Implementation of Evolvable Fuzzy Hardware for Packet Scheduling Through Online Context Switching

Ju Hui Li, Meng Hiot Lim and Qi Cao
School of EEE, Block S1
Nanyang Technological University
Singapore 639798
Tel: 65-67905408

Abstract: - Real-time cell scheduling is a promising area for the application of *evolvable hardware* (EHW). In this paper, we describe an intrinsic evolvable and online adaptive EHW for solving the packet switching problem. Based on the coding and evolution scheme, we refer to it as *evolvable fuzzy hardware* (EFH), an extension of our proposed *evolvable fuzzy system* (EFS) [1, 2] and *reconfigurable fuzzy inference chip* (RFIC) [3]. EFS is a good framework for online adaptation. RFIC on the other hand is a hardware architecture which supports context switching or online reconfiguration. By combining the advantages of the two, we can achieve an intrinsic evolvable and online adaptive EFH. The whole scheme is illustrated through application in solving the packet switching problem and the advantages are highlighted.

Key-Words: - Evolvable Fuzzy Hardware, Evolvable Fuzzy System, Cell Scheduling, Reconfigurable Fuzzy Inference Chip, On-line Adaptation, Intrinsic Evolvable Hardware

1. Introduction

For modern networks, various services are currently supported. Voice, video, file transfer, etc., are some of the important services provided by Internet, broad area network, and local area network. The IPv6 (Internet Protocol version 6) is now starting to mature. This new type of communication protocol can connect instruments all over the world into a single network. It has significant advantages over the IPv4, which is currently widely used. With the emergence of the advanced protocol, new services not currently available may be needed in the future. This tendency requires network nodes to accommodate for greater flexibility. The multiplexer is a very important network component for modern network infra-structure. It is used mainly to provide bandwidth sharing of high-speed link for network terminal equipments or network inter-nodes. There are many scheduling algorithms available, such *as first-in-first-out* (FIFO), *weighted round-robin* (WRR), *virtual clock* [11], *dynamic weighted priority scheduling* (DWPS) [10] and *deficit round-robin* (DRR) [12]. Each of these schemes has its own strengths and weaknesses. In this paper, we investigate the application of *evolvable hardware* (EHW) and some of its unique characteristics in cell scheduling. Other than emulating *general processor sharing* (GPS) or designing the algorithm in a packet-by-packet mode, EFH can provide a platform of designing a scheduling system directly from the standpoint of QoS requirements.

Evolvable hardware (EHW) was introduced in 1992 by Higuchi *et. al.* [4], combining evolutionary scheme and reconfigurable hardware device to solve problems. EHW is suitable for working under a dynamically changing environment. It is therefore promising in some situations whereby the operating environment besides being dynamic, is also unknown or unpredictable. Many research works till now deal with extrinsic EHW which carries out evolution by means of a software model and only downloads the elite into the reconfigurable hardware. On the contrary, intrinsic EHW tries to do away with the software simulation model. In an ideal case, both the evolution and reconfiguration can be carried out within the EHW such that it can trace and adapt to the changing operating environment to maintain good system performance. Due to practical implementation issues, intrinsic EHW achieved so far cannot entirely break away from the need for an external computational platform [5,6,7].

The complexity pertaining to the implementation of intrinsic EHW depends very much on the application areas. One potential area of application for intrinsic EHW is packet switching network. There have already been some works done on EHW application in ATM cell scheduling [8,9]. In their works, the authors presented schemes to solve this kind of problem using functional EHW. The functional EHW systems successfully evolved a circuit which can achieve similar service performance as traditional scheduling scheme. However, some significant limitations make the system not suitable for practical applications. The most significant limitation is that the system cannot evolve intrinsically and adapt online. We believe intrinsic evolvable and online adaptive EHW for such kind of application is viable according to our previous work [1,2,3]. The combination of these works can be termed as *evolvable fuzzy hardware* (EFH).

The rest of this paper is structured as follow. Section 2 discusses the application problem and the evolution scheme in the proposed EFH. The proposed EFH relies mainly on a reconfigurable fuzzy device termed as *reconfigurable fuzzy inference chip* (RFIC). In Section 3, our proposed RFIC will be outlined. In Section 4, the simulation results of EFH on ATM cell scheduling will be given. Section 5 will then present the conclusion and give directions on some future work.

