Performance improvement of dynamic buffered ATM switch

Abdulhamit Subasi a,*, Etem Koklukaya b

a Department of Electrical and Electronics Engineering, Kahramanmaras Sütçü İmam University, 46100 Kahramanmaras, Turkey
b Department of Electrical and Electronics Engineering, Sakarya University, 54187 Sakarya, Turkey

Received 16 April 2003; received in revised form 1 November 2004; accepted 26 January 2005
Available online 26 April 2005

Abstract

Performance of ATM networks depends on switch performance and architecture. This paper presents a simulation study of a new dynamic allocation of input buffer space in ATM switching elements. The switching elements are composed of input and output buffers which are used to store received and forwarded cells, respectively. Efficient and fair use of buffer space in an ATM switch is essential to gain high throughput and low cell loss performance from the network. In this paper, input buffer space of each switching element is allocated dynamically as a function of traffic load. A shared buffer pool is provided with threshold-based virtual partition among input ports, which supplies the necessary input buffer space as required by each input port. The system behaviour under varying traffic loads has investigated using a simulation program. Also, a comparison with a static allocation scheme shows that the threshold based dynamic buffer allocation scheme ensures an increased network throughput and a fair share of the buffer space even under bursty loading conditions.

© 2005 Elsevier Ltd. All rights reserved.

Keywords: ATM networks; Dynamic buffered switch; Dynamic threshold updating; Performance evaluation

* Corresponding author. Tel.: +90 344 251 23 15x282; fax: +90 344 251 23 42.
E-mail addresses: asubasi@ksu.edu.tr (A. Subasi), ekaya@sau.edu.tr (E. Koklukaya).

0045-7906/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.compeleceng.2005.01.001
1. Introduction

Telecommunication networks based on the Broadband Integrated Services Digital Network (B-ISDN) can support a variety of multimedia services such as voice, video and data. The Asynchronous Transfer Mode (ATM) is envisaged as the basic transfer mode for implementing the B-ISDN and is based on cell switching technique. To guarantee the speed and the quality of service (QoS) of CBR services, the cells have a fixed length of 53 octets and are known as cells. Since the switching technique is based on the cell switching principle, the events (arrival of cells and their onward transmission) occur only at regularly spaced points in time [1,2].

In the last decade or so, considerable research effort has gone into the development of ATM switches. Many architectures have been proposed over the years [3–11], whereby hardware implementation often turned out to be the major design constraint, due to the high speeds at which these switches should operate. Concurrently, a large number of articles were devoted to the performance analysis of the proposed architectures [10–22]. In many instances, simplifying assumptions had to be made to make the analysis feasible and/or tractable. Mostly, these assumptions not only pertain to the switch itself, but also to the traffic offered to the switch [15].

High-speed cell switches have been proposed to utilize buffers to prevent arriving cell from losing in ATM networks. Buffers can be located at input ports or output ports. They can be designed to be single in each input port or shared for all input ports. It was seen that completely shared buffering achieves optimal throughput-delay performance. However, there is a fairness problem in input queued shared-buffer switches that a single input port can take over most of the buffers, preventing cells destined for less utilized ports from gaining access under overload condition [14].

High-speed cell switches have been proposed to utilize buffers to prevent arriving cell from losing in ATM networks. Buffers can be located at input ports or output ports, and can be designed to be single in each input port or shared for all input ports. It was concluded that input queuing and completely shared buffering achieve optimal throughput-delay performance. However, there is a fairness problem in input queued shared-buffer switches that a single input port can take over most of the buffers, preventing cells destined for less utilized ports from gaining access under overload condition [14]. To overcome this problem, we add an output buffer to the switch module in addition to input buffer.

Two kinds of buffer sharing restriction have been proposed in shared-buffer switches. The first one named as the threshold scheme puts threshold on buffers that should be available to any individual input port. In this method, a threshold is assigned to each input port. An arriving cell is accepted only if the current queue length is smaller than the threshold. Only comparators and counters are needed in the threshold scheme. Thus, it is really simple to implement the threshold scheme [4,14,15].

