Fuzzy Optim Decis Making (2006) 5:113–122
DOI 10.1007/s10700-006-7330-1

Hardware implementation of fuzzy PID controllers

Taifu Li · Jundi Xiong · Rui Zhang · Qifu Tan · Ruizheng Xu

Abstract For traditional hardware implementation of fuzzy PID controllers, it is large at computation and bad in real-time performance, so, a kind of PID control algorithm, whose gain parameters could be tuned by their fuzzy system, was selected as studying example for a novel idea of hardware implementation. In this paper, authors presented hardware network of memory address mapping to implement fuzzy PID control algorithm, and designed the corresponding hardware system. The idea actually realizes fusion of hardware and intelligent algorithm. The implementation effectively simplified hardware circuits, the whole controller is very simple without CPU. Meanwhile, it is very easy to use, only connecting the sensor/transducer, the driver and the actuator is OK. The controller is very rapid in response, it need only two A/D conversion periods for outputting a required control signal. So the implementation could meet real-time performance effectively.

Keywords Fuzzy · PID · Memory address mapping

1. Introduction

In many industrial process control systems, the controlled process is complex in mechanism, and varying with time, so general PID control is very difficult to obtain satisfactory effects because it is not self-adaptive for many varying factors such as parameter-varying

T. Li (✉)
College of Electronic Information Engineering,
Chongqing University of Science and Technology, Chongqing, 400050, China
e-mail: litaifu@tom.com

J. Xiong · R.Zhang · Q.Tan · R.Xu
College of Information and Automation, Chongqing Institute of Technology,
Chongqing, 400050, China

J. Xiong
e-mail: xiongjundi@cqit.edu.cn

R.Zhang
e-mail: nebuladeep@163.com

and all kinds of disturbances. While these factors are over some bounds, the performance would worsen obviously, even exceed to the tolerance. In order to avoid the disadvantage of the general PID controller and improve its performance, documents (Astrom & Hagglund, 1989) and (Hoopes, Hawk & Lewis, 1983) both presented self-tuning PID controller, whose method is actually same as that of general self-tuning controller, i.e., recognizing the plant model on-line, then regulating PID gain parameters based on the recognized model. This method did not change the fact that general PID controllers tune parameters depended on their models, it is necessary to recognize the process model on-line, so, the computation is very large. Zhao and Tomizuka presented a fuzzy gain scheduling method of PID controllers (Zhao, Tomizuka & Isaka, 1993), whose fuzzy rules and reasoning are utilized on-line to determine the controller parameters based on the error signal and its first-order derivative. Gain fuzzy rules of PID controller are generally from 'rule of thumb' of skillful manipulators. Then these rules were fused into fuzzy system to tune PID gains on-line. In the discrete universe of the error, the change of error and output control signal, computing all possible output control values through fuzzy system of K_p, K_i, and K_d, in this way, their two-dimensional look-up table could be get. Gains parameters can be tuned by these look-up tables on-line, the computation had reduced greatly. However, it still is necessary to write lots of programs with hardware implementation in industrial control computer or micro-control unit, improvement at real-time performance is not very obvious. Therefore, the fuzzy PID control algorithm in the paper (Zhao, Tomizuka & Isaka, 1993) was selected as studying example. In this paper, authors presented hardware network of memory address mapping to implement the fuzzy PID control algorithm, and designed the corresponding hardware system and detailed implementation circuits.

2. Function representation based on memory address mapping

If, we take memory address [00H, FFH] as the domain of input variable, corresponding memory unit as the domain of function value [00H, FFH], then it is possible to establish a mapping between a memory address and a value of corresponding memory unit, this kind of mapping can realize function approximation. If the range of change of the address is [00H, FFH], the range of change of the value in memory unit is also [00H, FFH], in this way, this kind of address mapping can approach one-dimensional function. The sketch is shown in Fig. 1,

