logic – an Introduction", part 4, by Steven D. Kaehler, http://www.seattlerobotics.org/encoder/mar98/fuz/fl_part4.html

[1] http://www.fuzzytech.com/e/e_a_pdf.html

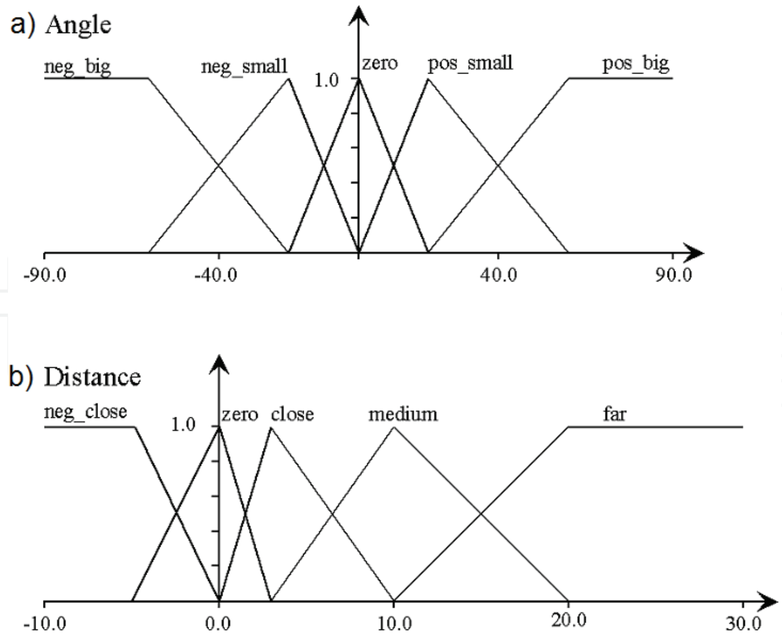



Fig. 39. Membership functions for the Container Crane Fuzzy Controller: a) for the *angle* and b) for the *distance*.

Table 12 determines the function of fuzzification for the piecewise-linear membership functions of the variable *angle* shown in Fig. 39(a). It contains linguistic variables, corresponding to them angle values and voltages, and also logical values evenly distributed within the voltage range. Linearity the membership functions gives the possibility to connect the points of logical values by straight lines. The corresponding fuzzification (equalization) function is given in Fig. 40(a). In this figure the variations of voltages from the average (equilibrium) point of summing amplifier are plotted along the axes.

| neg_big | neg_small | zero | pos_small | pos_big |
|---|---|---|---|---|
| -90°÷-60° (0÷0.58)$V$ | -20° 1.361$V$ | 0° 1.75$V$ | 20° 2.139$V$ | 60°÷90° (2.917÷3.5)$V$ |
| -2 0$V$ | -1 0.875$V$ | 0 1.75$V$ | +1 2.625$V$ | +2 3.5$V$ |

Table 12. Angle membership functions.

For implementation of the function $V_{out} = F_1(V_{in})$ shown in Fig. 40(a) three auxiliary functions should be introduced. These functions are represented in Fig. 40(b). Their sum with saturation on the levels ± 1.75$V$ determines the fuzzificated input function for the controller fuzzy inference part, which, as it has been already proved in previous sections, can be implemented as a multi-valued logic function.
In Fig. 40(b) the angle α and functions $\varphi_j$ (α) are represented in positive and negative voltages. These component functions and the fuzzifier output function $F_1$(α) can be implemented by the following way:

$$\begin{gathered}\varphi_1(\alpha) = -0.5S(4.5\alpha); \quad \varphi_2(\alpha) = -S(1.125\alpha + 2.19); \quad \varphi_3(\alpha) = -S(1.125\alpha - 2.19); \\ F_1(\alpha) = S(-\varphi_1(\alpha) - \varphi_2(\alpha) - \varphi_3(\alpha)) = S(\tfrac{1}{2}S(4.5\alpha) + S(1.125\alpha + 2.19) + S(1.125\alpha - 2.19)).\end{gathered} \quad (54)$$