The second kind of buffer sharing restriction is pushout scheme. An arriving cell is allowed to enter the buffer as long as the shared-buffer is not full. When the buffer is full, an incoming cell is allowed to enter the buffer by taking the place of a cell that is already in the buffer. For the fairness reason, the cell located at the head of the longest queue among the input ports is selected to be discarded first. It should be noticed that the pushing and pushed cells might belong to different input ports. Indeed, the performance of the pushout scheme is better than the threshold scheme. The performance of the pushout scheme is optimal because no input queue is ever starved for space and no space is ever held idle while some queue desires more. The drawback of the pushout
scheme lies in that it is too complicated to implement the buffer management in the switch. Because when the shared-buffer is full, writing a cell into a queue involves complicated cell sequence preservation [4,14,15].

Choudhury [4] proposed a dynamic threshold scheme. The key idea in the dynamic threshold scheme is that the maximum permissible length is proportional to the unused buffering for any individual queue at any instance of time. In fact, the dynamic threshold scheme uses part of the threshold scheme and part of the pushout scheme.

In this paper, we propose a dynamic input buffer allocation with dynamic threshold scheme in a cell switched ATM network. By using this scheme, we achieve a superior throughput and a reduced number of lost cells that the fixed input buffer suffers from at high loads. We have developed a simulation program of the proposed scheme and compared it to the static fixed size input buffer. This paper is organized as follows. In Section 2, some major issues about ATM switch are addressed; in Section 3, the dynamic buffer management scheme is presented; simulation study is shown in Section 4; results and discussion are provided in Section 5; finally, we draw our conclusions on this study in Section 6.

2. Dynamic buffer model description

An ATM switch contains a set of input ports and output ports, through which it is interconnected to users, other switches, and other network elements. It might also have other interfaces to exchange control and management information with special purpose networks. Theoretically, the switch is only assumed to perform cell relay and support of control and management functions. It will be useful to adopt a functional block model to simplify the discussion of various design alternatives.

Generally speaking, ATM switches are network elements that support functionalities such as cell transport, connection control and network management. An ATM switch must provide the same functions of routing and buffering as traditional switches, but at a much higher speed. Studies in recent years into the architecture of ATM switches have considered several buffering schemes [8].

Under bursty traffic, statistical multiplexing of ATM cells lead to severe cell loss. Large buffers are costly, create excessive cell delay and increase switch cost. Thus an appropriate number of buffers must be allocated in a switch and an optimal utilization of them must be made. The goal is to accommodate more incoming traffic cells from various sources and smooth out the burst arrival rate while limiting the overhead of the switch to a predetermined size. A feasible solution is a dynamic buffering strategy, which applies a threshold-based sharing policy to the buffers and a priority-based cell push-out policy to either buffered or incoming cells [15,23,24].

Choice of a threshold: A number of buffer allocation schemes have been investigated.

(1) Complete sharing accommodates all cells if the storage space is not full.
(2) Complete partitioning divides permanently the entire space to all input ports.
(3) Partial sharing reserves a certain number of buffers and the rest are partitioned among the input ports. When traffic is bursty and the aggregated cell arrival rate exceeds the link transmission rate then the shared buffers in the switch are quickly filled, which makes cell loss
unavoidable. This raises the question of how to choose the cells to be dropped, while maintaining fairness of cell loss among all input ports, when the switch buffer is full. Under symmetric traffic equal traffic on all ports, it has been proved that the optimal policy in shared memory switches with two output ports is the “push-out type with threshold”. In other words, whenever the buffer is not full, an incoming cell should be accepted. Once the buffer is full, a cell of type 1 (or of type 2) is allowed in the switch and a cell of type 2 (or of type 1) is pushed out, if the number of type 1 (or type 2) cells is below some threshold, such as T1 (or T2 for type 2), where buffer size = T1 + T2. Under asymmetric traffic, however, for a system with N ports, the optimal policy is still an open question. We are motivated to impose a threshold on buffer allocation to logical input queues, thus preventing one queue from draining out all the space while throughput is very low due to smaller buffers on other outgoing links [15,25–27].

The ATM network considered consists of n identical stages, each having $a^{n-1}$ switching elements (SEs). Each SE is an $a \times a$ crossbar, with input buffer modules (dynamic length) to store conflicting cells and output buffer (length two) modules to store unacknowledged cells. Fig. 1 shows a sample model of a $a \times a$ proposed shared-buffer switch structure. As shown in figure, an SE is composed of a input and output ports. At each input port, a logical input buffer module (IBM) of length $B$ is provided to hold forwarded cells from previous stages. Each output port includes an output buffer module (OBM) of length equal to two [1].