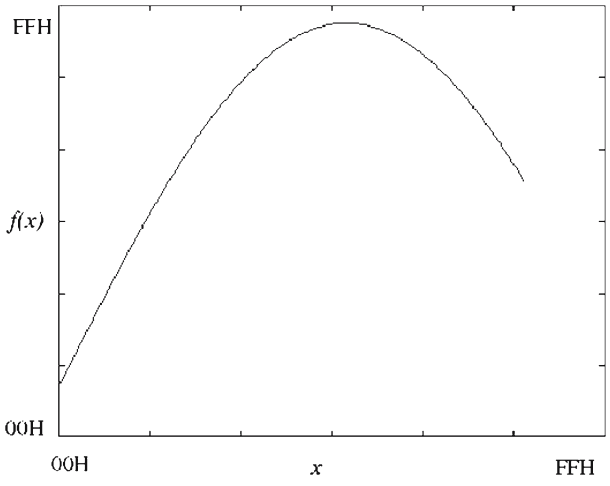

Fig. 1 One-dimensional function with memory address mapping

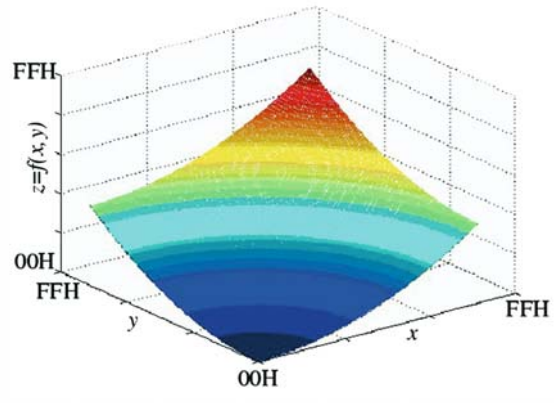

Fig. 2 Two-dimensional function with memory address mapping

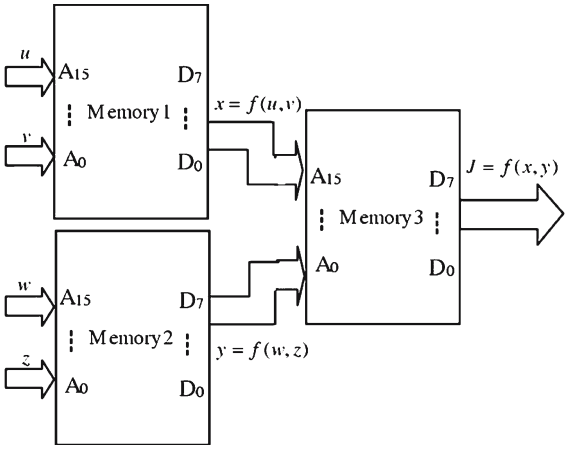

Fig. 3 Serial address mapping of several memories

memory address represents variable x, the function value in corresponding memory unit can be represented by $f(x)$. For one-dimensional function, it actually is a mapping, i.e., $R \rightarrow R$, one-dimensional memory address mapping is similar to $[00H, \text{FFH}] \rightarrow [00H, \text{FFH}]$, but there is still little difference because the input and output space in memory address mapping are both discrete.

Generally, memory address has multi-bit, we also divide it into high-bit address and low-bit address, or page address and page-inside address. If we apply high-bit address and low-bit address at the same time access to implement memory address mapping, this kind of mapping can realize two-dimensional function approach, the sketch is shown in Fig. 2. For twodimensional function, it actually is a mapping, i.e., $R^2 \rightarrow R$, two-dimensional memory address mapping is similar to $[00H, \text{FFH}]^2 \rightarrow [00H, \text{FFH}]$.

For three-dimensional or more dimensional function, we may apply composite function method. For example, function $J = f(x, y)$, and $x = g(u, v)$, $y = h(w, z)$, the complex function can be approached by serial address mapping of several memories, shown in Fig. 3. First, variable u and v both access memory 1 with high-bit and low-bit address, respectively, and output function value x, at the same time, variable w and z both access memory 2, and output function value y. Then takes x and y as input variable for memory 3, and output function value J with memory address mapping. For any mapping $R^n \rightarrow R^m$, if it

may be separated into many mapping $R^n \to R$, and every mapping $R^n \to R$ also may be separated into many mapping $R^2 \to R$ and $R \to R$, then the mapping $R^n \to R^m$ could be approximately implemented with memory address mapping networks.

