# Real-time Reconfigurable Micro-system Based on FPGA and CPLD For Dual-mode PID Control Through Backpropagation Neural Network

Zhuo Ruan [1] Yuzhang Han [2] Jianguo Han [3]
[1..3]Dept. of Electrics Eng. Beijing University of Chemical Technology, P.R China
email : maverick_r_j@msn .com hanjg@mail.buct.edu.cn
[2] Dept. of Computer Science, Univ. of Vienna, Austria, email: yuzhanghan@hotmail.com

*Abstract* -- **In this paper, a practically usable dual-mode control micro-system based on CPLD and reconfigurable FPGA is described. FPGA can be dynamically reconfigured under the control of CPLD to implement two models, Backpropagation neural network model and its training model, both of which are respectively directed to two control modes for industrial produce. One mode is neural network performed automatic control, the other one as human-interfered traditional control. That is, only one single FPGA is reconfigured with multifunction. This technique can be widely applied into other control fields such as the adaptive control in different environments, space-ship control, measuring control in rough situation, and even production control.**

*Key Words*: **Dynamically Reconfigurable, BP Neural Network, real-time, dual-mode (human interfered traditional control and neural network performed control)**

## I. INTRODUCTION

Although adaptive and automatic control are widely used in industries and scientific field, yet their implementation is mainly built on software program in personal or special computer which would enhance the cost of production or scientific research and increase the weight and complexity of the whole controlling system without any doubt. In this paper, FPGA-based dynamically reconfigurable technique is described to implement BP neural network, based on which dual-mode control is also realized in this simplified artificial intelligent control system. Due to high-cost and complex technique used in special chip for dynamical reconfiguring (please see the introduction in Part II of this paper), a kind of simplified but practically realizable and real-time reconfigurable FPGA-based intelligent system is necessary to design, which plays an equivalent role as the realization of dynamically reconfigurable computing. For example, real-time reconfigurable FPGA-based automatic control micro-equipment system is needed in a myriad of environments, first through human interfered traditional control to train a Bp neural network and secondly to implement automatic control under the instruction of completed neural network.

As followed, this improved reconfigurable dual-mode control system is controlled by CLPD with an external Flash-RAM for storing bit-streams corresponding to different function models implemented inside the reconfigurable FPGA. All the merits of this system is followed as:

*1) Relatively high-speed* operation in function model and chip-structure reconfiguration;
*2) small-sized* storing space and *High-speed* calculation**;**
*3) Dual-mode control* based on Neural Network so as to reduce the labor burden in control procedure;
*4) Controlled by CPLD* instead of MCU so as to enhance the parallelity and reconfiguration speed of the whole system.

The implementation of this technique involves BP neural network model implementation with VHDL, configuring data transmission between the micro-system and computer, in which we consider a SPRAM-based reconfigurable FPGA: Altera FPGA APEX 20k family and CPLD MAX7000 family. Thus we also choose Quartus II as CAD software for design and simulation.

## II. PROSPECTIVE FEASIBILITY ON TECHNIQUE

Since Xilinx new PR (partial reconfigurable) FPGA (XAPP290) is displayed in 2002, it is totally possible to apply the kind of FPGA chips in dynamically reconfigurable FPGA-based system, when it is available. PR FPGA is composed of two logic parts: Fixed logic and reconfigurable logic parts. For this matter, we could possibly configure the fixed logic part to simulate the function of controller and then design the reconfigurable logic part as BP-NN model and its training model. When reconfiguration occurs, the fixed part can still operate normally, as in [1]. If this chip is applied, it is totally possible that the reconfiguring speed would be reduced to less than 20ms.

## III. FUNCTION IMPLEMENTATION OF FPGA-BASED AND CPLD-CONTORLLED SYSTEM

Firstly, PPS (Passive Parallel Synchronous) configuring mode for single FPGA chip is considered in this improved micro-system, which allows FPGA to be reconfigured within high speed on real time by CPLD, whose performance is totally better than that to be controlled by MCU, because more pins and logic resources can be utilized by user than those of MCU and MCU never ever can operate parallelingly like CPLD. Figure 3.1 shows the structure and signal connection of PPS mode, as in [2], [3], [4]. After all chips' powering up, CPLD configures or reconfigures FPGA in system with Flash-RAM in which configuring bit-streams are stored in advance through serial port from PC to CPLD.