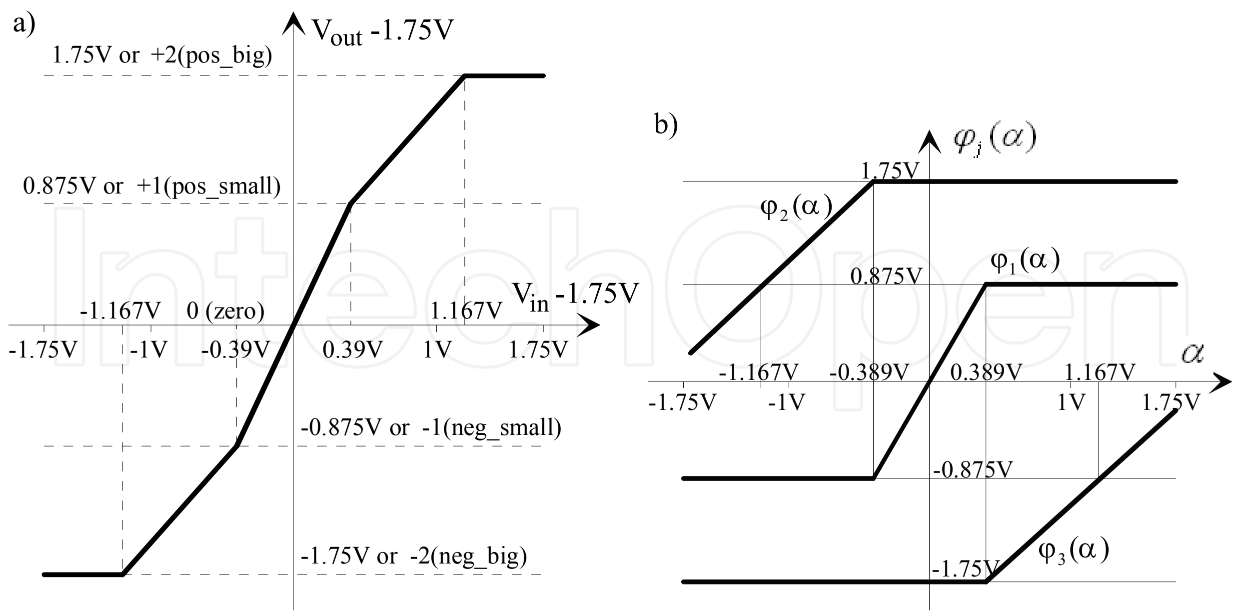



Fig. 40. a) Piecewise-linear function for fuzzification of the variable *angle*;
b) Component functions for the function represented in (a).

Now let us show how to construct and implement the fuzzification function for the input variable *distance*. As can be inferred from Fig. 40(b), the membership functions are characterized, first, by asymmetry of the measured distance ((-10 ÷ +30) yards) and, second, by the explicit asymmetry of the linguistic variable positions along the distance axis. It assumed that the complete range of the measured distance corresponds to the complete range of the supply voltages ($0V \div 3.5V$) or ($-1.75V \div +1.75V$) in deviations from the middle point of amplifiers. For this case, the fuzzification function is determined by Table 13.

| neg_close | zero | close | medium | far |
|---|---|---|---|---|
| ≤ -5 yards ≤ 0.4375$V$ | 0 yards 0.875$V$ | 3 yards 1,1375$V$ | 10 yards 1.75$V$ | ≥ 20 yards ≥ 2.625$V$ |
| -2 0$V$ | -1 0.875$V$ | 0 1.75$V$ | +1 2.625$V$ | +2 3.5$V$ |

Table 13. Distance membership functions

In this table the linguistic variable *close* corresponds to value "log.0" and the linguistic variable *zero* corresponds to the value "log.-1". The balance point of the amplifier input voltage corresponds to linguistic variable *medium*.
Corresponding function $V_{out}(V_{in}) = F_2(V_{in})$ is given in Fig. 41. For implementation of this function it is necessary to realize four auxiliary functions, whose sum with saturation on the levels $\pm 1.75V$ will give the desired result. The auxiliary functions are given in Fig.42. Their values and value of the variable $d$ (*distance*) are represented in negative and positive voltages.
The ways of forming the component functions given in Fig.42 and the function $F_2(d)$ are shown below:

$$\begin{aligned}
&\psi_1(d) = -S(2d+3.5); \quad \psi_2(d) = -0.25S(13.6d+10.14);\\
&\psi_3(d) = -0.25S(5.67d+1.75); \quad \psi_4(d) = -S(d-1.75);\\
&F_2(d) = S(-\psi_1(d)-\psi_2(d)-\psi_3(d)-\psi_4(d)) =\\
&S[S(2d+3.5)+0.25S(13.6d+10.14)+\ 0.25S(5.67d+1.75)+S(d-1.75)].
\end{aligned} \tag{55}$$