3. Fuzzy gain scheduling of PID controllers

Zhao, Tomizuka and Isaka, (1993)

The transfer function of a PID controller has the following form:

$$G_c(s) = K_p + K_i/s + K_d s \tag{1}$$

where K_p, K_i, and K_d are the proportional, integral, and derivative gains, respectively. Another useful equivalent form of the PID controller is

$$G_c(s) = K_p(1 + 1/(T_i s) + T_d s) \tag{2}$$

where $T_i = K_p/K_i$ and $T_d = K_d/K_p$. T_i and T_d are known as the integral and derivative time constants, respectively.

The discrete-time equivalent expression for PID control used in this paper is given as

$$u(k) = K_p e(k) + K_i T_s \sum_{i=1}^{n} e(i) + \frac{K_d}{T_s}\Delta e(k) \tag{3}$$

Here, $u(k)$ is the control signal, $e(k)$ is the error between the reference and the process output, T_s is the sampling period for the controller, and $\Delta e(k) \triangleq e(k) - e(k-1)$

The parameters of the PID controller K_p, K_i, and K_d or K_p, T_i, and T_d can be manipulated to produce various response curves from a given process. Finding optimum adjustments of a controller for a given process is not trivial. In succession, an on-line gain scheduling scheme of the PID controller based on fuzzy rules is introduced.

Figure 4 shows the PID control system with a fuzzy gain scheduler. It is assumed that K_p, K_d are in prescribed ranges $[K_{p\min}, K_{p\max}]$ and $[K_{d\min}, K_{d\max}]$, respectively. For convenience, K_p and K_d are normalized into the range between zero and one by the following linear transformation:

$$K_p' = \frac{K_p - K_{p\min}}{K_{p\max} - K_{p\min}} \tag{4}$$

$$K_d' = \frac{K_d - K_{d\min}}{K_{d\max} - K_{d\min}} \tag{5}$$

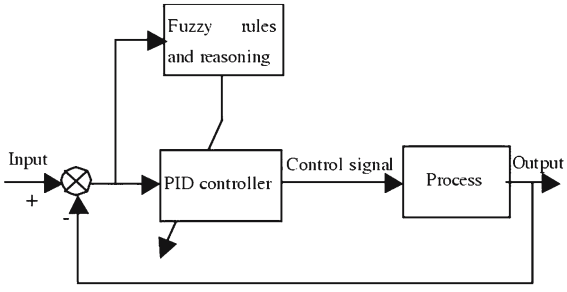

Fig. 4 PID control system with a fuzzy gain scheduler

The integral time constant is determined with reference to the derivative time constant, i.e.,

$$T_i = \alpha T_d \tag{6}$$

and the integral gain is thus obtained by

$$K_i = K_p/(\alpha T_d) = K_p^2/(\alpha K_d) \tag{7}$$

The parameters K'_p, K'_d, and α are determined by a set of fuzzy rules of the form

$$\begin{array}{l} \text{if } e(k) \; is \; A_i \text{ and } \Delta e(k) \text{ is } B_i, \text{ then} \\ K'_p \; is \; C_i, \;\; K'_d \text{ is } D_i, \;\; \text{and } \alpha = \alpha_i \\ i = 1, 2, \ldots, mi \end{array} \tag{8}$$

Here, A_i, B_i, C_i, and D_i are fuzzy sets on the corresponding supporting sets; α_i is a constant. The membership functions of these fuzzy sets for $e(k)$ and $\Delta e(k)$ are shown in Fig. 5. In this figure, N represents negative, P positive, ZO approximately zero, S small, M medium, B big. Thus NM stands for negative-medium, PB for positive big, and so on.

The fuzzy sets C_i and D_i may be either Big or Small and are characterized by the membership functions shown in Fig. 6.

The fuzzy set α may also be considered as a fuzzy number that has a singleton membership function as shown in Fig. 7. For example, α becomes 2 when α is Small. In this figure, S represents small, MS medium-small, M medium, B big.