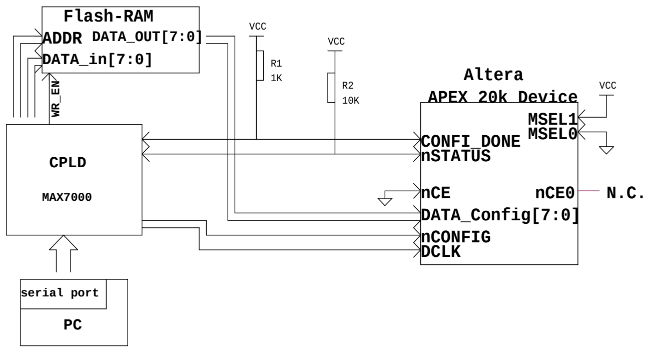


Fig. 3.1. PPS Configuring Mode

Secondly, the specific architecture of the whole system and signal connections are shown in Figure 3.2 to explain the system's operation principles:

In the beginning, all the FPGA configuring files are stored in Flash-RAM through the communication between CPLD and PC, when signal "wr_en" is set to "1". And later the micro-system can normally work without the aid of computer.

a. CPLD sets "wr_en" to '0' and receives original data from A/D, all of which are obtained from human-interfered traditional controlled network.
b. Flash-RAM used by CPLD to configure FPGA as BP neural network training model with the aid of special signals.
c. After "en_tr" is released to "1", one group of input data is sent to FPGA. When that "flag_in" is set to "1" is detected, CPLD resets "en_tr" to end data sending.
d. CPLD then repeats step c till N.N training is finished with a large number of groups of original data.
e. "flag_end" and "empty" signal from CPLD are referred as handshake signal and control FPGA to output neural network weight values, all of which is to be stored in empty space of Flash-RAM through CPLD, when "wr_en" is released to "1".
f. "int_sig" is to be set to "1", if weights transmission is ended. Then CPLD starts to reconfigure FPGA as BP neural network model and clears "wr_en" to "0".
g. Next "en_0" is set to "1" by CPLD, which indicates that CPLD enables Flash_RAM to send weight values and parameters back to FPGA.
h. Detecting " flag_0" is set to "1", CPLD starts sending digital controlling values to FPGA, also setting en_1 to "1". Having finished it, "flag_1" is set to "1" by FPGA and then FPGA starts automatic or non-human control stably through the output controlling vector from itself based on completed neural network model when "en_2" is set to "1" .
i. FPGA would repeat step g and h till the state of controlled objective is changed by external setting and then the system would restart neural network training in the interference of human so as to reduce controlling error.

**Note** to Figure 3.1.2:

1) Some control logic circuits connecting A/D with CPLD in the block graph above are omitted, since these are not the main point of this paper. Also we define "bus_1" to transmit both address and data in different times, while "bus_2" is referred to send or receive only data.
2) The parallel configuring data ports of FPGA can be used as common data ports for weights and threshold values transmission, after FPGA is successfully reconfigured.
3) Due to the pins fixed on PCB, the signal such as "en_tr" and "en_0", "flag_0"and "flag_in" in different function models should share the same pins, when reconfiguration occurs.

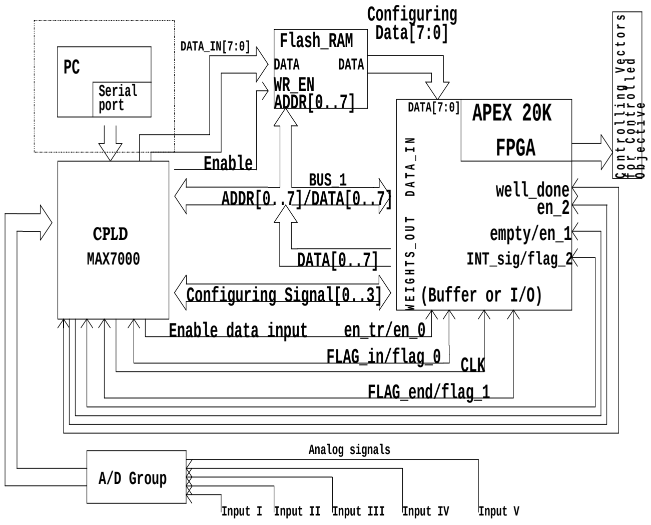


Fig. 3.2 System Architecture

## IV. BACKPROPAGATION NEURAL NETWORK ALGORITHMS IMPLETMENTATION

The FPGA-based system implements two functions respectively, one as operating BP neural network training model on line and the other operating BP neural network model mentioned above.

As discussed, Fig. 4.1 shows the structure of this BP neural network, including one input layer, one hidden layer both with four dots and one output layer with only one dot, as in [5].