2. Evolution Scheme

Multiplexer is a very important component in packet switching network. It adopts time division sharing scheme to provide bandwidth sharing for different flows. In this paper, two classes of cell flows with fixed packet size are mainly considered. *Class1* refers to cell flow which is sensitive to cell delay, for example, *Constant Bit Rate* (CBR). On the other hand, *Class2* refers to cell flow such as non-real-time *Variable Bit Rate* (nrt-VBR) which are not sensitive to cell delay but to cell loss. The whole system structure of EFH for packet scheduling is described as in Fig.1. BUF# and MP unit are normal components as in traditional scheduling scheme. BUF# in Fig.1 is used to store cells waiting for time slots. MP is the multiplexing unit for allocating time slots to *Class1* and *Class2*. In this problem, the size of BUF# is fixed. It is also assumed that the capacity of OUT channel is the same as the capacity of the input channels, 155.52Mhz. The performance of the multiplexing system can be described by the *quality of service* (QoS) which includes *Class1* cell delay, delay variation and *Class2* cell loss.

Due to the system's complexity, the evolution granularities such as transistor, gate, functional unit are not very suitable for evolving powerful circuits in a very short time. In order to overcome such a problem, we can adopt fuzzy rules as the evolution granularity. In Fig.1, TB# is the training buffer for acquiring training data online. RFIC block is a specific implementation of *reconfigurable fuzzy inference chip* described in [3]. Evolution Module is a hardware component to perform GA operations in order to evolve the desired fuzzy rule sets. Scheduling Model is used to simulate the behavior of MP unit on the cell flow stored in TB#. In a simple way, Scheduling Model can be an embedded scheduling system. It includes two components, MP simulator unit and RFIC unit. Every fuzzy rule set evolved by the Evolution Module can be evaluated by downloading onto the RFIC in the Scheduling Model. By incorporating the Scheduling Model, the evolved fuzzy rule sets can be evaluated without affecting the system's operation.

Because of the unpredictablility of cell flow, it is difficult to train a system that works equally well for all cell flow scenarios. The rationale for choosing an appropriate data set for training can be justified based on the principle of "locality". If we assume that the time period is very small, the cell flow of the next time period will be very similar to the current time period. Based on this justification, it is suitable to employ TB# of finite length to collect training data for the evolvable system.

In order to control the scheduling behavior using fuzzy rule set, we define two fuzzy variables c_1 and c_2 to describe the status of BUF1 and BUF2 respectively. c_1 is the ratio of *Class1* cell rate and the capacity of the OUT channel. c_2 is the ratio of the number of empty units in BUF2 and the length of BUF2. The membership functions for c_1 and c_2 are as shown in Fig.2. A fuzzy rule set for cell scheduling is as shown in Table 1. In Table 1, *T* means that the OUT channel is allocated to *Class1* and *F* means that *Class2* will be sent through the OUT channel. For genetic coding, the string "12222,11122,11112,11112,11111" represents the fuzzy rule set in Table 1. The value "1" corresponds to *T* and "2" corresponds to *F*. A "0" in the chromosome indicates there is no fuzzy rule defined for the corresponding input situation.

The rule set in Table 1 is designed based on human knowledge and intuition. It can be employed as a core rule set in the EFH system. In this system, the evolution system tries to derive a good chromosome for a specific situation, not necessarily an optimal one. After a fixed number of generations, if a better chromosome is derived, the current working chromosome is replaced immediately. Otherwise, the current rule set continues to be applicable. In order to guarantee a certain level of minimum acceptable performance, the system relies on the core rule set shown in Table 1 as a default startup rule set.

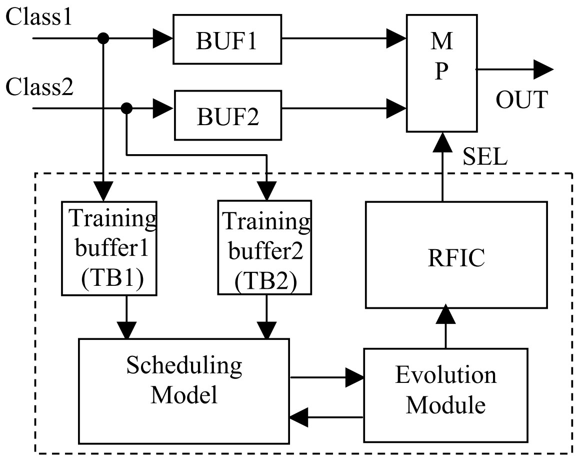

Fig.1 **Adaptation framework for EFH**

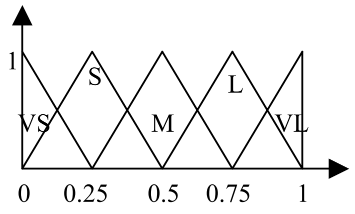

Fig.2 **Membership function**

Table 1 **A fuzzy set for ATM cell scheduling**

c_2 \ c_1	*VS*	*S*	*M*	*L*	*VL*
VS	*T*	*F*	*F*	*F*	*F*
S	*T*	*T*	*T*	*F*	*F*
M	*T*	*T*	*T*	*T*	*F*
L	*T*	*T*	*T*	*T*	*F*
VL	*T*	*T*	*T*	*T*	*T*