The fuzzy rules in Eq. (8) may be extracted from operator's expertise, and refer document (Zhao, Tomizuka & Isaka, 1993) for the detail. A set of rules, as shown in Table 1 , may be used to adapt the proportional gain (K'_p). The tuning rules for K'_d and α are given in Tables 2 and 3 respectively. In the tables, B stands for Big, and S for Small.

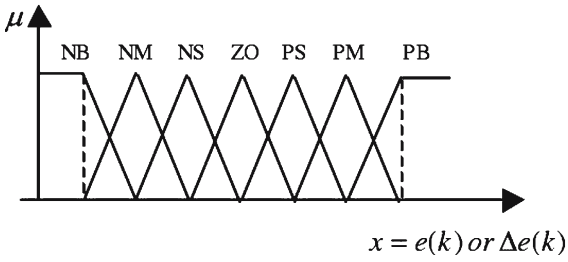

Fig. 5 Membership functions for $e(k)$ and $\Delta e(k)$

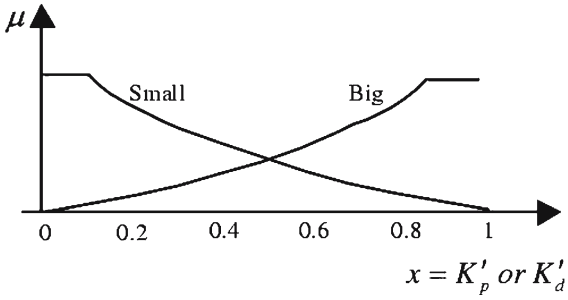

Fig. 6 Membership functions for K'_p and K'_d

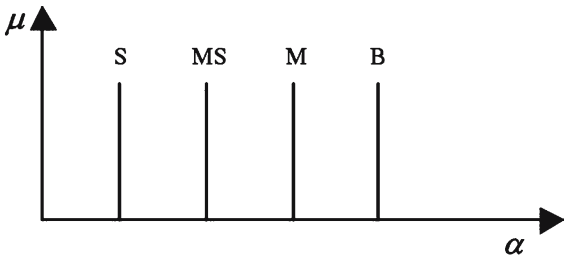

Fig. 7 Singleton membership function for α

Table 1 Fuzzy tuning rules for K'_p

$\Delta e(k)$ e(k)	NB	NM	NS	ZO	PS	PM	PB
NB	B	B	B	B	B	B	B
NM	S	B	B	B	B	B	S
NS	S	S	B	B	B	S	S
ZO	S	S	S	B	S	S	S
PS	S	S	B	B	B	S	S
PM	S	B	B	B	B	B	S
PB	B	B	B	B	B	B	B

Table 2 Fuzzy tuning rules for K'_d

e(k) $\Delta e(k)$	NB	NM	NS	ZO	PS	PM	PB
NB	S	S	S	S	S	S	S
NM	B	B	S	S	S	B	B
NS	B	B	B	S	B	B	B
ZO	B	B	B	B	B	B	B
PS	B	B	B	S	B	B	B
PM	B	B	S	S	S	B	B
PB	S	S	S	S	S	S	S

Table 3 Fuzzy tuning rules for K'_p

e(k) $\Delta e(k)$	NB	NM	NS	ZO	PS	PM	PB
NB	2	2	2	2	2	2	2
NM	3	3	2	2	2	3	3
NS	4	3	3	2	3	3	4
ZO	5	4	3	3	3	4	5
PS	4	3	3	2	3	3	4
PM	3	3	2	2	2	3	3
PB	2	2	2	2	2	2	2