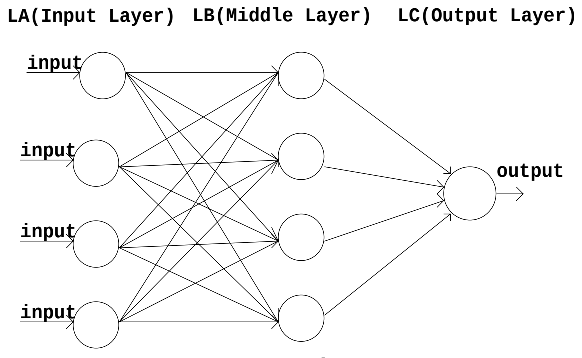


Fig. 4.1 Bp Architecture

### A. BP Neural Network Model

The output functions of LB layer and LC layer are respectively expressed as in (1), (2),

$$b_r = f(\sum_{i=1}^{4} W_{ir} \bullet a_i + T_r) \quad [r=1,2,3,4] \tag{1}$$

$$c_j = f(\sum_{r=1}^{4} V_{rj} \bullet b_r + \theta_j) \quad [j=1] \tag{2}$$

Here, we define $f()$ as Sigmoid function, expressed in (3),

$$M/(1+\exp(-N \bullet X)) \quad (3)$$

N and M both are constant, as in [6].

## B. *BP Neural Network Training Model*

Based on the thought of back allocation of final error, BP neural network can be trained and adjusted through these formulas as followed as in [6]:

a. Give the initial values to $W_{ir}$ , $V_{rj}$ , $T_r$ , $\theta_j$ at random.

b. Start operation with each group of original data ( $A^k(a_1,a_2,a_3,a_4)$ , $c^k$ ), according to the following steps.

S1. Input $A^k$ and calculate them from LA to LC, expressed in (4), (5),

$$b_r = f(\sum_{i=1}^{4} W_{ir} \bullet a_i + T_r) \quad [r=1,2,3,4] \quad (4)$$

$$c_j = f(\sum_{r=1}^{4} V_{rj} \bullet b_r + \theta_j) \quad [j=1] \quad (5)$$

S2. Calculate the error between output value $c_j$ and measuring data $c^k$, expressed in (6),

$$d_j = (1-c_j) \bullet c_j \bullet (c^k - c_j) \quad (6)$$

if considering M and N to be 1.

S3. Error back-allocation in LB Layer, expressed in (7),

$$e_r = (1-b_r) \bullet b_r \bullet (\sum_{j=1} V_{rj} \bullet d_j) \quad (7)$$

if considering M and N to be 1.

S4. Weights and threshold values ( $V_{rj}$ , $\theta_j$ ) adjustment between LB layer and LC layer, express in (8), (9),

$$V_{rj} = V_{rj} + \alpha \bullet b_r \bullet d_{j=1} \quad (8)$$

$$\theta_j = \theta_j + \alpha \bullet d_{j=1} \quad [0 < \alpha < 1] \quad (9)$$

if considering M and N to be 1.

S5. Weights and threshold values ( $W_{ir}$ , $T_r$ ) adjustment between LB layer and LA layer, expressed in (10), (11),

$$W_{ir} = W_{ir} + \beta \bullet a_i \bullet e_r \quad (10)$$

$$T_r = T_r + \beta \bullet e_r \quad [0 < \beta < 1] \quad (11)$$

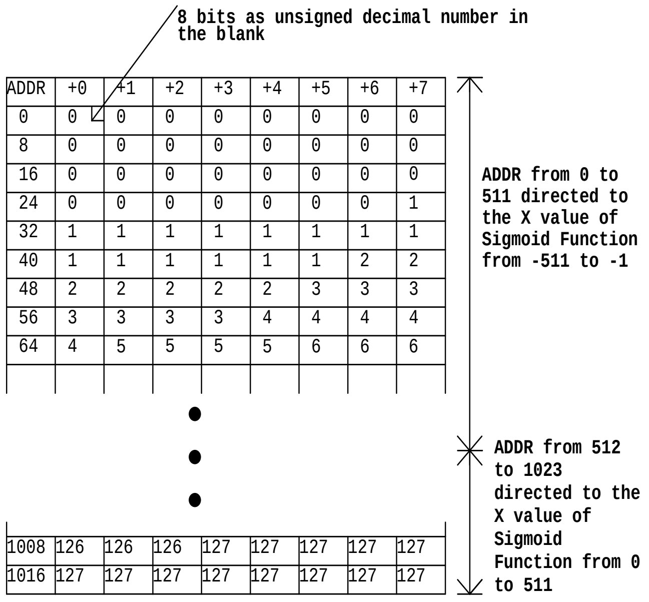
if considering M and N to be 1.

c. Repeat the steps until $d_j$ is small enough or equal to Zero.