In Table 1, each cell entry is interpreted as a fuzzy rule that maps the inputs c_1 and c_2 to the output SEL. For example, the first fuzzy rule represents "if <c_1 is *VS*> and <c_2 is *VS*> then <*SEL* is *T*>". This means that the strength of firing is taken as the minimum of the degrees of matching between the two inputs and the antecedents of the rule. The fuzzy aggregation is carried out by averaging the fuzzy conclusions derived for all the rules. There are two aggregation results for the *T* and *F* outputs. The larger value determines the final crisp conclusion.

In this system, we fix the rule number to be a small number to keep the search space manageable. The fitness function adopted in the proposed EFH system is described by Eq.1, 2 and 3.

$$F = \kappa - \left| AveDelay - \lambda \times DelayFactor \right| \quad (1)$$

$$AveDelay = {}^{1}\!/_{\tau} \times \sum_{i=1}^{\tau} m(i) \quad (2)$$

$$DelayFactor = \rho \times \upsilon \quad (3)$$

In Eq.1, κ is a very large constant. λ is an adjustable parameter to indicate the desired *Class1* cell delay. *AveDelay* represents the average cell delay suffered by the cell flow stored in TB1. *DelayFactor* is a constant used as a reference for scaling the value of λ based on the desired *Class1* cell delay. In Eq.2, τ is a variable denoting the number of *Class1* cell units in TB1 sent during evaluation. $m(i)$ is the waiting time of the *i*th cell in TB1 before being sent. In Eq.3, ρ is a constant corresponding to the time required to send a cell through the output channel. The value of ρ depends on the bandwidth capacity of the output channel. υ denotes the size of TB#.

3. Hardware Implementation

In the system architecture of Fig.1, there are two RFIC blocks, which are the key components. One RFIC block is for packet scheduling control and the other is included in the Scheduling Model block. For the Scheduling Model, chromosomes need to be evaluated within a very short time period. The duration of the evaluation can significantly affect the system performance. The concept of RFIC is illustrated by the block architecture in Fig.3 [3]. There are 4 main blocks that make up the RFIC, FIM (*fuzzy inference map*), CMU (*context memory unit*), AEM (*address encoding mechanism*) and OAM (*output aggregation mechanism*). In Fig.3, *k* indicates the number of input bits. *p* refers to the width of the data bus. *v* and *w* are the sizes of the linguistic term sets for Input1 and Input2 respectively. *m* depends on the size of the linguistic term set for the output variable. In the application for cell scheduling, *k* is 5, *p* is 5, *v* and *w* are 5 and *m* is 1.

In the RFIC, the FIM is a critical part for accommodating the various fuzzy contexts. It stores all the fuzzy conclusions for every input situation. For a rule set of 25 rules as in Table1, the FIM block is divided into 25 groupings called *partition blocks* (PB). Each PB stores the fuzzy conclusions covering all possible input situations. For example, $PB_{<1,1>}$ maintains a mapping for the fuzzy antecedents such as "if <c_1 is *VS*> and <c_2 is *VS*>". The digitized inputs c_1, c_2 and genes of the working fuzzy rule set stored in the context register of CMU can be combined together to address a specific memory location within this PB. The content in every PB is determined by the adopted inference scheme.

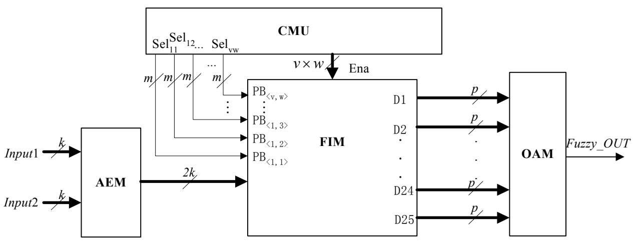

Fig.3 **Scheme of RFIC**

The content of the context register in CMU is the working fuzzy rule set. CMU generates Ena and Sel signals for the PBs based on the context register. The Ena signal is used to decide which PB is active. The Sel signal combines with the output of AEM to form the address signals to access specific memory location in every active PB.