Selecting the Mamdani product inference engine, singleton fuzzifier and center average defuzzifier, then, K'_p, K'_d and α may be tuned by the following:

$$K'_p(k) = \frac{\sum_{l=1}^{49} \bar{y}_p^l \mu_{A^l}(e(k)) \mu_{B^l}(\Delta e(k))}{\sum_{l=1}^{49} \mu_{A^l}(e(k)) \mu_{B^l}(\Delta e(k))} \tag{9}$$

$$K'_d(k) = \frac{\sum_{l=1}^{49} \bar{y}_d^l \mu_{A^l}(e(k)) \mu_{B^l}(\Delta e(k))}{\sum_{l=1}^{49} \mu_{A^l}(e(k)) \mu_{B^l}(\Delta e(k))} \tag{10}$$

$$\alpha(k) = \frac{\sum_{l=1}^{49} \bar{y}_\alpha^l \mu_{A^l}(e(k)) \mu_{B^l}(\Delta e(k))}{\sum_{l=1}^{49} \mu_{A^l}(e(k)) \mu_{B^l}(\Delta e(k))} \tag{11}$$

where, A^l, B^l are membership function shown in Fig. 5, $\bar{y}_p^l$, $\bar{y}_d^l$, and $\bar{y}_\alpha^l$ are the corresponding center average value shown in Figs. 6 and 7.

4. Fuzzy PID controller implemented by memory address mapping network

According to Eqs. (4), (5), (7), and (9), (10), (11), we may get on-line self-tuning parameters K_p, K_d, and K_i. Moreover, rounding the $e(k)$ and $\Delta e(k)$ at the range of their universe, we may get look-up table for K_p, K_d, and K_i. In this way, complex computation can be simplified obviously. However, simple look-up table programs are also very large, it is possible to influence the performance in real time. In this section, how to design above-mentioned fuzzy PID controller based on memory address mapping network is introduced.

For Eq. (3), replace it with the following:

$$u(k) = u_{pd}(k) + u_i(k) \tag{12}$$

where

$$u_{pd}(k) = K_p e(k) + \frac{K_d}{T_s} \Delta e(k) \tag{13}$$

In Eq. (13), K_p is function with respect to $e(k)$ and $\Delta e(k)$, K_d also is function with respect to $e(k)$ and $\Delta e(k)$, T_s is constant, so $u_{pd}(k)$ is also function with respect to $e(k)$ and $\Delta e(k)$. In addition, if there are signals of $e(k)$ and $e(k-1)$, it actually includes information $\Delta e(k)$ because of $\Delta e(k) \triangleq e(k) - e(k-1)$. So, K_p, K_d and $u_{\mathrm{pd}}(k)$ can be converted into function with respect to $e(k)$ and $e(k-1)$. Known $e(k)$ and $e(k-1)$, we can get $u_{\mathrm{pd}}(k)$ through formula (4), (5), (9), and (10).

The block diagram of fuzzy PID controller implemented by memory address mapping network is shown in Fig. 8. The analog signal of the error can be get through comparing the setting reference signal and feedback signal by subtracter IC1. After A/D converter, get the digital signal of the error, the $(2k-1)^{\mathrm{th}}$ error signal $e(2k-1)$ is locked in Flip-latch 1 (IC3), the $(2k)^{\mathrm{th}}$ error signal $e(2k)$ is locked in Flip-latch 2 (IC4). Taken $e(2k-1)$ and $e(2k)$ as address of Memory (1), we may get u_{pd} by memory address mapping.

In Eq. (7), K_i is function with respect to K_p, K_d, and α, in which, K_p, K_d, and α all are function with respect to $e(k)$ and $\Delta e(k)$, so K_i is two-dimensional function with respect to $e(k)$ and $\Delta e(k)$, or with respect to $e(k)$ and $e(k-1)$. Similar to Memory (1), we also may get K_i through address mapping in Memory (2).

In Eq. (14), $\sum_{i=1}^{n} e(i)$ may implemented by address mapping in Memories (3) and (4) realizes product of K_i, $\sum_{i=1}^{n} e(i)$, and constant T_s, output value of Memory (4) is u_i. Address mapping in Memory (5) implements addition operation with respect to u_i and u_{pd}, in this way, output value of Memory (5) $u(2k)$ is the digital control signal of fuzzy PID controller, in which, $2k$ represents that outputting one control signal need two A/D conversion periods. After D/A converter, the digital control signal will be converted into analog control signal (0–5V, or −5V – +5V), then outputted toward the actuator to control the plant.