The following assumptions are made in our model:

1. Cells are assumed of a fixed length and are generated randomly at the beginning of a clock period. They are admitted at the IBM of stage 0 only if there is place in the corresponding IBM.
2. The switch uses input queuing and shared buffer, i.e. each input port of the fabric has a logical queue, but these queues all share the same fabric memory. Each input port has a logical buffer of size $B$. An arriving cell destined for input port I joins the input queue I from head to

![Fig. 1. Proposed shared-buffer ATM switch with input and output buffer.](image)
When the queue length of input queue I reaches $B$ (i.e. queue I is full), the control logic will choose a shortest-length input queue among the other $a - 1$ input queues, said as input queue J. Now, input queue J is used to allocate the new arriving cell destined for the input port I, and the new arriving cells destined for input port I will borrow the storage place from input queue J and join to input queue J from tail to head sequentially. The control logic sets a connection between input queue I and input queue J. When there is no empty space left in both input queue I and input queue J, an arriving cell destined for input port I will be discarded and an arriving cell destined to input port J will take the place of the last cell stored in the queue and destined for input port I.

3. The time needed to allocate space to an IBM is assumed negligible with respect to the cycle timing.

4. The switching network operates synchronously at the rate of $\tau$, which is equal to the sum of the time taken to select a cell and forward it to the next stage IBM.

5. At the beginning of each cycle, cells waiting to be forwarded in the IBMs are checked, and if two cells are destined to the same OBM, they are randomly resolved.

6. Cells generated by the cell sources are uniformly distributed over all SEs' IBMs. A source of cells is connected to each port of stage 0 IBMs. The rates at which cells are generated at the source nodes determine the traffic load.

7. The output ports of the SEs in the final stage of the switching network are connected to sink nodes. Hence, any cell that arrives at those output ports is accepted by the sink nodes with a probability of one.

8. A cell is transferred from the OBM of stage $k$ to the IBM of stage $(k + 1)$ when the buffer space is available at stage $(k + 1)$ IBM and a cell is available for transmission at stage $k$ OBM. A cell is available at stage $k$ OBM when there are two cells in the OBM, or there is one cell and it is not held up.

9. A cell is accepted by the $(k + 1)$th stage IBM in a cycle if it is not full or it is full, and a cell in that IBM is able to move forward to an OBM.

10. The SEs are $a \times a$ switches which have the capability of connecting an input to one of the output ports labelled 0 and $a$, depending on the state of the corresponding value of the destination address.

11. The operation of loading a cell from the IBM to the OBM, and its transmission to the next stage, takes place in parallel with the same cycle. If that cell is accepted in the next stage IBM, it does not contribute toward the delay experienced by the cell in the network.

Intuitively, in order to adapt to different mixes of incoming traffic, the threshold should be changing dynamically, unlike static partitioning. Some kinds of traffic applications, such as video and audio, are very sensitive to delay and cell loss. Under a bursty condition, the more traffic of such types the switch can accommodate, the better the QoS will be. Yet, the statistical nature of the significant part of the traffic, its burstiness, and its variability combine to pose difficulties in distributing buffers of fixed size. Therefore, in a statistical multiplexing environment with heterogeneous traffic, before arriving at an optimal threshold, it is quite important to define congestion detection criteria for dynamic threshold updating. The complexity and the performance of the switch mechanism play a critical role in preventive congestion control. Our algorithm detects the congestion and updates the corresponding threshold.
3. Design objectives of the proposed buffer allocation

Consider an \( a \times a \) ATM switch with input-queued shared-buffer as shown in Fig. 1. The switch uses input queuing and shared buffer, i.e. each input port of the fabric has a logical queue of length \( B \), but these queues all share the same fabric memory. An arriving cell destined for input port \( I \) joins the input queue \( I \) from head to tail sequentially. When the queue length of input queue \( I \) reaches \( B \) (i.e. queue \( I \) is full), the control logic will choose a shortest-length input queue among the other \( a - 1 \) input queues, said as input queue \( J \). Now, input queue \( J \) is used to allocate the new arriving cell destined for the input port \( I \), and the new arriving cells destined for input port \( I \) will borrow the storage place from input queue \( J \) and join to input queue \( J \) from tail to head sequentially. The control logic sets a connection between input queue \( I \) and input queue \( J \). When there is no empty space left in both input queue \( I \) and input queue \( J \), an arriving cell destined for input port \( I \) will be discarded and an arriving cell destined to input port \( J \) will take the place of the last cell stored in the queue and destined for input port \( I \). Because input port \( I \) has borrowed the buffer from input port \( J \), the last cell destined for input port \( I \) occupied the storage place of input queue \( J \) will be discarded when an arriving cell is destined for input port \( I \).

