Transputer/Occam Japan 5
S. Noguchi and M. Yamamoto, Eds.
IOS Press, 1993

Transputer Design for Real-time Fuzzy Controller

Kazuhiko HASEGAWA *
Dept. of Naval Architecture & Ocean Engineering
Faculty of Engineering, Osaka University
2-1, Yamada-oka, Suita, 565 Japan

Job van Amerongen and André W.P. Bakkers
Control Laboratory
Control, Sysems and Computer Engineering Group
Electrical Engineering Department, University of Twente
P.O. Box 217, 7500 AE Enschede, The Netherlands

Abstract. Transputer design for real-time fuzzy controller is described. The process under considering is an inverted pendulum. To control the pendulum upright, it requires fast computation. Statefeedback control is easily implemented for real-time control, but it is not applicable for motions with large nonlinearity. Fuzzy control is then applied for this problem and the discussion will be how the transputers are used. Parallel computation should be maximumly introduced to achieve high performance. For this purpose, basic building block method was used in which the fuzzy controller is described in combining elementary components. In the method each component can be separately calculated and the total network can be coded in PAR block of these components in OCCAM convention. Finally, a result of the real system is shown.

Keywords transputer application, parallel computation, fuzzy controller, real-time control, inverted pendulum

1. Introduction

The real-time control becomes more and more important according to the rapid development of robotics and control theory. In this paper, control problem of an inverted pendulum is treated. An inverted pendulum is the system of inherently unstable and nonlinear. It is quite difficult for normal optimal theory to control it for a wide range of motion and disturbance. Human beings have a capability for control it, just like balancing a pole on a palm or riding a bicycle. However, it is difficult even for human beings to control it for a long time or to control very short pole. Then fuzzy

*This research was done at *University of Twente* during his stay of Aug. 1991 - Apr. 1992.

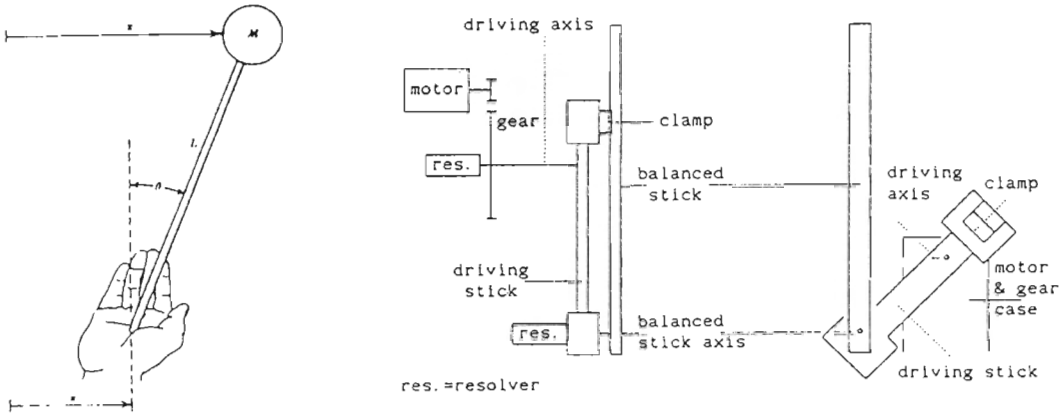


Fig. 1 Manual balancing of a stick and its experimental setup

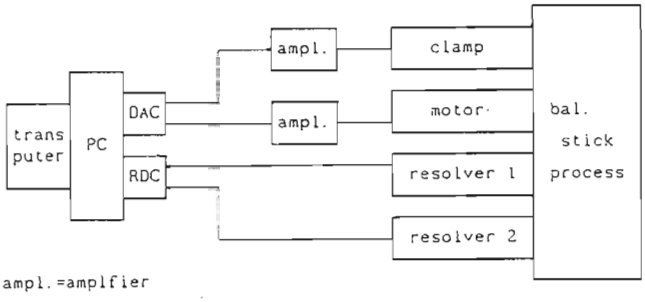


Fig. 2 Block diagram of the balanced stick system

theory is applicable for these situations. It is based on linguistic-based rules, so we can apply it for the problem for which we can tell the solution. The problem is its computation speed. There are already proposed several kinds of fuzzy LSI chips. Yamakawa and Sakai designed it based on analog circuit[1]. Togai and Watanabe, on the other hand, proposed complete digital-based chip[2]. Some companies sell the fuzzy inference board for PC using these chips. However, we can do the same thing using transputers. We have developed fuzzy controller utilizing transputer's superiority of parallelism and modularity.

2. Description of the Control Process

An inverted pendulum is a classic control problem. The task of this problem is to keep a pole or a stick in upright condition by moving the end of the stick. Usually we, human beings, put the stick on our palms and shift palms mainly in the horizontal plane as shown in Fig. 1.

Several types of the experimental setup for this problem are available, but in this research the setup also shown in Fig. 1 was used. This setup involves more nonlinear effect than so-called cart-pole system.

The motor is controlled by a computer, an IBM-PC compatible. The computer is equipped with a 16-bit DAC (Digital to Analog Converter). On the driving axis and on the balanced stick axis resolvers are mounted to measure the angles. One resolver mounted on the driving axis measures the angle of the driving stick, which we call ϕ, and the other mounted on the balanced stick axis measures the angle between the driving stick and the balanced stick. As we define the angle between the balanced stick and a fixed vertical line as θ, the second resolver measures the angle $\phi - \theta$.