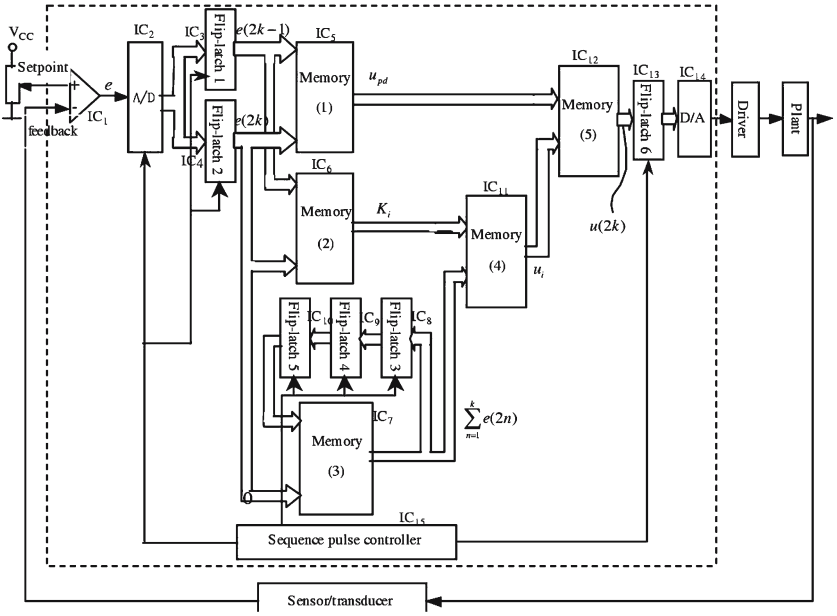

Fig. 8 Fuzzy PID controller based on memory address mapping network

For whole controller's operation, it is controlled by time sequence pulse controller that control A/D converter and all flip-latches, all pulse sequences are shown in Fig. 9. The time sequence pulse controller actually is very simple, only including frequency-division and inverse-phase circuits, its detail circuits actually has been designed in left down corner in