When the queue length of input queue \( I \) decreases to \( B \) or the queue length of input queue \( J \) increases to \( B \), the control logic assigns two different shortest-queue-length buffers among the other \( a - 2 \) input ports to input port \( I \) and \( J \). In the proposed buffer management scheme, when the queue length of each input buffer increases to \( B \) or decreases to \( B \), the control logic assigns alternative shortest-queue-length buffer for storage.

Take a simple example for illustration. In a \( 4 \times 4 \) ATM switch, the size of each queue is six. There are six, two, four and three cells stored in input queue 1, 2, 3, and 4, respectively. When an arriving cell is destined for input port 1, the control logic will select a shortest-length input queue, which is input queue 2 to store the arriving cell. The arriving cell is allocated at the tail of the input queue 2. If there are four arriving cells destined for input port 2, these four arriving cells are stored in input queue 2, and the cell belonging to input port 1 but stored in input queue 2 is discarded. However, because the queue length of input port 1 and 2 reaches six, the control logic selects two shortest-length queues (input queue 3 and input queue 4) to input port 1 and input port 2. Thus, the proposed buffer management scheme improves the cell loss performance because each input buffer can borrow the storage space from other input buffers [14].

In our model, the control and the data information are transmitted in parallel. In this model, the control signals exchanged between consecutive stages are changed, whereby the control logic consists of three state machines, each one working independently of the other, dealing with cell transmission, acknowledge reception and acknowledge transmission for accepted cells. At the beginning of every cycle, the control logic determines which cell is to be forwarded to the next stage, which is forwarded regardless of the state to the next stage input buffer SE. Therefore, in a cycle, a cell forwarded from stage \( k \) output buffer SE is accepted by stage \( (k + 1) \) SE, if its IBM is not full or stage \( (k + 1) \) SE has forwarded a cell from its full IBM to stage \( (k + 2) \) SE. In this way, cells are forwarded without using a control signal in a cycle, providing a faster throughput. However, when a cell forwarded cannot be accepted by stage \( (k + 1) \), it has to be dropped. This would result in cell loss in the event that these cells are not retransmitted. Therefore,
each SE output port has an OBM of length two. As soon as a cell is transmitted to the next stage, the control logic saves the cell in the OBM [12]. The algorithm is shown below:

In each cycle do in parallel:

a. Cell transmission
   {Check the status of the OBM. 
    CASE full: select the unacknowledged cell in the OBM and forward it to the next stage. 
    CASE not full: check the status of the IBM. 
    If a cell to the OBM is available then forward the cell and save the cell in the OBM, 
    Else wait till the next cycle.}

b. Ack transmission for accepted cells
   {If a cell is available at the input port and the IBM is not full or IBM is full but another buffer is 
    available then accept the cell and send an ack to the previous stage.}

c. Ack reception
   {If an ack is received from the next stage then clear the cell awaiting an ack in the previous 
    OBM.}

End parallel do

As mentioned, the OBM is two cells long. As long as the OBM is full, there is always an unacknowledged cell that needs to be retransmitted. Hence, instead of loading a new cell from the IBM, the waiting cell is retransmitted thus eliminating lost cells. The proposed buffer management scheme can prevent a single input port taking over most of the buffer to achieve fairness purpose.

4. Simulation study

A simulation program was designed and developed using C++. A data structure was implemented to model the network connectivity and simulated cells. A cell is characterized by four attributes: (a) destination address; (b) time admitted into the network; (c) time released from the network; and (d) time to set the flag of unacknowledged cells to be cleared. An SE has the following attributes: (a) IBM’s current length; (b) list of cells in each IBM; (c) list of cells in each OBM; and (d) next stage input port number linked to each OBM.