To simplify the positioning the balanced stick upright, a clamp is mounted on the driving stick. If the balanced stick is in line with the driving stick, the clamp can keep it in this position. If the clamp is pulled back, the balanced stick can rotate freely. The clamp is also controlled by the computer.

The computer is equipped with a transputer board, on which 7 INMOS T800 transputers can be used. In this study, 4 of them are used and coded in OCCAM2. After a C-program has called the function that boots the transputer, both programs are running at the same time, synchronizing each cycle. The cycle time is 10 msec.

The block diagram of the total system is shown in Fig. 2.

3. Fuzzy Theory

3.1 Definition of Membership Function

Membership function is a mathematical way of representing a certainty of a concept. A concept is treated by a linguistic variable such as PB, which stands for "positive big" or NS, for "negative small". So, each membership function may have value from 0 to 1. The shape of the function is arbitrary.

However, for the simplicity, we treat the membership functions under the following assumptions.

- Each membership function is represented by a triangle whose height is 1 if bottom is defined as 0.
- Every two adjacent membership functions overlap just on the peak the other membership function each other.
- Both sides of outer region are exceptionally defined as combination of ramp and step functions.

This means any value of the considering variable is represented by two linguistic valiables except in the outer regions, and each membership function will vanish at the peak of the adjacent membership function. Let's show an example. In the case of the age of a person, suppose the "young" person is peak at 20 and the "old" person is peak at 60. There is no person under 20, who looks already old and there is no person over 60, who looks still young. If he is 40 years old, we can define his oldness using the fuzzy logic as:

$$40\text{years old} = \begin{cases} \text{Young} & \text{for 50 \% certainty} \\ \text{Old} & \text{for 50 \% certainty} \end{cases} \tag{1}$$

Every person can be expressed by two closely related linguistic variables such as "young" and "relatively young". There is no person except "very, very young" person

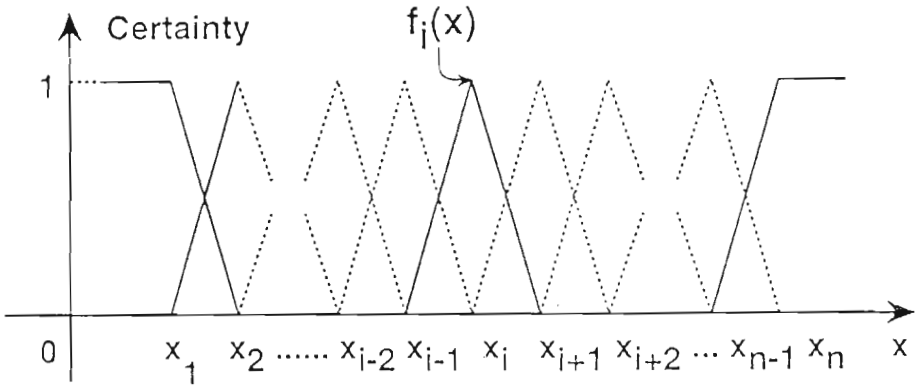


Fig. 3 Definition of membership functions

such as a baby or "absolutely old" person such as of over 100 years old, who can be said by one linguistic variable. For persons in the outer regions we use still linguistic variable but at the heighest certainty.

However, for flexibility, we reserve the following generalization.

- Each membership function is composed of an asymmetrical triangle.

This means that the peak of each membership function is not at the center of the width of the membership function. According to the above assumptions, we can define the membership function of a "linguistic variable" as follows.

$$f_i(x) = \begin{cases} 0 & \text{if } x < x_{i-1} \\ 1/(x_i - x_{i-1}) \cdot (x - x_i) + 1 & \text{if } x_{i-1} \le x < x_i \\ -1/(x_{i+1} - x_i) \cdot (x - x_i) + 1 & \text{if } x_i \le x < x_{i+1} \\ 0 & \text{if } x_{i+1} \le x \end{cases} \tag{2}$$

where

$$\begin{aligned} i &= 1, 2, \ldots, n \\ x_0 &= -\infty \\ x_{n+1} &= \infty \end{aligned}$$

Fig. 3 shows the membership functions thus defined.

3.2 Fuzzy Reasoning

As fuzzy control is equivalent with fuzzy reasoning in its procedure, fuzzy reasoning will be explained here.

Fuzzy reasoning will be done using several sets of fuzzy rules. Each fuzzy rule may contain unlimited number of linguistic variables, but it can be expressed by a generalized fuzzy rule containing of two inputs and one output. That is, for example,

$$\begin{aligned} &If \quad \text{the error } (e) \text{ is PB and the rate of error}(\dot{e}) \text{ is PB,} \\ &\textit{then} \ \text{the control output(u) should be NB.} \end{aligned} \tag{3}$$

Table 1 An example of fuzzy rules in a matrix

e / $\dot{e}$	NB	NS	ZE	PS	PB
NB	PB	PS	PS	ZE	ZE
NS	PS	PS	PS	ZE	ZE
ZE	PS	PS	ZE	NS	NS
PS	ZE	ZE	NS	NS	NS
PB	ZE	ZE	NS	NS	NB

It can be described in simpler way as:

$$u = \text{NB} \quad \text{if } e \text{ is PB, and } \dot{e} \text{ is PB} \tag{4}$$

If we treat the above relation in a matrix format, the fuzzy rules can be summarized as shown in Table 1.