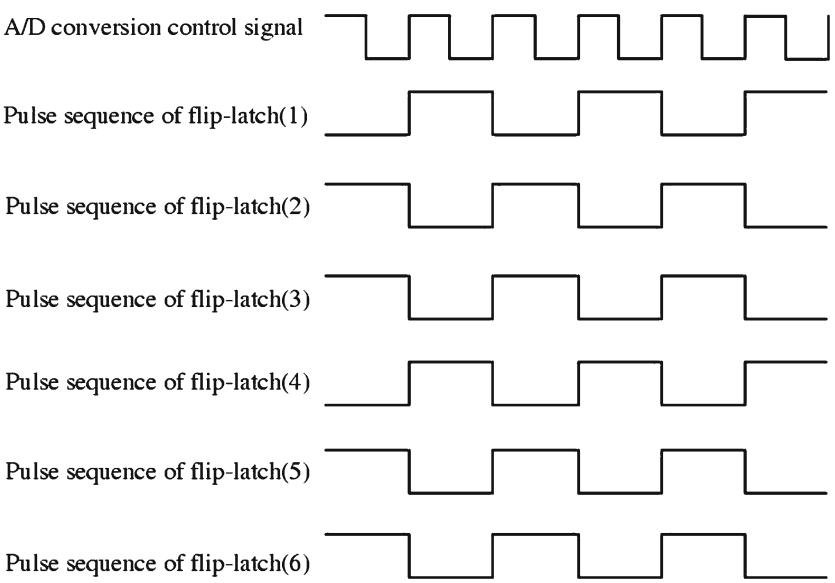

Fig. 9 Time sequence pulse control signal

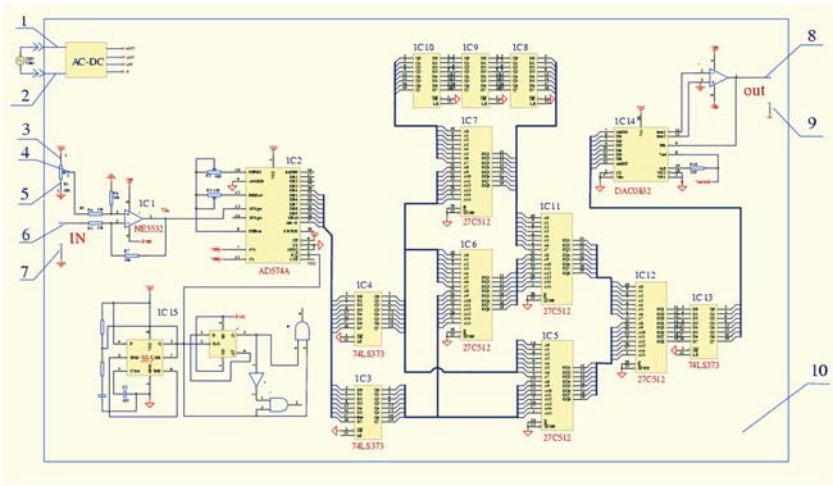

Fig. 10 The detailed implementation circuits

Fig. 10, The detailed implementation circuits about the whole fuzzy PID controller is shown in Fig. 10.

This kind of hardware implementation design still has not been verified by the actual circuits and experiment, but it is certain to realize it because authors have developed various fuzzy controllers with memory address mapping method. If you are interested it, the detail can be found in documents (Li & Zhao, 2004; Li, Ren & Zeng, 2005; Li, Zhang & Chen, 2005; Li, Huang & Deng, 2005).

5. Conclusion

Aimed at complex algorithm and large computation in fuzzy PID controller, authors present a new methodology to implement it based on hardware structure of memory address mapping networks. The methodology overcomes a disadvantage that there is no relation between memory address and data in corresponding memory unit in traditional computer principle, and solves the puzzle that complicated intelligent control algorithm is bad in real time, slow in response. This kind of implementation methodology is very high in speed, outputting one control signal only need two A/D conversion periods (about 1–2 μS), similar to conditioned reflex of human and animal. The circuits of whole fuzzy controller without CPU are very simple. The fuzzy controller overcomes some troublesome problem to apply Micro-control Unit (MCU) to develop fuzzy control system. Developers of the control system do not need developing any programs, directly connect peripheral analog device, its application is very simple. If the developed fuzzy controller is integrated into special chip as a kind of general controller, user applies the controller is so simple as fastener and screw, and cost is very low, volume is very small.

Acknowledgements The supports from Chongqing Natural Science Foundation of China (No. 2004-8661) and Chongqing Educational Committee Science Technology Research Project (No.040603) are gratefully acknowledged.

References

Astrom, K.J., & Hagglund, T. (1989). An industrial adaptive PID controller. *In: Proceedings 1989 IFAC Symposium on Adaptive System in Control and Signal Processing*, 293–298.

Hoopes, H.S., Hawk, W.K., & Lewis, R.C. (1983). A self-tuning controller. ISA Transition *22*, 49–58.

Li, T., Huang, L., & Deng, R. (2005) Control puzzle of complicated system and explorations into a solution of fusing intelligence with hardware, *Advances in Systems Science and Application. 5*(3), 411–419.

Li, T., Ren, H., & Zeng, W. (2005). A Realization Method of Computational Intelligence with Memory Address Mapping and its Application on Fuzzy Control. *In: Proceedings of International Symposium on Computational Intelligence and Industrial Application*, Dec, 20–24, P.R.China: Hainan.

Li, T., Zhang, R., & Chen, H. (2005). A high-speed fuzzy controller with human simulated intelligent weight coefficient. *Proceedings of International Conference on Mechatronics and Information Technology*, September 20–23, P.R.China: Chongqing.

Li, T., & Zhao, M. (2004). A kind of fuzzy controller implemented by memory address mapping. *IEEE International Conference Robotics, Intelligent Systems and Signal Processing* , October 8–13, P.R.China: Changsha.

Zhao, Z.Y., Tomizuka, M., & Isaka, S. (1993). Fuzzy gain scheduling of PID controllers. *IEEE Trans. on Systems, Man, and Cybernetics, 23*(5), 1392–1398.