We have made following assumptions during the implementation of simulator. These are as follows:

(1) The $N$ processors generate cells at each stage cycle with probability $q(1)$ (input load).
(2) The destination of each cell at each stage cycle by processors is set randomly by a random number generator to simulate uniform traffic.
(3) If there is a conflict among cells within an SE, one of them is selected randomly.
(4) The throughput and the delay are measured at each output port of the network, and averaged over the network size and simulation time span to get the normalized throughput and the normalized delay of the network.
(5) Cell service at each buffer module is done by using first-in-first-out (FIFO) policy.
The simulation starts by initializing the IBM and OBM for each SE to null. The ATM switching topology is then initialized to implement the data structure of each SE. Other initializations include the simulation clock, which is equal to the clock cycle the switching network operates at, the current pool size used to null, and the current number of simulated cells to zero. We changed threshold updating frequency according to the weight of each traffic factor. These are used to determine new shares of the buffer space. After the initializing step, the simulation loop begins. Each simulation loop includes the following computations:

1. For each SE in the last stage, do the following: if a cell exists in any one of the two OBMs, then remove this cell from the OBM. This simulates the acceptance of cells by the sink node. Set the cell attribute (time released) to the current simulation time in order to calculate the network delay.

2. Check the current number of simulated cells. If this number has exceeded the number specified by the user, and no cells exist in the IBM and OBM in the network, then the simulation should stop. Calculate and display the performance measures (throughput and delay).

3. Check each SE’s OBM. If a cell in any of the two OBMs is flagged to be cleared as a result of acknowledge reception from the next stage, then clear this cell in that OBM. Update the OBM length accordingly.

4. Check each SE’s OBM in the network from the last stage to the first:
   a. If the OBM of that SE is full, this means that an unacknowledged cell exists in this OBM. Select this cell and forward it to the next stage IBM if a place exits in. Set the flag of this cell to be cleared in case a place exist in the next stage IBM (acknowledge reception).
   b. If the OBM is not full: If cells exist in that SE’s IBM: check if two cells from two IBM in that SE are destined to the same OBM. If they are, then randomly select one, and forward it to the destined OBM. If the next stage IBM is full, then clear the forwarded cell from its IBM, and return its memory space to the pool. If the next stage IBM is not full, then admit this cell to the next stage IBM (if there is place in the pool), and clear the cell from the SE’s IBM. Return its memory space to the pool. Cells do not exist in IBM: go to step (5).

5. After all SEs in the switching network are processed in step 4, randomly generate a cell with a random output destination address and admit it to the corresponding IBM if a place exits. Increase the number of simulated cells by one, and save the time the cell is admitted to the network for later analysis.

6. Thresholds of the logical queues for all input ports are equally assigned during the initialization phase.

7. Thresholds are updated every $M$ time slots.

8. Any excess of buffer space from any logical queue goes to the shared buffer pool.

9. A cell accommodation rule (CAR) is applied whenever a cell arrives at the switch. A CAR is needed to make use of the dynamic thresholds. When the buffer pool is not full, any incoming cell should be accepted; otherwise a selective cell admission/drop mechanism is enforced.

10. Increase the simulation clock.

11. Go to step (2).
5. Results and discussion

The principal goal of this work is to find an ATM switch which improves the network performance. A new model was introduced in an ATM environment. In this model, a shared buffer pool provided with threshold-based virtual partition was proposed. Also performances of the static and the dynamic buffered networks have been compared to each other.

A simulated study of ATM networks has been performed and the following terms are used for comparison with the existing models:

- **Normalized throughput**: the average number of cells passed by the switching network per output link per cycle.
- **Normalized delay**: the average number of cycles required for a cell to pass the network per output link per cycle.

Two major factors are important in determining the performance of ATM networks, which are the buffer length and traffic load. As the buffer length is dynamically allocated, it will only contribute to an increase in throughput for various loads as compared to the fixed IBM maximum length. The simulation results are shown in Figs. 2–7. Each graph shows the variation of a given parameter as a function of the traffic load for various IBM lengths and for the dynamic scheme. The input parameters of the simulation program are traffic load, buffer length and number of cells to be simulated. The performance measures include the following: normalized throughput and delay.

