FUZZY DECISION DIAGRAM REALIZATION BY ANALOG CMOS SUMMING AMPLIFIERS[1]

V. Varshavsky [1], I. Levin[2], V. Marakhovsky[3], A. Ruderman[4], N. Kravchenko[5]

[1,2,4,5] School of Engineering, Bar Ilan University, ISRAEL

[3] The University of Aizu, JAPAN

ABSTRACT

A functional completeness of summing amplifier with saturation in a multi-valued logic of an arbitrary value proven in previous works gives a theoretical background for analog implementation of fuzzy devices. Practical design techniques for multi-valued analog fuzzy controllers still have to be developed. Compared with the traditional approach, analog CMOS fuzzy controller implementation has the advantages of higher speed, lower power consumption, smaller die area and more. The paper introduces some special design techniques and provides design example for an industrial fuzzy controller implementation solidified by SPICE simulations

1. INTRODUCTION

As shown in [1, 2], a summing amplifier with saturation is a functionally complete element in any multi-valued logics. Thus it may serve a basis for implementing analog fuzzy devices.

The subject of the study is design techniques for analog CMOS circuits that implement fuzzy controller multi-valued functions.

Let's suppose the odd value logic $(k = 2a + 1$ and $-a \le x_j \le +a)$ and assume that $X = \{x_0, x_1, \cdots, x_{n-1}\}$ is a set of input multi-valued variables and $y = F(X)$ - an output variable. Then for a multi-valued logic function we may build an analog of the Shannon's decomposition in the binary logic:

$$y_i = F_i(X) = \bigcup_{\alpha=-a}^{+a} [\text{if } x_j = \alpha \text{ then } y = F(x_j = \alpha, X \backslash x_j)] \tag{1}$$

Equation (1) can be further expanded enabling to build a fuzzy circuit by using variables expulsion method. For this purpose, we need a sub-circuit implementing the function:

Given a basic element realizing (2), we can implement a fuzzy device directly according to the system of fuzzy rules. However, note that equations (1) and (2) represent multi-valued functions in a piece-wise constant manner. An example of a 7-valued function is given in Fig.1,a. Taking into account fuzzification and defuzzification procedures, fuzzy function should be piece-wise linear. Fig.1,b gives an example of such a representation with evenly distributed input/output linguistic variables.

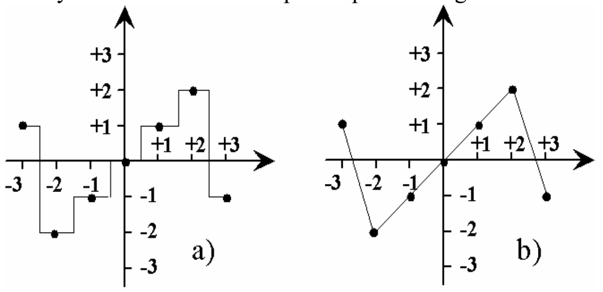

Figure 1. 7-valued function of one variable.

$$\text{if} \quad Z = A \quad \text{then } y = F(Z = A, X \setminus Z) \tag{2}$$

where $Z \subset X$ and A is a value combination of variables Z .

2. MASKING SUMMING AMPLIFIER INPUT

As in [1, 2], we will consider an inverting summing amplifier

$$S(A \cdot X; \beta) = \begin{cases} +a & \text{if} \quad \sum_{i=1}^{n} \alpha_i \cdot x_i + \beta \le -a \\ -\sum_{i=1}^{n} \alpha_i \cdot x_i - \beta & \text{if} \quad +a > \sum_{i=1}^{n} \alpha_i \cdot x_i + \beta > -a \\ -a & \text{if} \quad +a \le \sum_{i=1}^{n} \alpha_i \cdot x_i + \beta \end{cases} \tag{3}$$

where $A = \{\alpha_1, \alpha_2, ..., \alpha_n\}$ – weight coefficients, $X = \{x_1, x_2, ..., x_n\}$ – analog or multi-valued variables, β – offset (shift),
$\pm a$ – saturation values (in the case of k-valued logic, $k = 2a + 1$).
Let us introduce a masking function $M_\alpha(x)$

$$M_\alpha(x) = \begin{cases} +a & \text{if} \quad x \le \alpha - 1 \\ a(\alpha - x) & \text{if} \quad \alpha - 1 < x < \alpha + 1 \\ -a & \text{if} \quad \alpha + 1 < x \end{cases} \tag{4}$$

where $-a \le \alpha \le a$ is a fixed value of the variable *x*. It can be easily seen that as $x = \alpha$ $M_\alpha(\alpha) = 0$. Fig. 2 illustrates an example of the function $M_{-1}(x)$ for $k = 7$.