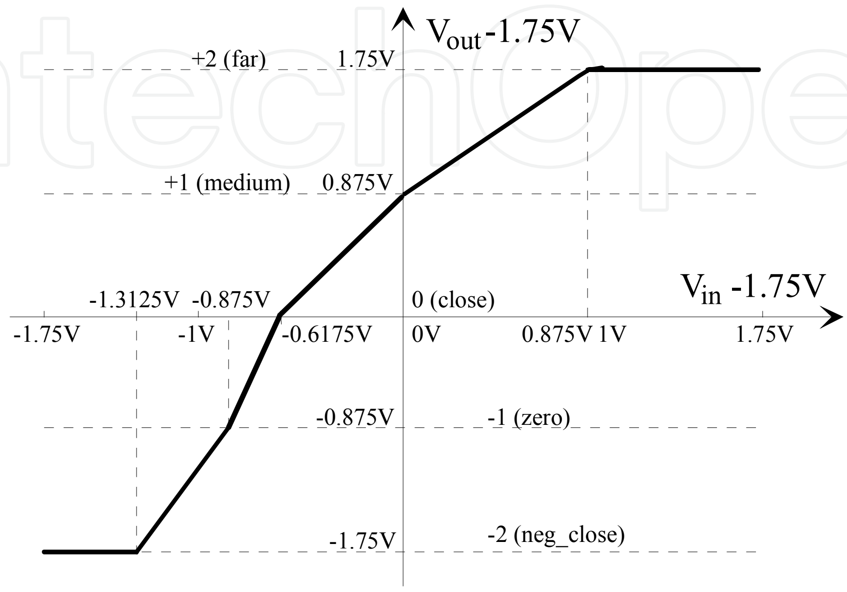



Fig. 41. Piecewise-linear fuzzifications function for the variable *distance*.

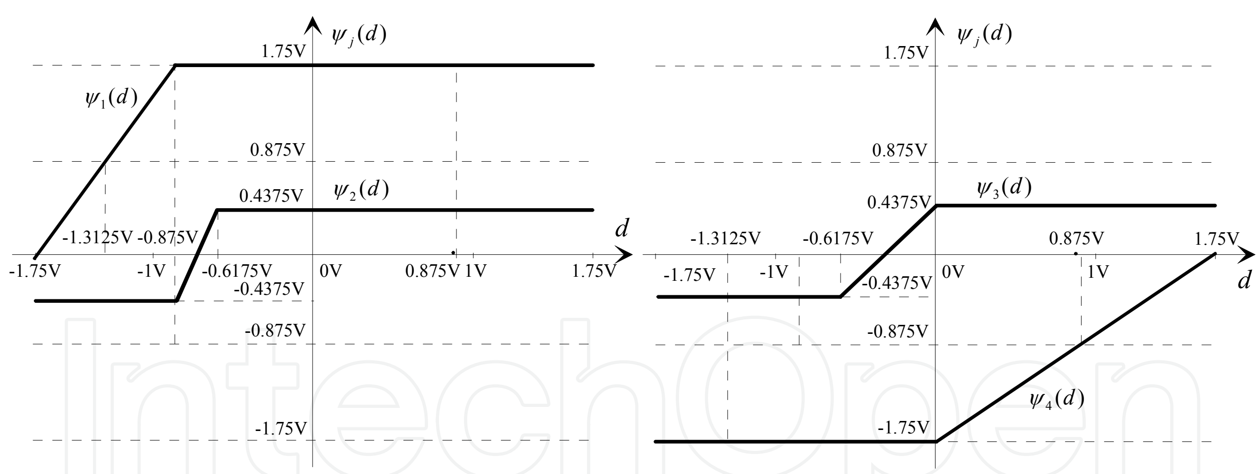



Fig. 42. Component functions for the function represented in Fig.41.