In this example, each variable e, $\dot{e}$, and u and same set of linguistic variables, *i.e.* NB, NS, ZE, PS and PB, but in general, each variable has different numbers of linguistic variables. Here we denotes the number as l, m, and n respectively.

If a set of inputs (e and $\dot{e}$) is given, fuzzy reasoning will be done as follows.

- Determine the membership function values of each input.

$$\left.\begin{array}{ll} \tilde{e}_l(e) = f_{el}(e) & \text{where } l = \text{NB, NS, ZE, PS and PB} \\ \tilde{\dot{e}}_m(\dot{e}) = f_{\dot{e}m}(\dot{e}) & \text{where } m = \text{NB, NS, ZE, PS and PB} \end{array}\right\} \tag{5}$$

- For each cell of the control matrix, choose the minimum value of $\tilde{e}$ and $\tilde{\dot{e}}$.

$$\left.\begin{array}{ll} w_{ijk}(e, \dot{e}) = \min(\tilde{e}, \tilde{\dot{e}}) & \text{where } k = \text{NB, NS, ZE, PS and PB} \\ & \text{and } i, j = \text{corresponding values in the matrix} \end{array}\right\} \tag{6}$$

- Choose the maximum value from the different combinations of i and j for each k.

$$w_k(e, \dot{e}) = \max_{i,j}(w_{ijk}) \tag{7}$$

The procedure is illustrated in Fig. 4.

3.3 Sum-upped Membership Function

Sum-upped membership function is defined as taking max value of overlapped membership functions; *i.e.* the envelop of the overlapped membership functions as shown in Fig. 5.

The sum-upped membership function is composed by several weighted membership functions of different u_k values. The area and moment of each weighted membership function is described as:

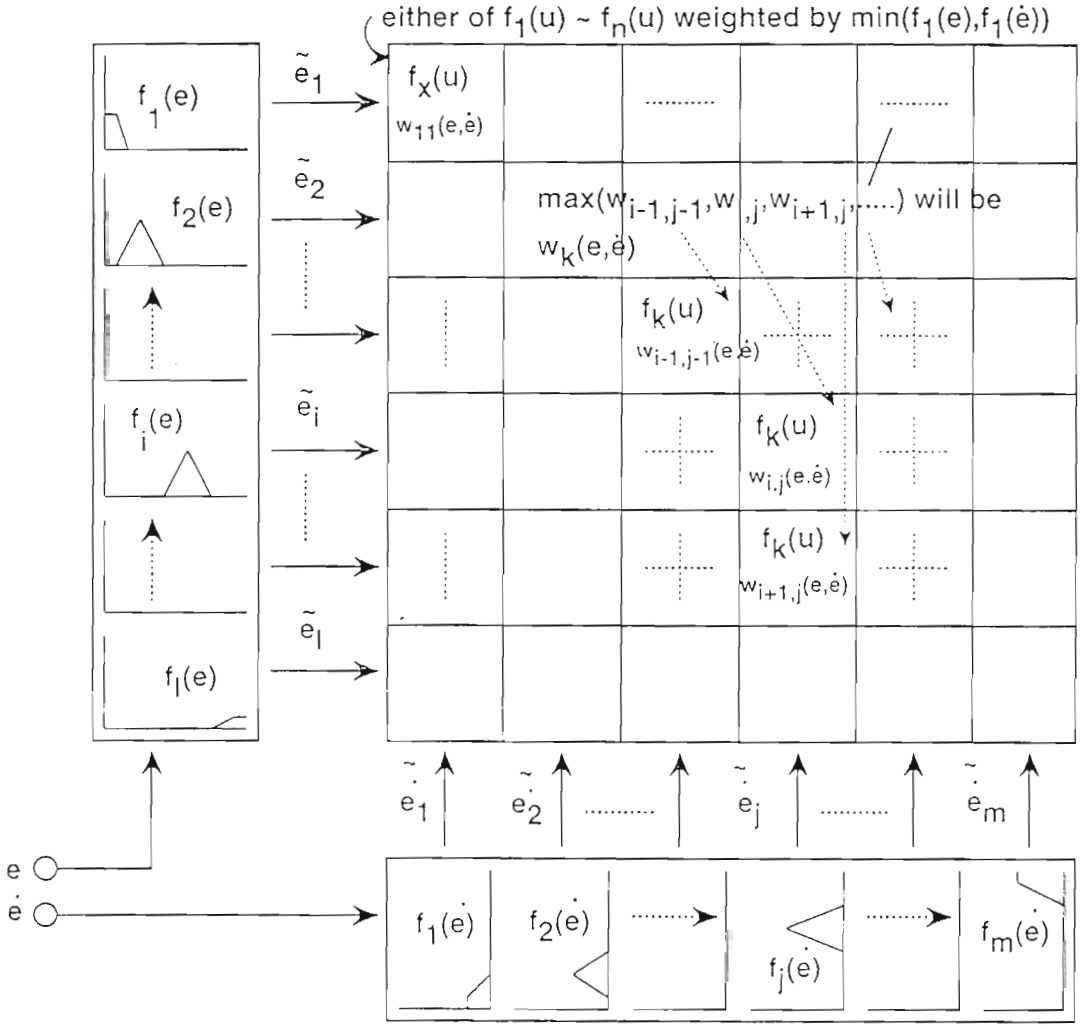


Fig. 4 Illustrated procedure of fuzzy reasoning

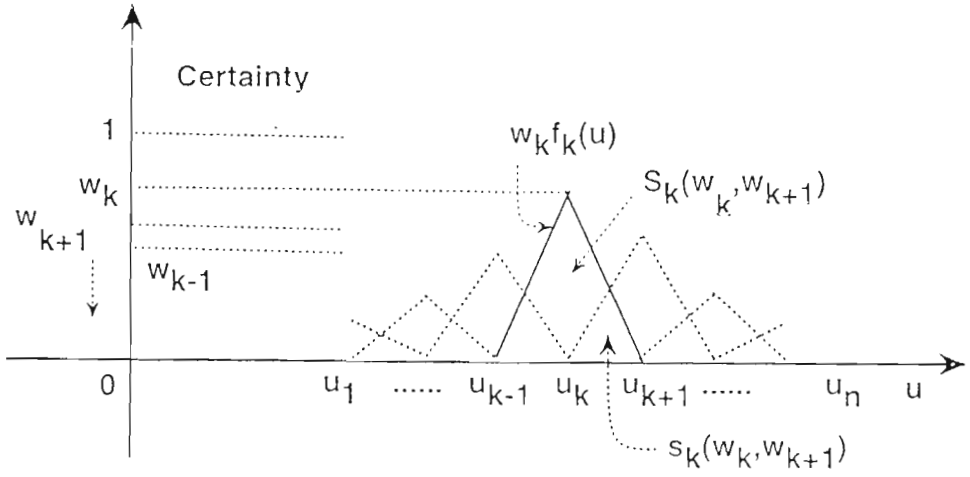


Fig. 5 Sum-upped membership function

$$\begin{aligned} S_k(w_k) &= \int_{u_1}^{u_n} f_k(u)du \\ &= 1/2 \cdot (u_{k+1} - u_{k-1}) \cdot w_k \end{aligned} \tag{8}$$

$$\begin{aligned} M_k(w_k) &= \int_{u_1}^{u_n} u f_k(u)du \\ &= S_k(w_k) \cdot (5u_{k-1} - 4u_k + 5u_{k+1})/6 \end{aligned} \tag{9}$$

However, for the sum-upped membership function, we should be carefully to take account of the overlapped areas. Thus,

$$S(e, \dot{e}) = \sum_{k=1}^{n} (S_k(w_k) - s_k(w_k, w_{k+1})) \tag{10}$$

$$M(e, \dot{e}) = \sum_{k=1}^{n} (M_k(w_k) - m_k(w_k, w_{k+1})) \tag{11}$$

where

$$s_k(w_k, w_{k+1}) = (u_{k+1} - u_k) \cdot 1/2 \cdot \frac{w_{k+1}u_k + w_k u_{k+1}}{w_k + w_{k+1}} \tag{12}$$

$$m_k(w_k, w_{k+1}) = s_k(w_k, w_{k+1}) \cdot \frac{5w_k u_k + w_{k+1}u_k + w_k u_{k+1} + 5w_{k+1}u_{k+1}}{6(w_k + w_{k+1})} \tag{13}$$

Finally, the output u can be derived as a center of gravity of the sum-upped area of the output membership function.

$$u(e, \dot{e}) = \frac{M(e, \dot{e})}{S(e, \dot{e})} \tag{14}$$

4. Parallel Realization of Fuzzy Controller

4.1 The Basic building block method

The basic building block method[3][4] can be used for realizing the parallel programming. Starting with very small processes and build a system with these elementary processes, the internal scheduler on the transputer will realize parallelism while running these processes. Fig. 6 shows the layout of one process given and Fig. 7 illustrate how the total system is interconnected and coded.

Some examples of the elementary processes are shown in Fig. 8. Using these elementary processes, more complicated processes will be also realized in parallel strategy. An example of integrator is shown in Fig. 9. However, we should pay an attention for avoiding deadlock, if a large number of elements or components are interconnected. So-called I/O-parallel blocks should be used for deadlock free network[5]. Fig. 10 shows the I/O-parallel version of an integrator.

```
PROC P (CHAN OF REAL32
  a,b,c,d,e)
  INT x,y:
      .
      .
    OCCAM code
      .
      .
:
```

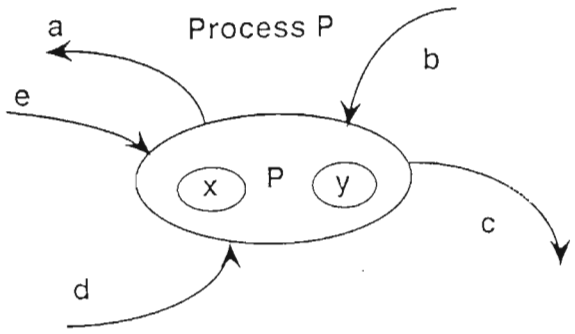


Fig. 6 One process layout

```
PROC main (CHAN OF REAL32
  a,b,c,d,e,f,h,j,k)
  PAR
    L(a,b,c,d)
    M(a,h,j,e)
    N(h,j,k,f)
    O(b,c,g,h)
    P(d,e,f,g)
:
```