Fig. 2 shows the network’s normalized throughput vs. input traffic load. As the network traffic load increases, the normalized throughput increases because of the increased number of cells that are passed by the network as a result of the increased load. However, this increase is saturated at high loads because of the limited input buffer length and increased conflicts. As can be seen, the

![Normalized Throughput vs. Traffic Load](image)
The proposed scheme provides a higher throughput than the threshold scheme and the no control scheme, because of the flexible IBM length which accommodates all cells admitted to the network. Also, as the maximum IBM length is increased beyond a certain value (i.e. 10) all three switch model provide approximately the same normalized throughput.

In Fig. 3, the normalized delay increases with traffic load due to the limited input buffer length and increased conflicts between cells that are destined to the same output port. In the proposed scheme, this delay is higher because more cells can be accommodated in the IBMs as the traffic increases, leading to an increase in waiting delays in the IBMs.

Fig. 4 shows the normalized throughput vs. buffer size for various IBM lengths. Fig. 5 shows the normalized delay vs. buffer size for various IBM lengths. In the proposed scheme, normalized
throughput is better than the threshold scheme and the no control scheme. However, because of the novel architecture in the proposed scheme, the normalized delay is worse than the other two schemes. It is seen that the normalized throughput increases as the buffer size increases. The normalized delay increases as the buffer size increases. The normalized throughput reaches the saturation point very quickly after the six buffer sizes. The increase in normalized throughput is very significant up to the buffer size of 3–4. The normalized delay increases almost linearly with the buffer size. So, adding buffers to a cell switching network can increase throughput.

Fig. 5 compares the normalized delay vs. buffer size for various IB lengths.

Fig. 6 compares the normalized throughput vs. network size for the proposed scheme, the threshold scheme and no control scheme. When the network size $N$ increases, the normalized...
throughput decreases in all models. However, the performance of the proposed scheme is much better than the other two.

Fig. 7 compares the normalized delay vs. network size. In this case, the normalized delay of each input port with no control scheme is better than the threshold scheme and the proposed scheme, because the cell arrival rate for each incoming port increases.

In summary, these results showed that performance improvement is achieved in proposed model as compared to other two models. The network performance in our simulation is measured according to the metrics: normalized throughput and normalized delay. These results are not surprising because the no control scheme constitutes a bottleneck that yields a reduced network throughput and therefore, degradation in the network performance. The proposed scheme will always accept new cells to the IBM s as the network traffic increases, because the buffer pertaining to the IBM s of switching elements is automatically increased upon demand. This is only limited by the maximum allowable buffer size allocated to the switch ports. We also note that the cell loss is achieved for the ATM switch when we add an extra output buffer.

6. Conclusion

In this paper, a novel shared buffer switch with shortest-queue-length buffer management in ATM networks was proposed. In the proposed buffer structure, if the queue length of an input queue reaches to $B$, the control logic chooses a shortest-length input queue among the other $a - 1$ output queues to store the new arriving cell. This proposed structure will decrease the number of out-of-order and lost packets for high-traffic loads. In addition, an increased network throughput and an increased network performance is achieved. Thus, by using the proposed model, a superior and efficient ATM switch is achieved. However, the allocation time needed to allocate a packet space to an IBM is the only drawback, which increases the network delay. The allocation
time increases when multiple IB requests are initiated. However, it may be reduced by using a clustered buffer for each stage, or for the SEs in the same level. This allows parallel buffer allocation, which decreases the allocation time. Thus, the proposed buffer management scheme gives fairness to every input port. Also when we add a buffer at the output port in the switch, the proposed buffer management scheme can prevent packets from being lost. Thus packet lost in the switch is prevented by using output buffers. Besides, compared to the pushout scheme, the proposed scheme is easy to implement in the input-queued shared-buffer ATM switches.

References


Abdulhamit Subasi graduated from Hacettepe University in 1990. He took his M.Sc. degree from Middle East Technical University in 1993, and his Ph.D. degree from Sakarya University in 2001, all in Electronics Engineering. He is an Assistant Professor at Kahramanmaras Sütçü İmam University. His areas of interest are application of neural networks, biomedical signal processing, computer networks and security.

Etem Koklukaya received his B.Sc., M.Sc. and Ph.D. degrees from Istanbul Technical University (1978), Yildiz Technical University (1982) and Istanbul Technical University (1990), respectively. He is with the Department of Electrical and Electronics Engineering at Sakarya University, Sakarya, Turkey. His research areas are intelligent and expert system applications in electronics and biomedical engineering.