[1] This research partially was supported by BSF under grant No. 2000154.

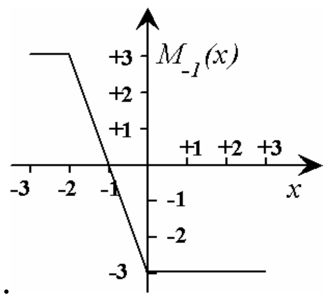

Figure 2. $M_{-1}(x)$ diagram for $k = 7$.

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

$$M_\alpha(x) = \begin{cases} S(ax;|\alpha|V_{dd}) & \text{if } \alpha < 0 \\ S(ax) & \text{if } \alpha = 0 \\ S(ax;|\alpha|V_{gnd}) & \text{if } \alpha > 0 \end{cases}. \quad (5)$$

Using the mask-function, it is possible to implement the fuzzy rule

$$\text{if } x=\alpha \text{ then} y=F(x=\alpha,Y) \text{ else} y=0, \quad (6)$$

that extracts a value of the function $F(x=\alpha,Y)$ at the point $x=\alpha$ as the circuit built of summing amplifiers shown in Fig.3.

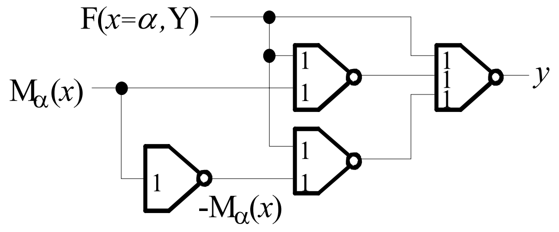

Figure 3. Implementation of the rule (6).

This implementation can be written in analytical form as

$$y=S(S(M_\alpha(x);F(x=\alpha,Y));S(-M_\alpha(x);F(x=\alpha,Y)); F(x=\alpha,Y)). \quad (7)$$

For example, if $\alpha=-1$, $F(x=-1,Y)=2$, and $k=7$, the behavior of the circuit in Fig.3 can be illustrated by Fig.4.

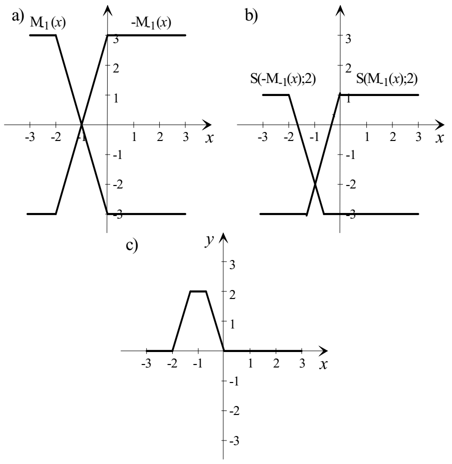

Figure 4. Example of implementing the rule " if *x=-1* then *y=2* else *y=0* ", (*k*=7).

The rule

$$\begin{gathered}\text{if } \gamma \le x \le \delta \text{ then} \\ y=F(\gamma \le x \le \delta, Y)=\Phi(Y) \text{ else } y=0\end{gathered} \quad (8)$$

that covers wider variables change range can be realized as

$$\begin{cases} M_{\lambda,\delta}(x) = S(M_{\lambda-1}(x); M_{\delta+1}(x)); \\ -M_{\lambda,\delta}(x) = S(M_{\lambda,\delta}(x)); \\ y = S(S(M_{\lambda,\delta}(x); \Phi(Y)); \\ \quad S(-M_{\lambda,\delta}(x); \Phi(Y)); \Phi(Y)). \end{cases} \quad (9)$$

This implementation corresponds to the circuit presented in Fig.3 if the inputs $M_\alpha(x)$ and $F(x=\alpha,Y)$ are changed to $M_{\lambda,\delta}(x)$ and $\Phi(Y)$ respectively.

3. EXAMPLE

This example is taken from [3]. Stepper motor fuzzy controller implements the function of two linguistic variables - "position error" x and "force error" y. The variables x and y each have seven values NL, NM, NS, ZE, PS, PM, PL and their membership functions are shown in Fig.5.