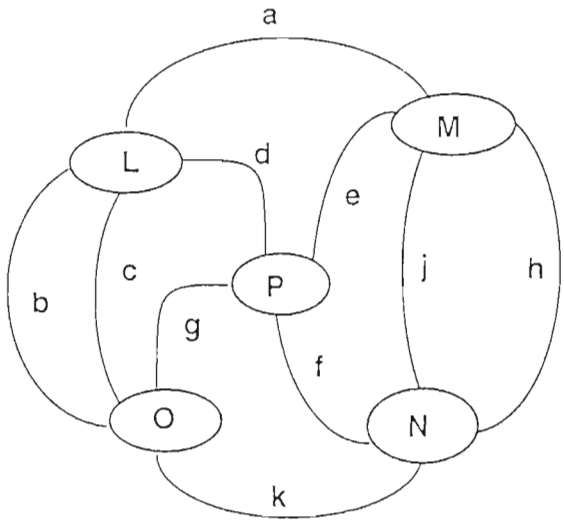
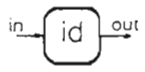


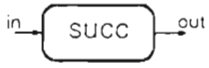
Fig. 7 Multi processes layout

identity



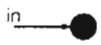

```
PROC id (CHAN OF INT in, out)
  WHILE TRUE
    INT x :
    SEQ
      in ? x
      out ! x
:
```

successor



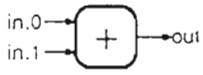

```
PROC succ (CHAN OF INT in, out)
  WHILE TRUE
    INT x:
    SEQ
      in ? x
      out ! x + 1
:
```

black hole



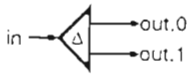

```
PROC black.hole (CHAN OF INT in)
  WHILE TRUE
    INT any:
    in ? any
:
```

plus




```
PROC plus (CHAN OF IN
T in.0, in.1, out)
  WHILE TRUE
    INT x.0, x.1 :
    SEQ
      PAR
        in.0 ? x.0
        in.1 ? x.1
      out ! x.0 + x.1
:
```

delta



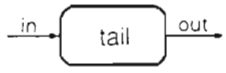

```
PROC delta (CHAN OF
INT in, out.0, out.1)
  WHILE TRUE
    INT x :
    SEQ
      in ? x
      PAR
        out.0 ! x
        out.1 ! x
:
```

(used for broadcasting)

prefix


```
PROC prefix (VAL INT n,
             CHAN OF INT in, out)
  SEQ
    out ! n
    id (in, out)
:
```

tail




```
PROC tail (CHAN OF INT in, out)
  SEQ
    INT any :
    in ? any
    id (in, out)
:
```

Fig. 8 Elementary processes

```
PROC int (CHAN OF INT in, out)
  CHAN of INT a,b,c:
  PAR
    delta (a, out, b)
    prefix (0, b, c)
    plus (in, c, a)
:
```

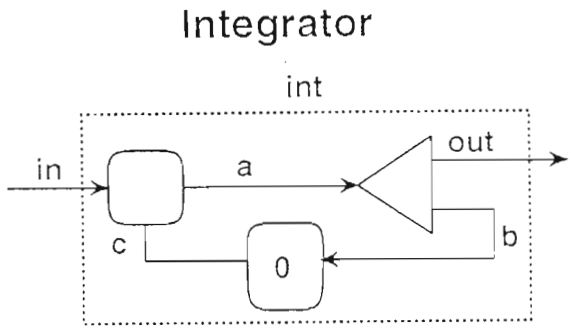


Fig. 9 Integrator component

```
PROC int (CHAN OF REAL32 in,out,init)
  REAL32 x,t :    <- STATE
  SEQ
    init ? x; t   <- INITIALIZE STATE AND TIME
    WHILE TRUE
      PRI ALT
        init ? x;t
          SKIP
        TRUE & SKIP
          REAL32 dx :
          SEQ
            PAR
              in ? dx     <- Input in parallel
              out ! x     <- with output
            x:= x + (dx * t)
:
```

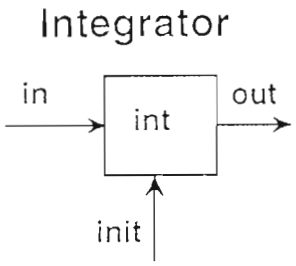


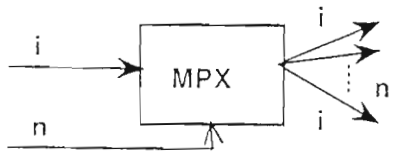
Fig. 10 I/O-parallel integrator

4.2 Components for fuzzy controller

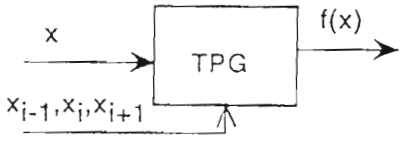
In the case of the fuzzy controller described in the previous section, we can define all basic components as follows.