AEM is the address generator unit. It is responsible for generating major address signals for the PBs of the FIM. Its output is derived by combing the two digitized inputs. OAM is a circuit to fulfill fuzzy aggregration. It includes two parts of aggregation for *T* and *F* respectively which is composed of Ave_2 blocks and Ave_3 block to perform 2-averaging and 3-averaging [3]. The inputs of OAM are from every PB. If one PB is active, it outputs the data stored in the unit specified by the output of AEM and the output signal Sel of CMU to OAM, otherwise it outputs 0.

4. Simulation

In order to demonstrate the viability of EFH for solving packet scheduling problem, we carried out simulations on the cell flow scenario in Fig.4. The simulation results of EFH will be compared with *first-in-first-out* (FIFO) and *dynamic priority scheduling* (DWPS) [10]. FIFO is a very general scheduling scheme which can achieve very good balance of cell losses but very poor *Class1* cell delay. DWPS is an improvement of *round-robin* scheduling scheme. It can adapt to the changes of the cell flow.

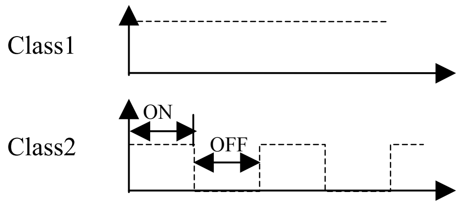

Fig.4 **Two classes of cell flows**

For the cell flow scenario, *Class1* is the cell flow with cell bit rate of 155.52MHz while *Class2* is the cell flow also with cell bit rate of 155.52MHz. The difference between *Class1* and *Class2* is that *Class1* has constant cell bit rate while *Class2* has 2 ms ON period and 2 ms OFF period. For the simulation, the length of BUF# is 100 units, the length of TB# is 300 units, the population size is 12 and the generation number is 14. The simulation results of EFH (λ =0.35), FIFO and DWPS are as given in Fig.5, 6, 7, 8, and 9. From Fig.5 we can see that EFH can achieve lower *Class1* cell delay than FIFO and DWPS. From Fig.6 it can be seen that the balance of cell losses by using these three systems are good, which means that none of them biases much on *Class1* cell flow. Fig. 7, 8 and 9 present the delay distribution suffered by *Class1*. In Fig.7, it can be seen that most of the *Class1* cells suffer delay between 270μS and 400μS for the EFH. The probability of cell delay above 400μS approaches 0, indicating a small delay variation for *Class1* packets. The delay for FIFO in Fig.8 is mainly concentrated around two bars, 270μS and 530μS. The probabilities for these two values are both very significant. The delay for DWPS in Fig.9 is in the range of 270μS and 470μS. But besides the high probability of 270μS, the delay for DWPS also concentrates around 400μS. From these comparisons, it is concluded that EFH can provide smaller delay variation than the other two algorithms.

Another very good property of EFH is that the system's performance can be adjusted very intuitively by decreasing or increasing the value of λ in Eq.1. The smaller the value, the smaller the *Class1* cell delay. This property cannot be achieved conveniently using traditional scheduling methods. The tunability of EFH can be seen from the simulation results. Results of simulation for different λ values are as shown in Fig.10 and 11. In Fig.10 and 11, when λ is 0.4, the *Class1* cell delay and *Class1* cell loss are very small. Accordingly, the *Class2* cell delay and *Class2* cell loss are very big. If good balance of *Class1* cell loss and *Class2* cell loss is desired, a bigger value can be assigned to λ. In Fig.10 and 11, both the *Class1* cell loss and *Class2* cell loss are moderate when λ is 0.6. In the case when the QoS of *Class2* needs to be significantly emphasized, the value of λ can be further increased. The larger the λ, the better the QoS of *Class2*. The QoS for *Class2* when λ is 0.8 is the best in Fig.10 and 11.

In principle, *Class1* cell delay can be adjusted in the range from 0 to $\rho\times\upsilon$ if λ is within 0 and 1. This means that *Class1* cell delay has a very wide range of tunability. In another word, according to the correlation between cell loss and cell delay, *Class1* cell loss and *Class2* cell loss can also be tuned to a very wide range when λ is adjusted. According to the fitness function, a rough *Class1* cell delay can be estimated after deciding the value of λ. On the other hand, if one knows the satisfactory *Class1* cell delay requirement, the value of λ can also be approximated. This delay tunability is very useful to give some priorities to other flows under the case when a flow does not require much on small delay but has requirement of small delay variation. This philosophy can take better advantage of the network bandwidth and resource without much effects on network QoS.