The Inference Engine Rule Matrix for the output linguistic variable as a multi-valued function is shown in Table 1 that comprises only two different columns defining two force error functions (Table 2).

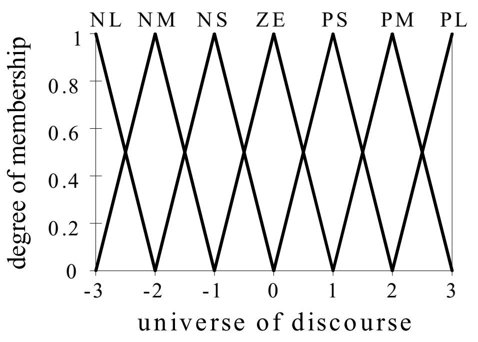

Figure 5. Fuzzy sets for force error and position error inputs.

Table 1. Inference engine rule matrix function

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

Table 2. Two different functions of the force error

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

Fig.6 shows the graphs of these functions.

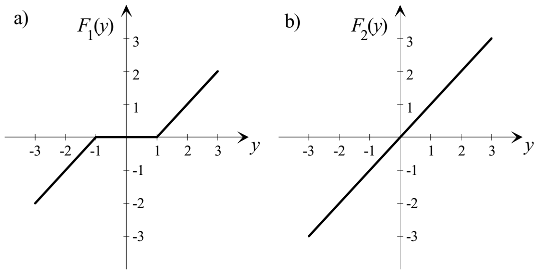

Figure 6. Components of the function decomposition defined by Table 1 for *x*.

The functions $F_1(y)$ and $F_2(y)$ can be implemented as

$$F_1(y) = S_3(\tfrac{2}{3}S_2(\tfrac{3}{2}y;\tfrac{3}{2}V_{gnd});\tfrac{2}{3}S_1(\tfrac{3}{2}y;\tfrac{3}{2}V_{dd})); \qquad (10)$$
$$F_2(y) = y$$

It can be seen from the Table 1 that the behavior of the controller's output decomposed by variable x has the form:

$$\begin{aligned} &\text{if} \quad -2 \le x \le +2 \quad \text{then} \quad Output = F_2(y) \\ &\qquad\qquad\qquad\qquad \text{else} \quad Output = F_1(y) \end{aligned} \qquad (11)$$

It is possible to split the rule (11) into two rules:

$$\begin{aligned} &\text{if} \quad -2 \le x \le +2 \quad \text{then} \quad Output = F_2(y) \\ &\qquad\qquad\qquad\qquad \text{else} \quad Output = 0 \end{aligned} \qquad (12)$$

and

$$\begin{aligned} &\text{if} \quad -2 \le x \le +2 \quad \text{then} \quad Output = 0 \\ &\qquad\qquad\qquad\qquad \text{else} \quad Output = F_1(y) \end{aligned} \qquad (13)$$

The rule (12) can be implemented in accordance with (9), (10), and (7) as

$$\begin{cases} M_{-2,+2}(x) = S_4(M_{-3}(x); M_{+3}(x)); \\ -M_{-2,+2}(x) = S_5(M_{-2,+2}(x)); \\ \Phi_1 = S_{11}(S_6(M_{-2,+2}(x); F_1(y)); \\ \qquad S_7(-M_{-2,+2}(x); F_2(y)); F_2(y)). \end{cases}$$

To implement the rule (13) let us first implement the rule (14):

$$\begin{aligned} &\text{if} \quad -2 \le x \le +2 \quad \text{then} \quad Output = F_1(y) \\ &\qquad\qquad\qquad\qquad \text{else} \quad Output = 0 \end{aligned} \qquad (14)$$

as

$$S_{10}(S_8(M_{-2,+2}(x);F_2(y));S_9(-M_{-2,+2}(x);F_2(y));F_2(y)).$$

The rule (13) can be represented as

$$\begin{aligned} &\text{if} \quad -2 \le x \le +2 \quad \text{then} \quad Output = -F_1(y) + F_2(y) \\ &\qquad\qquad\qquad\qquad \text{else} \quad Output = 0 + F_1(y) \end{aligned}$$

with implementation

$$\Phi_2 = S_{14}(S_{13}(S_{12}(S_{10});F_2(y))).$$

Finally, the controller output is

$$Output = S_{16}(S_{15}(\Phi_1;\Phi_2)).$$

It is possible to further simplify this controller implementation. Having noticed that the amplifier S_{10} works without saturation, we may implement the function $-\Phi_2$ as

$$-\Phi_2 = S_{10}(S_8;S_9;2F_1(y))$$

and the controller output will look like

$$Output = S(-\Phi_1;-\Phi_2) = S_{11}(S_6;S_7;F_2(y);S_{10}).$$

Fig.7 illustrates the structural scheme of the controller implementation (the input weights are marked).