### 5.2 Fuzzifier implementations 5.2

For the completion of the fuzzifier design it only remains to determine the values of the input resistances of summing amplifiers and to conduct SPICE simulation for checking correctness of the implementations (54) and (55). These implementations are represented graphically in Fig.43(a) and Fig.43(b) respectively. Their schematics, which have been used for SPICE simulations, are shown in Fig.44 (a) and Fig.44 (b).

Summing amplifiers used in the schematics are constructed on the bases of three-stage push-pull CMOS operational amplifier in accordance with Fig.35(a).

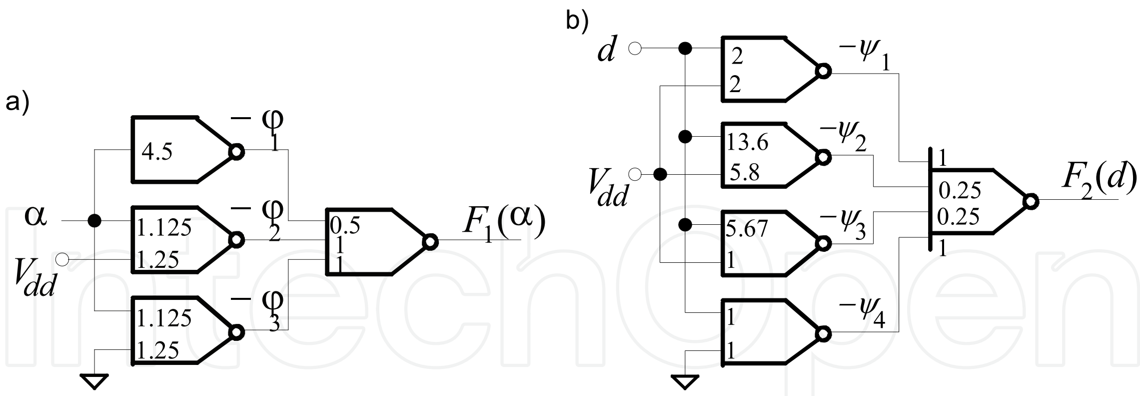



Fig. 43. Fuzzifiers of the variables a) *angle* and b) *distance*.