5. Conclusion and Future Works

In this paper, we described the packet switching problem and the evolution scheme of EFH for solving this problem. We also addressed the implementation issues of EFH by describing the RFIC, a reconfigurable fuzzy inference chip that is able to handle real-time context switching. For the demonstration of the viability of EFH, the simulation results for the packet switching were given which shows that the EFH system can perform as well as other scheduling schemes, namely FIFO and DWPS. The EFH can also provide better properties than other schemes in terms of flexibility. The more advantageous aspect of EFH is the tunability. This was also demonstrated through simulation results. From the research work shown in this paper, it is evident that EFH is very suitable for controlling data flow. It can adapt to the changes of the cell flows and effectively control the multiplexing of the cell flow. Our future work will focus further on studying the characteristics of EFH system and its hardware implementation.

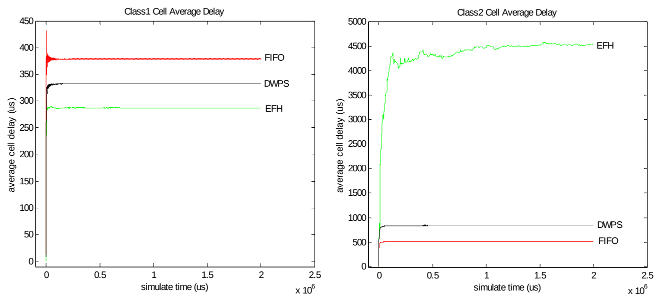

Fig.5 **Cell delay of Class1 and Class2**

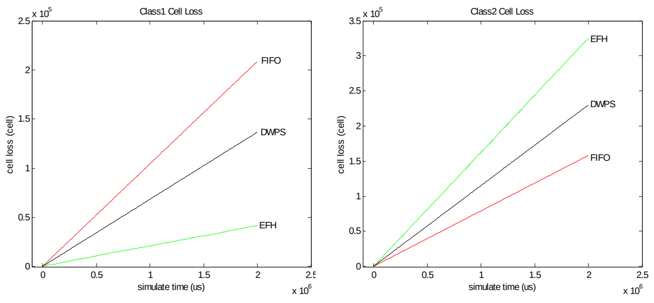

Fig.6 **Cell loss of Class1 and Class2**

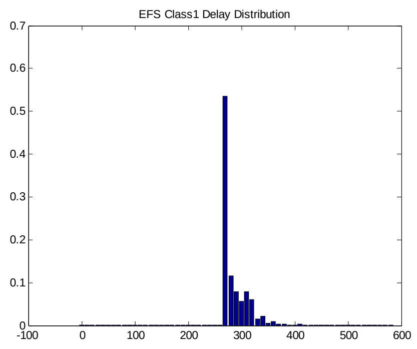

Fig.7 **Class1 delay distribution for EFS**

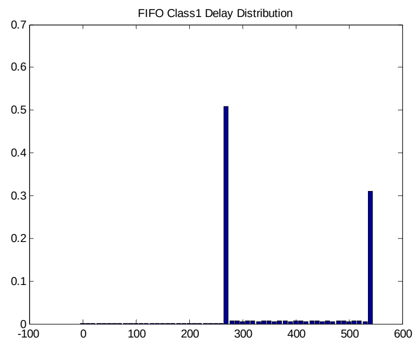

Fig.8 **Class1 delay distribution for FIFO**

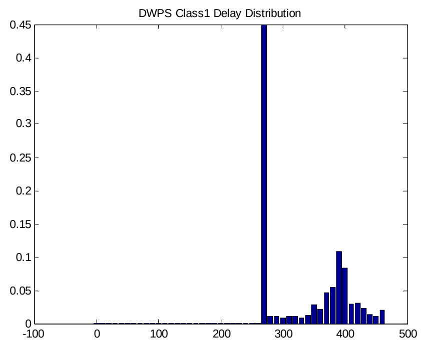

Fig.9 **Class1 delay distribution for DWPS**

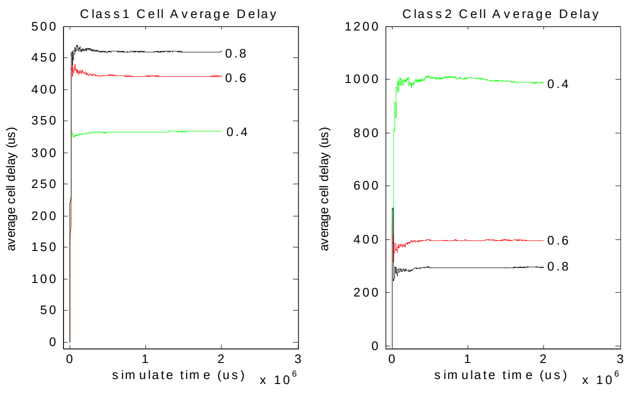

Fig.10 **Cell delay tunability**

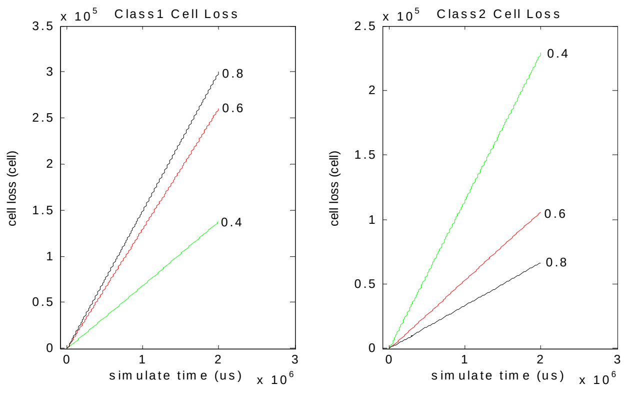

Fig.11 **Cell loss tunability**

References:

[1.] J.H. Li and M.H. Lim, "Evolvable fuzzy system for ATM cell scheduling", LNCS2606, (*Proc. of ICES 2003*), pp. 208-217, 2003

[2.] J.H. Li and M.H. Lim, "A Framework for Evolvable Fuzzy Hardware," Special session on Evolutionary Computing for Control and System Applications, ICONIP/SEAL/FSKD, 18-22 Nov 2002, Singapore.