In accord with the bit-length of data in the system, firstly we define M= $2^{10}/10$ =102.4, N= $2^8 - 1$ and change '1' in (6), (7) with 127 and also amplified the range of $\alpha$ and $\beta$ to the integers between 0 and 127. With these changes FPGA can easily operate integer-calculation instead of real-number calculation. Next, ROM in the EAB (Embedded Block) of FPGA is adopted to store all the discrete value of the Sigmoid function mentioned above, which can be generated to "*. MIF" files through C or MATLAB language, see Fig. 4.2. In such case, only to input the address (that is the X value of Sigmoid function) of the embedded ROM, the output of ROM can be used as Y value of sigmoid function. More important, this way not only saves a lot logic resource from the implementation of linearized sigmoid function, but also effectively takes the advantage of EAB resource inside FPGA.

**Note:**

Quartus II can easily compile "*.MIF" files into configuring files for FPGA as the initial values for EAB ROM. That is, when FPGA has been reconfigured, its EAB ROM can be automatically filled with the initial values stored in the "*.MIF" files.


8 bits as unsigned decimal number in the blank
ADDR from 0 to 511 directed to the X value of Sigmoid Function from -511 to -1
ADDR from 512 to 1023 directed to the X value of Sigmoid Function from 0 to 511

| ADDR | +0 | +1 | +2 | +3 | +4 | +5 | +6 | +7 |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 16 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 24 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 32 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 40 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 2 |
| 48 | 2 | 2 | 2 | 2 | 2 | 3 | 3 | 3 |
| 56 | 3 | 3 | 3 | 3 | 4 | 4 | 4 | 4 |
| 64 | 4 | 5 | 5 | 5 | 5 | 6 | 6 | 6 |
| | | | | | | | | |
| 1008 | 126 | 126 | 126 | 127 | 127 | 127 | 127 | 127 |
| 1016 | 127 | 127 | 127 | 127 | 127 | 127 | 127 | 127 |

Fig. 4.2 "*. MIF" file's Format in Quartus II

## V. RESOURCE AND TIMING ANALYSIS OF FPGA-BASED FUNCTION IMPLEMENTATION

### A. *Resource Analysis of FPGA*

In this real-time reconfigurable system, EP20K200FC484-2XV device (Family APEX 20k) is selected and this device includes 8320 logic cells and 10588 registers which is sufficient to implement BP neural network model or its training model. If we use two FPGA chips to implement this system, BP neural network model will cost about 3092 logic cells and 73% of its embedded memory, while its training model will cost 6700 logic cells and 73% of its embedded memory. That is, the usage of reconfigurable FPGA-based system saves approximately 3700 logic cells, if the two models mentioned above would turns into one model and it is implemented in only one APEX II chip (it includes about 16600 logic cells) or in the other case this system could save exactly one chip of EP20K200FC484-2XV device, if the two function models are implemented with two FPGA chips. In sum, the practical plan of reconfigurable FPGA-based and dual-mode control system takes good use of FPGA internal logic cell resource and EAB. The resource usage information can be easily informed, if Quartus II is chose to design FPGA and CPLD.

## B. *Timing analysis on FPGA-reconfiguring Procedure and BP Operation*

*1) Reconfiguring Procedure*

PPS configuring mode using CPLD is utilized in the system mentioned above, which can offer more stable data stream than PS (passive serial) mode and thereby reduce the error that happens during configuring and reconfiguring procedures. In addition, when FPGA is within user-mode, the weights and threshold values is to be stored in the Flash-ROM and also is to output to FPGA when it is reconfigured. Thus the choice of synchronous one-byte-output Flash-ROM will be more efficient in the system operation than synchronous one-bit-output Flash-ROM.