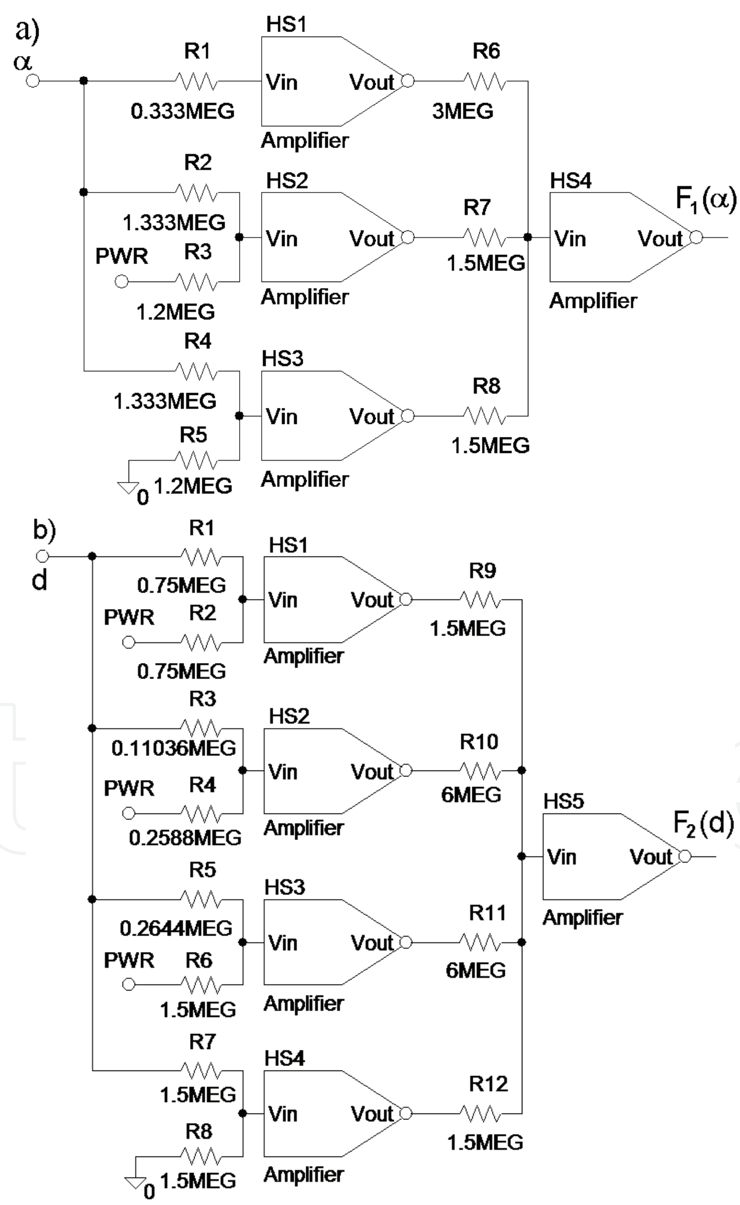



Fig. 44. Fuzzifier schematics a) for the *angle* and b) for the *distance*.

Results of SPICE simulation of the fuzzifiers for variables *angle* and *distance* are shown in Fig. 45.

It is easy to see that the simulation plots are exactly the same as it is required for fuzzification of the input variables *angle* (Fig.40(a)) and *distanc*e (Fig.41). This proves the correctness of the fuzzifier implementations.