[3.] Q. Cao, M.H. Lim, J.H. Li and W.L. Ng, "A Context Switchable Fuzzy Inference Chip", submitted to IEEE Transactions on Fuzzy Systems.

[4.] T. Higuchi, T. Niwa, T. Tanaka, H. Iba, H.D. Garis, and T. Furuya, "Evolving Hardware with Genetic Learning", Proc. of Simulation of Adaptive Behaviour, pp. 417-424, MIT Press, 1992.

[5.] M. Iwata, I. Kajitani, Y. Liu, N. Kajihara, and T. Higuchi, "Implementation of a Gate-Level Evolvable Hardware Chip", LNCS 2210 *(Proc. of ICES2001)*, pp. 38-49, 2001

[6.] J. Langeheine, K. Meier and J. Schemmel, "Intrinsic Evolution of Quasi DC Solutions for Transistor Level Analog Electronic Circuits Using a CMOS FPTA Chip", Proc. NASA/DoD Conference. on Evolvable Hardware, pp. 75 –84, 2002

[7.] J. Langeheine, S. Folling, K. Meier and J. Schemmel, "A CMOS FPTA Chip for Intrinsic Hardware Evolution of Analog Circuits", Proc. of 3rd NASA/DOD Workshop on Evolvable Hardware, pp. 172-175, Long Beach, CA, USE, 2001. IEEE Press.

[8.] W. Liu, M. Murakawa and T. Higuchi, "ATM Cell Scheduling by Function Level Evolvable Hardware", Proc. of ICES1996, LNCS 1259, pp.180-192, Springer-Verlag, 1997.

[9.] W. Liu, M. Murakawa and T. Higuchi, "Evolvable Hardware for On-line Adaptive Traffic Control in ATM Networks", Genetic Programming 1997, Proc. of the Second Annual Conference, pp. 504-509, Morgan Kaufmann Publishers, 1997.

[10.] T. Lizambri, F. Duran and S. Wakid, "Priority Scheduling and Buffer Management for ATM Traffic Shaping", The Seventh IEEE Workshop on Future Trends of Distributed Computing Systems 20, Dec. 1999.

[11.] L. Zhang, "VirtualClock: A new traffic control algorithm for packet switching networks," in Proc. ACM SIGCOMM'90, Aug.1990, pp.19-29.

[12.] M.Shreedhar and G. Varghese, "Efficient fair queuing using deficit round robin," in Proc. ACM SIGCOMM'95, 1995, pp.231-242.