- Name of the component : Function of the component
  Description of the function
  < inputs
  > outputs
  ≪ preset values (*optional*)
  with graphical description (*see right*)

- MPX : Multiplexer
  to distribute n times copies of the input
  < 1 input
  > n outputs
  ≪ n

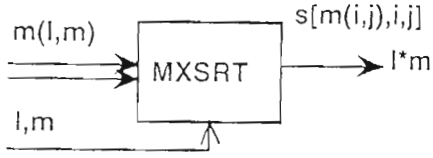


- TPG : Triangle Pulse Generator
  to transform $f(x)$ from x
  where x is an input and y is output
  defined by a unit triangle pulse $f(x)$
  (whose height is 1)
  < 1 input
  > 1 output
  ≪ x-coordinates of left, middle and right edges

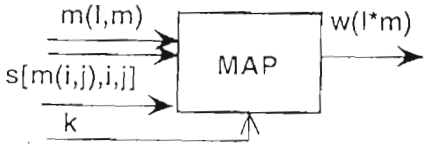


- MIN : Minimum Detector
  to choose minimum value from two inputs
  < 2 inputs
  > 1 output

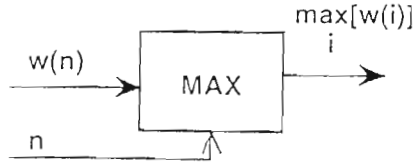
- MXSRT : Matrix Sorter
  to sort $l \times m$ matrix values with its position information
  < a $l \times m$ two-dimentional array
  > a $l \times m$ array of packet
  containing the value, i position
  and j position in the input matrix
  ≪ l and m



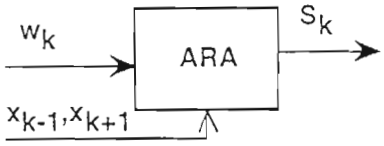
- MAP : Mapping Transformer
  to map MIN($\tilde{e_i}, \tilde{e_j}$) to MXSRT(k)
  < a $l \times m$ two-dimentional array
  > a $l \times m$ array
  ≪ k



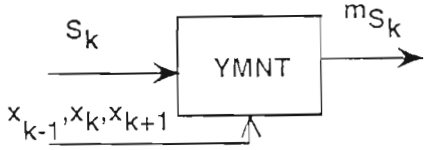
- MAX : Maximum Detector
  to choose maximum value from a n array
  < a n array
  > 1 output
  ≪ n



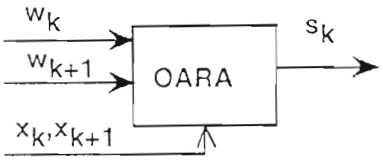
- ARA : Area of a Triangle
  to calculate the area of a triangle
  < height of the peak
  > area of the triangle
  ≪ x-coodinates of left and right edges



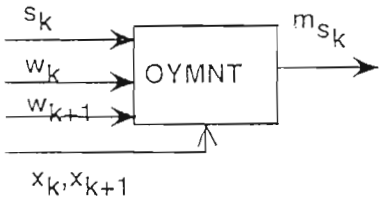
- YMNT : Moment of Area of a Triangle
  to calculate the moment of area of a triangle against the y-coordinate
  < area of the triangle
  > moment of area of the triangle
  ≪ x-coordinates of left, peak and right edges



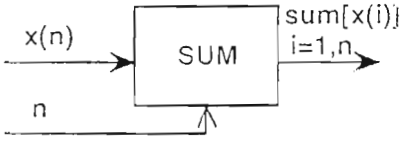
- OARA : Area of Overlapped Triangle
  to calculate the area of the overlapped triangle between two adjacent triangles
  < heights of two adjacent triangles; w_k and w_{k+1}
  > area of the overlapped triangle
  ≪ x-coordinates of peaks of adjacent triangles



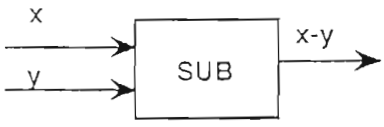
- OYMNT : Moment of Area of Overlapped Triangle
  to calculate the moment of area of the overlapped triangle between two adjacent triangles against the y-coordinate
  < area of overlapped triangle
  < heights of two adjacent triangles; w_k and w_{k+1}
  > area of the overlapped triangle
  ≪ x-coordinates of peaks of adjacent triangles



- SUM : Summation of n inputs
  to calculate summation of n inputs
  < n inputs
  > summation
  ≪ n

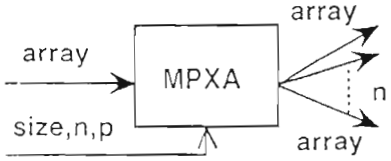


- SUB : Subtracter
  to calculate subtraction
  < two inputs
  > subtraction $x - y$

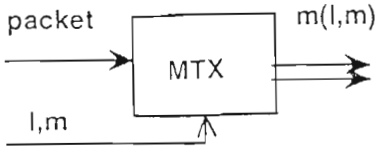


- DIV : Division
  to calculate division
  < two inputs
  > division $x \div y$

- MPXA : Array Multiplexer
  to distribute n times copies of the input array, each of component contains k data packet
  < an array of packets
  > n copies of the input array
  ≪ size of the array, n, number of the packet



- MTX : Matrix Maker
  to make $l \times m$ matrix from a $l \times m$ array input
  $<$ a $l \times m$ packet containing the value, the parameters i and j
  $>$ a $l \times m$ two-dimentional array
  $\ll$ l and m



4.3 Fuzzy controller in basic building block method

The whole controller can be therefore described using the above mentioned components as shown in Fig. 11. In the figure, the inputs are e and $\dot{e}$, for example, the deviation and its derivative as well as the control table (matrix), and the output is u, for example, an driving force of the considering system. All processes describing in the figure can be determined at the same time, which means it can be controlled in real time.