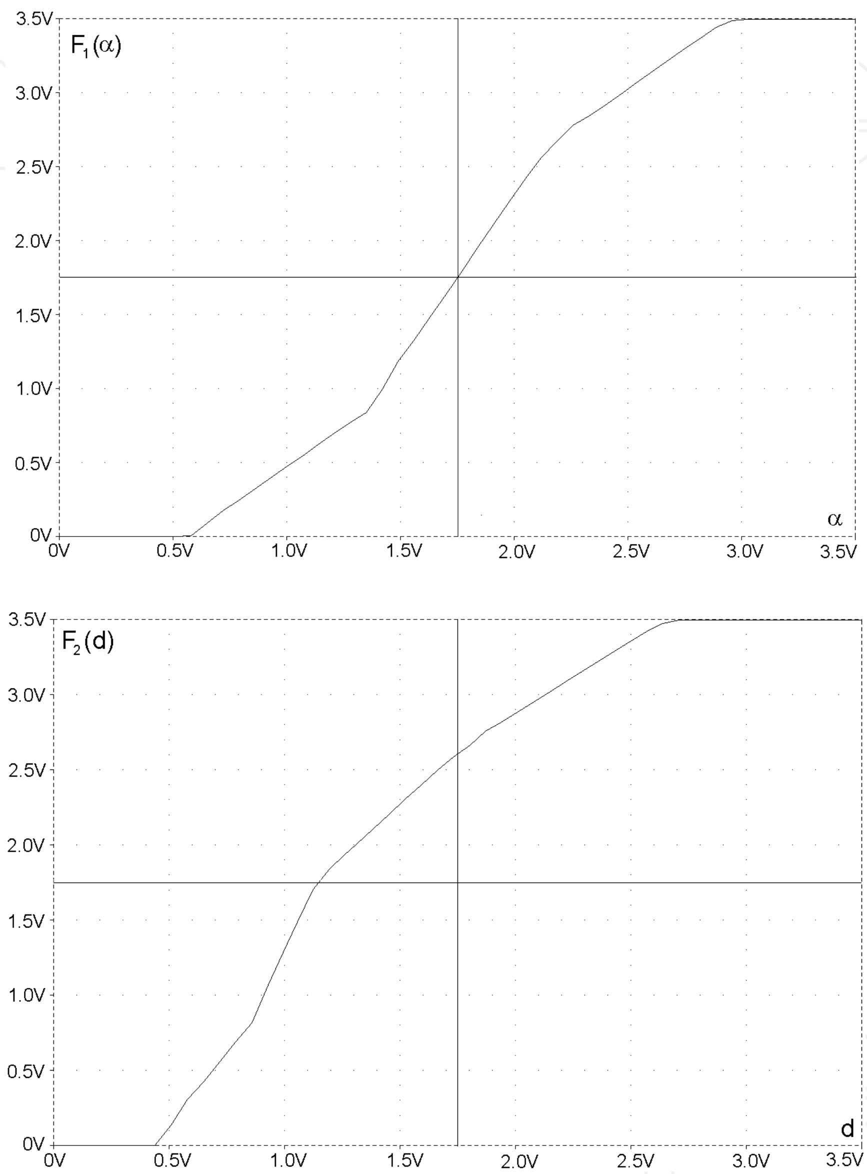



Fig. 45. Outputs of the fuzzifiers (shown in Fig.44) derived by SPICE simulation.

It should be noted that in the case of software implementation of the fuzzification and defuzzification functions, their component functions may be chosen not only piecewise-linear but providing any reasonable approximations.

## 6. Conclusion

Thus, it was shown that all parts of fuzzy controllers can be effectively implemented on bases of summing amplifiers with saturation in accordance with the proposed methodology.

This methodology is oriented to hardware implementation of fuzzy controllers as analog devices. Certainly the traditional approach to implementation of fuzzy controllers provides more accurate control and gives better approximation extracted levels in comparison with suggested. But in many cases our approach gives so simple circuits of controllers that their implementation on the base of standard processors looks rather redundant. Moreover, hardware implementation have advantages of better response time and reliability, low power consumption, smaller die area, etc. It should be noticed that the methodology also admits software implementation of the controllers by means of simulation using the summation operation with restrictions.

In all examples of controllers presented in the paper, the push-pull summing amplifier containing three CMOS invertors is used. Obviously this amplifier circuit is the simplest among operational amplifiers of other types but unfortunately it has the worst characteristics. It was chosen only by two reasons: first, to simplify SPICE simulation of designed controllers and second, to show that using even such primitive and imperfect building block gives rather appropriate characteristics of designed controllers. Certainly in real projects of controllers it is better to use another types of operational amplifiers, e.g., a differential amplifier.

Someone may object that summing amplifiers in all examples of controllers designed with help of suggested methodology contain resistors of large values and it is very difficult to implement these resistors in CMOS VLSI technology. Indeed it is correct. In our case p-well resistors (1-10K Ohms/sq.) or pinch resistors (5-20K Ohms/sq.) can be used. These resistors are compatible with CMOS technology but occupy very large die area, possess bad accuracy, and have big temperature and voltage coefficients. By these reasons the possibility of creating a dynamical model of the summing amplifier with saturation using capacitors instead of resistors has been considered. This consideration gave positive results and perhaps will be published in the future.

The proposed methodology has been applied for designing several devices specified as fuzzy controllers, showed high efficiency and gave very economical implementations. Techniques of synthesizing fuzzy devices in the offered base should get further developing and problems of implementability under the conditions of real production should be resolved in the nearest future.

## 7. References

*TI SPRA028 (1993). Implementation of Fuzzy Logic Selected Applications*, Texas Instruments, January.

Yager, R.R. & Zadeh, L.A. (Ed.) (1992). *An Introduction to Fuzzy Logic Applications in Intelligent Systems*, Kluwer International Series in Engineering and Computer Science, 165, Jan., 356 p.

Klir, G.J.; Yuan B. & Zadeh L.A. (Ed.) (1996). *Fuzzy Sets, Fuzzy Logic, and Fuzzy Systems*, (Advances in Fuzzy Systems - Applications and Theory), World Scientific Pub Co.; Vol. 6, 826 p.

Marks II, R.J. (Ed.) (1994). *Fuzzy Logic Technology and Applications*, IEEE Technology Update Series, Selected Conference Papers, 575 p.

Varshavsky, V.; Marakhovsky, V.; Levin, I. & Kravchenko, N. (2003). Summing Amplifier as a Multi-Valued Logical Element for Fuzzy Control, *WSEAS Transactions on Circuit and Systems*, Issue 3, Vol. 2, pp. 625 – 631.

Varshavsky, V.; Levin, I.; Marakhovsky, V.; Ruderman, A. & Kravchenko, (2004). CMOS Fuzzy Decision Diagram Implementation, *WSEAS Transactions on Systems*, Issue 2, Vol. 3, pp. 615 – 620.