PPS timing waveform is shown in Figure 5.2.1, according to which the time needed by FPGA reconfiguration can be estimated, as in [3], [4] and also the time estimation of address output and other operation within CPLD, please see the formula expressed in (12) and its instruction.

$$T = T_{POR} + T_{CFG} + T_{CF2CK} + T_{DCLK}/2 + T_{DLCK} \bullet (V-1) \bullet 8 + T_M + T_{E\&S} \qquad (12)$$

$T_{POR}$ (Time for powering up and resetting) takes about 60~80ms for powering and resetting on inside FPGA;

$f_{DCLK} = 30MHz$ (Maximum 33.3MHz);

$V$ is defined as bit volume of configuring file (*.rbf). In both BP neural network model and its training model, $V$ is equal to 208KB;

$T_{E\&S}$ (Time for error/status checking) takes about 1 to 3.4ms;

$T_{CFG} + T_{CF2CK}$ costs 61us (See hardware parameters);

$T_M$ , as the cost of CPLD during the reconfiguring procedure, is defined as "0";

From the analysis above, the minimum value of T has been estimated to be 116.7~136.7ms, relatively short time for reconfiguration and even shorter than that of PS mode, so called real-time reconfiguration.

**Note** to the calculation of $T_M$ :

Different from MCU, CPLD can totally work on a myriad of tasks with any internal or external interruption, called high-speed parallelity. For this matter, CPLD can reconfigure FPGA, check the feed-back signal from FPGA and operate other function of this system at the same time. And thus, $T_M$ is considered to be Zero. That is, reconfiguration procedure is free of any other interference and interruptions.

*2) BP and Its Training Operations*

The general rule for global clock signal definition of FPGA is that its frequency should be higher than the maximum delay between register and register; otherwise the error will happen in the data-transmitting procedure. So timing analysis inside FPGA from QuartusII Software is shown below:

a. In BP neural network model, to best test the system, standard CLK signal with T=90ns (>81.912ns, the maximum register-register delay) is input to FPGA. Thus, soft sensing with just one group of data takes about 8~12ns.
b. When FPGA is configured as BP neural network training model, the same timing analysis is adopted, and therefore CLK signal with T=130 ns (>125ns, the maximum register-register delay) is input to FPGA. In such case, neural network training with single one group of data costs about 16~20ns.

In sum, the period of standard CLK should be changed when reconfiguration occurs in these two various models. In our design, the simple method of frequency division is used to implement it with VHDL.

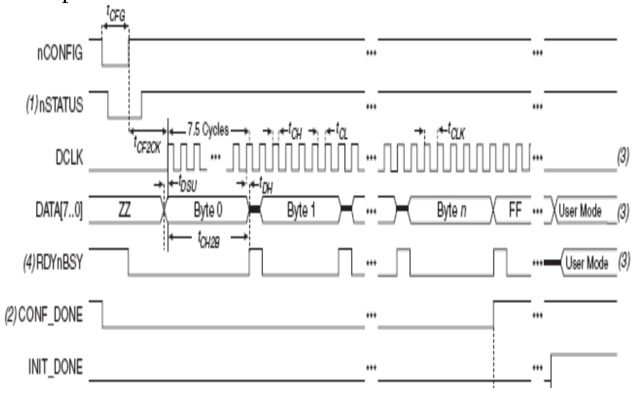


Fig. 5.2.1 Reconfiguring Timing Wave of PPS Mode

## VI. DUAL-MODE DISCRETE PID CONTROL AND ITS APPLICATION

Discrete PID control method is wildly used in industrial controlling procedure. Now, its implementation counted on the reconfigurable FPGA-based system mentioned above is displayed in this section. Firstly please see the discrete PID model in Figure 6.1, as in [7] and also expressed in formula (13) and the expression of the increment of control vector in (14),

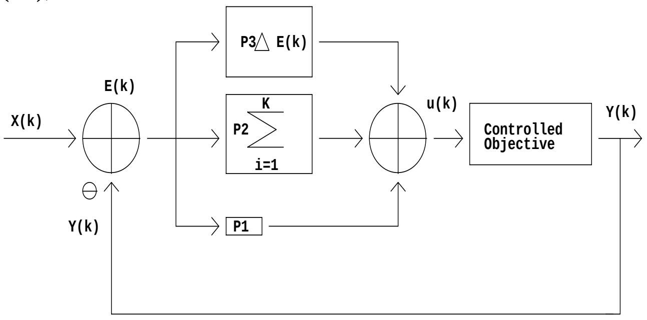


Fig. 6.1 Block Graphic of Discrete PID

$$u(k) = p_1 e(k) + p_2 \sum_{i=1}^{k} e(k) + p_3[e(k) - e(k-1)] \qquad (13)$$

$$\Delta u(k) = u(k) - u(k-1) \qquad (14)$$

Deducted from (13) and (14), Formula (15) as the mathematic model is utilized to implement PID control based on BP neural network,

$$u(k) = u(k-1) + p_1[e(k) - e(k-1)] + p_2 e(k) + p_3[e(k) - 2e(k-1) + e(k-2)] \qquad (15)$$