5. Design and Results of Fuzzy Controller

5.1 Design of fuzzy controller for an inverted pendulum

The fuzzy controller was designed for the control process of an inverted pendulum. The system is divided into two parts; the control of the balanced stick and the control of the driving stick.

The membership functions are defined as shown in Fig. 12 and the control rules used are the same with Table 1 except replacing e with θ or ϕ and $\dot{e}$ with their derivatives. The two outputs are added with weights and applied to the motor.

5.2 Results of Fuzzy Controller

A sample result of the fuzzy controller is shown in Fig. 13. This is the case for $\phi = 4$ deg. It works stable for over several hours. It is also found that this controller is stable for impulsive and constant disturbances applied by a finger up to a certain amplitude.

The system is also tested for other control methods such as statefeedback control, neural control and even manual control[6]. It will be also possible to switch these controllers utilizing the parallel properties of transputers.

6. Conclusions

The design of fuzzy controller was done for control an inverted pendulum. To achieve fast computation, transputer networks are introduced and parallel strategy is discussed. The main conclusions obtained are summarized as follows:

1. A generalized fuzzy controller is designed for real-time control.
2. Basic building block method is introduced to realize parallel calculation of the fuzzy controller.

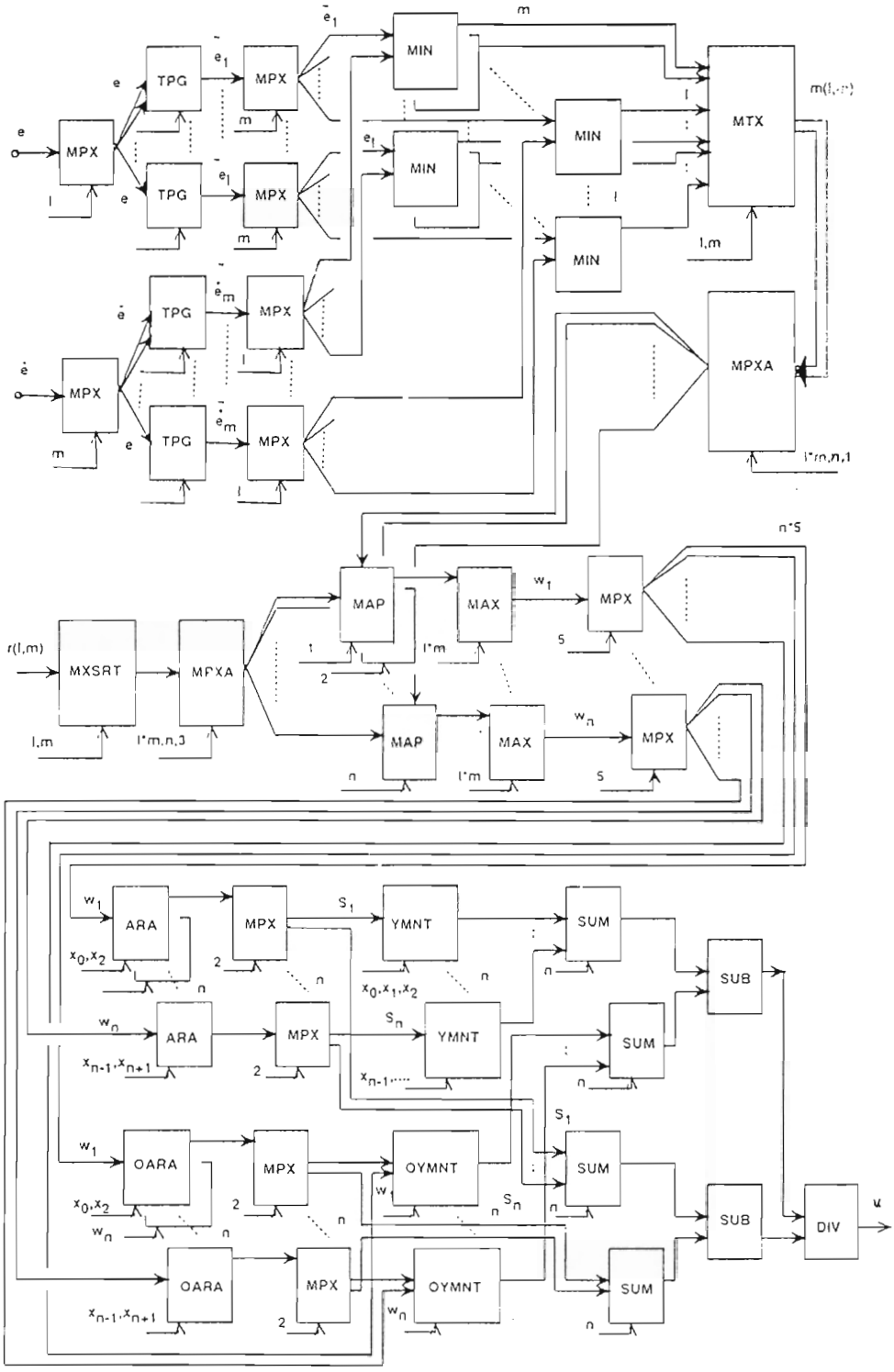


Fig. 11 Fuzzy controller in basic building blocks network

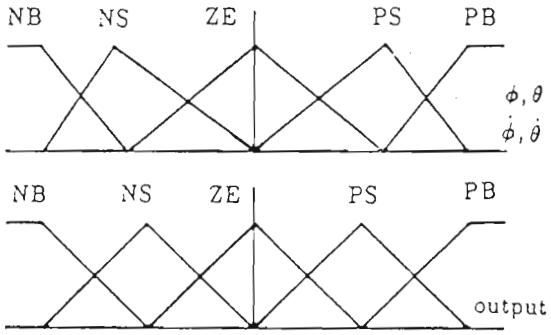


Fig. 12 Definition of membership functions

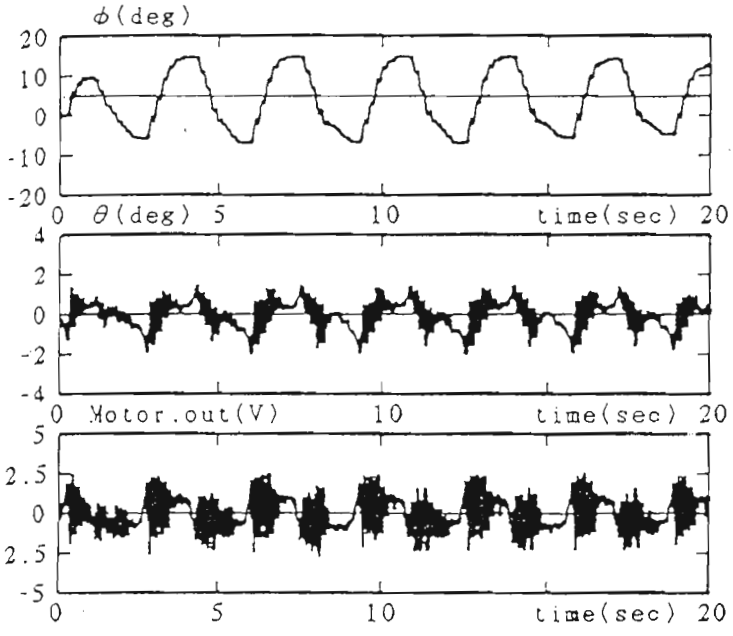


Fig. 13 A sample result of fuzzy controller

3. The results of fuzzy controller was fairly good for its computation time and for control performance.

The following points are the recommendation for the future study.

1. More discussion should be made for scheduling the parallel blocks.