Post E. (1921). Introduction to a general theory of elementary propositions, *Amer. J. Math.*, 43, pp. 163 – 185.

Kandel, A. & Zadeh, L.A. (Ed.) (1993). *Fuzzy Control Systems*, CRC Press LLC, pp. 81 – 86, pp. 168 – 172.

Kimura, K. & Kawawa, T. (1993). *Exhaust Emission Control Device for Internal Combustion Engine*, Toyota Motor Corp., Patent of Japan No. 05-312020.

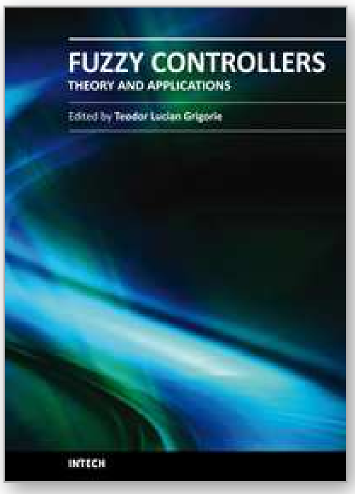
Dualibe, C.; M. Verleysen, M. & Jespers, P.G.A. (2003). *Design of Analog Fuzzy Logic Controllers in CMOS Technologie*s, Kluver Academic Publishers.

Varshavsky, V.; Marakhovsky, V.; Levin, I. & Kravchenko, N. (2004). Functionally Complete Element for Fuzzy Control Hardware Implementation, *47th IEEE Midwest Symposium on Circuits and Systems*, Hiroshima, Japan, Vol. 3, July, pp. 263 – 266.

Varshavsky, V.; Marakhovsky, V.; Levin, I. & Kravchenko, N. (2004). Fuzzy Controller CMOS Implementation, *WSEAS Transactions on Circuits and Systems*, Issue 9, Vol. 3, November, pp.1762 – 1769.

Varshavsky, V.; Marakhovsky, V. & Levin, I. (2005). CMOS Fuzzification Circuits for Linear Membership Functions, *WSEAS Transactions on Systems*, Issue 4, Vol. 4, April, pp. 238 – 243.

Varshavsky, V.; Marakhovsky, V. & Levin, I. (2005). CMOS Fuzzification Circuits for Linear Membership Functions, *WSEAS Transactions on Systems*, Issue 4, Vol. 4, April 2005, pp.238 – 243.




**Fuzzy Controllers, Theory and Applications**
Edited by Dr. Lucian Grigorie

ISBN 978-953-307-543-3
Hard cover, 368 pages
**Publisher** InTech
**Published online** 28, February, 2011
**Published in print edition** February, 2011

Trying to meet the requirements in the field, present book treats different fuzzy control architectures both in terms of the theoretical design and in terms of comparative validation studies in various applications, numerically simulated or experimentally developed. Through the subject matter and through the inter and multidisciplinary content, this book is addressed mainly to the researchers, doctoral students and students interested in developing new applications of intelligent control, but also to the people who want to become familiar with the control concepts based on fuzzy techniques. Bibliographic resources used to perform the work includes books and articles of present interest in the field, published in prestigious journals and publishing houses, and websites dedicated to various applications of fuzzy control. Its structure and the presented studies include the book in the category of those who make a direct connection between theoretical developments and practical applications, thereby constituting a real support for the specialists in artificial intelligence, modelling and control fields.

**How to reference**
In order to correctly reference this scholarly work, feel free to copy and paste the following:

Victor Varshavsky, Viacheslav Marakhovsky, Ilya Levin and Hiroshi Saito (2011). Hardware Implementation of Fuzzy Controllers, Fuzzy Controllers, Theory and Applications, Dr. Lucian Grigorie (Ed.), ISBN: 978-953-307-543-3, InTech, Available from: http://www.intechopen.com/books/fuzzy-controllers-theory-and-applications/hardware-implementation-of-fuzzy-controllers

INTECH
open science | open minds

**InTech Europe**
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

**InTech China**
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

© 2011 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the Creative Commons Attribution-NonCommercial-ShareAlike-3.0 License, which permits use, distribution and reproduction for non-commercial purposes, provided the original is properly cited and derivative works building on this content are distributed under the same license.