The paper title (on the first page) should begin 35 mm (1-3/8 inches) from the top edge of the page, centered, first letters capitalized, and in Times 14-point, boldface type. The authors' name(s) and affiliation(s) appear below the title in capital and lower case letters. Papers with multiple authors and affiliations may require two or more lines for this information.

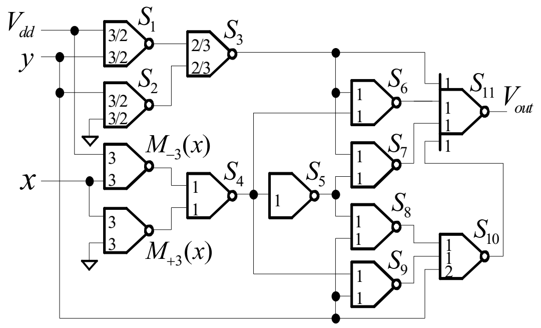

Figure 7. Structural diagram of the controller.

4. SPICE SIMULATION

We used three-stage and five-stage push-pull CMOS operational amplifiers with 2 Mom feedback resistors as a basic building block for the controller represented in Fig.7. Controller functioning was checked by SPICE simulation (MSIM 8), MOSIS BSIM3v3.1, 7 level models of 0.4µm transistors used.

Three-stage amplifier example is presented in Fig. 8.

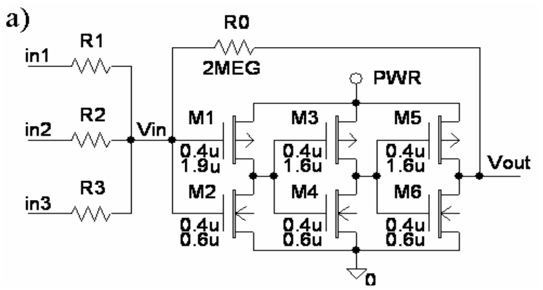

Figure 8. Implementation of the summing amplifier using three-stage push-pull CMOS operational amplifier.

The source voltage in our experiments was 3.5V. The variable *x* changed linearly from 0V to 3.5V.

The variable *y* changed discretely in accordance with its logical values and was kept constant within one cycle of *x* change.

SPICE simulation results confirm the controller correct logical functioning. The accuracy of output signal voltage levels for controller design based on three-stage amplifiers may be essentially improved if five-stage amplifiers are employed. This allows to considerably increase the number of logical gradations (e.g., 11 – 13 instead of 7).

To achieve the best rendering both in the proceedings and from the CD-ROM, we strongly encourage you to use Times-Roman font. In addition, this will give the proceedings a more uniform look. Use a font that is no smaller than nine point type throughout the paper, including figure captions.

5. CONCLUSIONS

Analog implementation of fuzzy controllers has the advantages of better speed and reliability, lower power consumption etc. Such kind of implementation is possible due to a functional completeness of summing amplifier with saturation in a multi-valued logic of an arbitrary value. In this paper, we suggested a special design procedure that may be a useful instrument for a systematic analog fuzzy controllers design. Our theoretical considerations are solidified by a real life stepper motor fuzzy controller design example and associated SPICE simulation results.

6. REFERENCES

[1] V. Varshavsky, V. Marakhovsky, I. Levin, and N. Kravchenko, "Fuzzy Device", *New Japanese Patent Application No. 2003-190073, filed to Japan's Patent Office, Feb. 6, 2003.*

[2] V. Varshavsky, V. Marakhovsky, I. Levin, and N. Kravchenko, "Summing Amplifier as a Multi-Valued Logical Element For Fuzzy Control", *WSEAS Transactions on Circuit and Systems*, *Issue 3, Volume 2, July 2003, pp. 625 – 631.*

[3] J.G.Hollinger, R.A.Bergstrom, and J.S.Bay, "A Fuzzy Logic Force Controller for a Stepper Motor Robot*", Proceedings of the 1992 IEEE International Symposium on Intelligent Control, pp. 204-209.*