2. More easy and systematic method to build up the block network should be studied (*cf.* **EPICAS**[7]).

Acknowledgements

The authors would like to express their sincere thanks to the colleagues of the BSC Group, Electrical Engineering Department, University of Twente, especially to Mr. Wilbert T. C. van Leunen. They provided various discussions and technical assistances to achieve this project. The first author also wants to send his gratitude to Ministry of Education of Japan for its financial support of his stay.

References

[1] T. Yamakawa and K. Sasaki, "Fuzzy Memory Device", Proc. of the 2nd IFSA Congress, Tokyo, pp. 551–555, 1987.
[2] M. Togai and H. Watanabe, "A VLSI Implementation of Fuzzy Inference Engine: Toward an Expert System on a Chip", Proc. of the 2nd Conference on Artificial Intelligence Applications, Miami Beach, pp. 192–197, 1985.
[3] A. W. P. Bakkers, "Application of Parallel Processing to Robot Control", Proc. of the First International Conference on Robotic Technologies TECHRO'88, Bjuni, Burgas, Bulgaria, 1988.
[4] A. W. P. Bakkers and J. van Amerongen, "Transputer Based Control of Mechatronic Systems", Proc. of the 11th World Congress of IFAC, Tallinn, Estonia, USSR, 1990. (Revised version is available in Occam2 and Transputer Engineering, An Intensive Three-day Course Intended for Engineers, Part 2, pp. 51–67, University of Twente, 1991.)
[5] P. H. Welch, "Emulating Digital Logic Using Transputer Networks (Very High Parallelism = Simplicity = Performance)", Parallel Architectures and Languages Europe, Vol. I, Lecture Notes in Computer Science, Vol. 258, pp. 357–373, Springer Verlag, June, 1987.
[6] K. Hasegawa, J. van Amerongen and W. T. C. van Leunen, "Learning Aspects of a Fuzzy Controller for an Inverted Pendulum", Proc. of the IFToMM-jc International Symposium on Theory of Machines and Mechanisms, Vol. 2, pp.605–610, Nagoya, Japan, 1992.
[7] D. F. G. Nocetti and P. J. Fleming, "Parallel Processing in Digital Control", Advances in Indistrial Control, Springer Verlag, 1992.