In the implementation of this model, the system controls the producing procedure under the human's interference, when working in the first mode of dual-mode control. At the same time, according to Formula (15), e(k), e(k-1), e(k-2), measurable disturbing parameter and u(k) as five input of BP training model are all used to train a BP neural network. Once the training procedure has been completed, FPGA should be reconfigured to establish BP neural network and control production automatically, that is the second mode of dual-mode control. Of course, if external environment influences the controlling error to be amplified, the automatically controlling mode can be cancelled immediately, so then the micro-system re-operates reconfiguration and trains the neural network again. The whole procedure is shown with flow-chart in Figure 6.2.

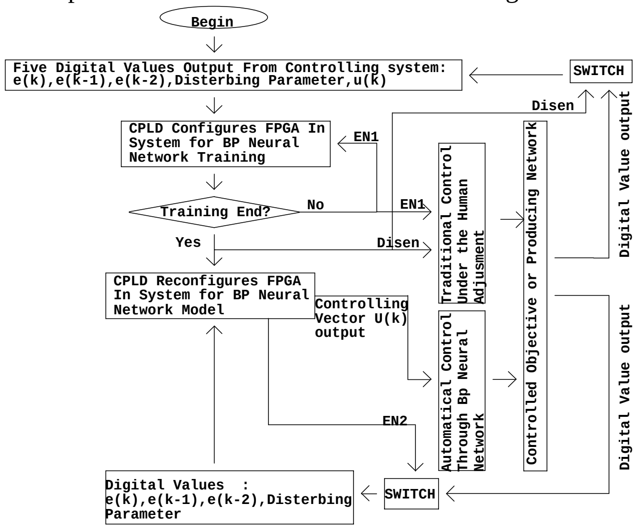

Begin
Five Digital Values Output From Controlling system: e(k),e(k-1),e(k-2),Disterbing Parameter,u(k)
SWITCH
Disen
CPLD Configures FPGA In System for BP Neural Network Training
EN1
Training End?
No
EN1
Yes
Disen
Traditional Control Under the Human Adjusment
Controlled Objective or Producing Network
Digital Value output
CPLD Reconfigures FPGA In System for BP Neural Network Model
Controlling Vector U(k) output
Automatical Control Through Bp Neural Network
EN2
Digital Value output
Digital Values : e(k),e(k-1),e(k-2),Disterbing Parameter
SWITCH

Fig. 6.2 Dual-mode Control Procedure

## VII. CONCLUSIONS

In this improved real-time micro-system described above, FPGA is reconfigured in system by CPLD which can greatly enhance the parallelity of the system and reconfiguring speed in the dual-mode control procedure, compared with MCU- and FPGA-based real-time reconfigurable system. On the other hand, dual-mode control implementation on BP neural network is a good way to convert all the controlling information into weights and threshold values and stored in BP neural network on real time so that the controlled network or objective can be adjusted automatically and the labor burden could also be largely reduced. Once the external disturbing parameter is largely amplified, the neural-network training can be restarted again so as to adjust controlling precision. Of course, in our future work, a more effective neural network model will be design or sought and finally realized on this real-time micro-system. For sure, some new algorithm of artificial neural network will solve some problems in our current design, such as slow-speed convergence, partial limited efficiency.

## VIII. REFERENCES

[1] XILINX Corp., “Partial Reconfiguration--New for 4.2i”, 2002

[2] Altera Corp. “*APEX 10K Programmable logic Device Family Data Sheet*”, 2004

[3] Altera Corp. “*Configuration Handbook*”, 2004

[4] Altera Corp. “*Configuration Element Data Sheet*” , 2000

[5] Michael. A. Abibe, “*The Handbook of Brian Theory and Neural-Network, Page 144”, 2003*

[6] Zengliang liu, “*Fuzzy Logic and Neural Network*”. B.U.A.A Press, 1996

[7] Guofang Zhang, Shusheng Gu, Mingshun Wang, *“Computer-based control system”*, Press of Metallurgy Industry, *2004*

[8] Paul Hasler and Jeff Dugger, “analog VLSI Implementations of Neural Networks”, *The Handbook of Brain Theory and Neural Network*, 2003

[9] Dan Hammerstrom, “Digital VLSI for Neural Networks”, *The Handbook of Brain Theory and Neural Network*, 2003