Development and Performance Analysis of a Data-Reduction Algorithm for Automotive Multiplexing

Syed Misbahuddin, Member, IEEE, Syed Masud Mahmud, and Nizar Al-Holou, Senior Member, IEEE

Abstract—Automotive multiplexing allows sharing information among various intelligent modules inside an automotive electronic system. In order to achieve an optimum functionality, the information should be exchanged among various electronic modules in real time. Data-reduction techniques are used to send the data over a transmission medium at a high speed. They can be employed in automotive multiplexing systems to improve the information exchange rate among various intelligent modules. Some off-the-shelf data-reduction algorithms have been considered for automotive multiplexing. However, their applications have been limited to text data classes only. This paper introduces a data-reduction algorithm that can be applied to all data classes found in automotive multiplexing, including body- and engine-related data. Detailed performance analysis of the algorithm is presented in this paper. Although this algorithm has been developed to fit in the automotive environment, it can also be applied to nonautomotive applications in which extensive information exchange is performed among control modules via a multiplexing bus. The proposed algorithm uses SAE J1939 as a base protocol. However, it can be used with other automotive multiplexing protocols without loss of generality.

I. INTRODUCTION

The auto industry is planning to introduce a high-speed serial data (HSSD) communication system as a central nervous system for vehicles [1]. All intelligent modules, including electronic control units, smart sensors, and smart actuators, can be connected to the HSSD bus. There is a growing need for information exchange among intelligent modules [2]. The data traffic over the HSSD will increase as the number of modules increases in automobiles. For real-time control, intermodular information must be available to all modules within a short period of time. A delay of information longer than the specified time window may cause degradation in performance. In some cases, this delay may lead to malfunctioning of the vehicle.

Data-reduction (DR) techniques are used to transfer the same amount of information in a relatively shorter interval of time [3], [4]. They can be applied to meet the timing constraints in automotive multiplexing networks and to avoid system malfunctioning. DR techniques have received extensive attention for general applications such as image storage, data storage, and data transmission. However, the applications of DR techniques in automotive multiplexing have received relatively less attention.

Due to their special applications in the automotive environment, automotive multiplexing systems need careful selection of data-reduction algorithms. DR techniques for automotive multiplexing have been investigated by Kempf and Strenzl for automotive body electronics [5], [6]. Michael et al. applied various state-of-the-art data-compression algorithms to automotive multiplexing in an experimental vehicle [6]. From their studies, they concluded that there are six data reduction algorithms that may be considered for automotive multiplexing:

1) simple Huffman coding;
2) adaptive Huffman coding;
3) arithmetic coding;
4) higher order arithmetic coding;
5) textual substitution coding;
6) command data stream reference coding [6].

Huffman coding assigns a relatively shorter bit sequence to the symbols having higher frequency of occurrence, and longer bit sequence to the symbols having lower frequency of occurrence. The main limitation of Huffman coding is the requirement of keeping a copy of the probability table at each node in the automotive multiplexing system. Also, one or more bit inversions during data transmission may cause loss of synchronization at the receiving end. Adaptive Huffman coding is the extension of Huffman code in which the Huffman tree is adjusted on the fly based upon the previously seen data.

In arithmetic coding, first the frequency of each symbol is determined. Once the probability of occurrence of each symbol is known, a range of real numbers is assigned to each symbol. The length of this range is equal to the probability of the symbol. If a symbol has probability 0.1, for example, then the assigned range of numbers will be [0.0 – 0.1]. Following the arithmetic algorithm, a message consisting of a stream of symbols can be represented by a single floating-point number [7]. An extension of arithmetic coding is higher order arithmetic coding, in which the probability of each incoming symbol is calculated on the basis of the context in which the symbols were previously encountered. After determining these probabilities, encoding of arithmetic coding is used. A higher order arithmetic coding scheme requires a large amount of memory at each node [7].

In a textual substitution algorithm, variable length strings of symbols are encoded into a single token. This token is used as an index to a phrase dictionary maintained at the receiving end [8]. This data-compression algorithm has been used to devise another algorithm, CDSR coding. The CDSR coding scheme is
especially designed for automotive multiplexing application. In the CDSR scheme, a reference dictionary is maintained at each node in the multiplexing system. When a message is generated, the reference dictionary at the transmitting side is referred, and a token is generated instead of the actual message. This token indicates the position of the first symbol in the transmitted message and the message length. A copy of the message available at the receiving end is located with the help of the transmitted token. Kempf and Strenzal further investigated the application of common data-stream coding and proposed a communication protocol to overcome the drawback associated with it [5].

Among all six data-compression algorithms, simple Huffman coding and common data-stream coding are two promising candidates for automotive multiplexing [6]. Unfortunately, these algorithms are only useful for text data class in a car-body multiplexing system [6]. To overcome this limitation, we have presented a generalized data-reduction algorithm in [9]. This data-reduction technique can be applied to all kinds of data classes generated for various automotive control functions, including engine management, transmission control, steering control, and body control. The algorithm is designed to work with the SAE J1939 protocol [10]. However, this algorithm is general and can be modified easily for all automotive multiplexing protocols, including CAN, VAN, A-BUS, and J1850 [11]. The objective of this paper is to present the performance analysis of the data-reduction algorithm introduced in [9].

II. SAE J1939 Protocol

Controller area network (CAN), introduced by Bosch, is a network protocol designed specially for in-vehicle networking [12]. CAN is the only protocol so far that supports all classes of automotive communication defined by the Society of Automotive Engineers (SAE) [2]. The previous version of CAN, version 1.2, has been successfully implemented in passenger cars, trains, and factory automation. The CAN protocol has been extended with a second message format called CAN V 2.0 [12]. The SAE has accepted this extended version of CAN protocol as the basis for in-vehicle networks in trucks and buses. The protocol defined by the SAE is called SAE J1939 [10]. It defines all relevant communication layers to support automotive multiplexing. Within J1939, the message is called the protocol data unit (PDU). There are two existing formats for the protocol data unit. PDU1 includes the source and destination address, while format PDU2 is used to send the data frames that are not destination specific. In both PDU1 and PDU2, SAE has reserved a bit for future use, which is called the reserved or R bit. We suggest that the R bit can be used to indicate data compression. With this adoption, no additional bit will be required to reflect the data-compression process. In the proposed data-reduction algorithm, the R bit will be redefined as the data-compression bit (DCB). In the case of compression, DCB is set to “1”; when no compression is performed, it is set to “0.” A PDU1 format with DCB is shown in Table XIII.

III. PROPOSED DATA REDUCTION ALGORITHM

The SAE J1939 protocol defines various messages that may be generated in an automotive multiplexing system [14]. A careful study of the J1939 messages reveals that some of the bytes in the data field may remain constant for some time. Moreover, many messages change slowly. This observation might lead to a good data-reduction technique.

A. Assumptions

In an automotive multiplexing system, most of the intelligent control modules (ICMs) communicate with each other through message passing. A signal analysis of all messages generated in automotive multiplexing has shown that each module transmits or receives a fixed set of messages [15]. The following assumptions have been made in the proposed DR algorithm.

1) An ath ICM transmits messages and receives k messages. These messages are represented by \( S_T = \{m_1, m_2, m_3, \ldots, m_n\} \) and \( S_R = \{m_1, m_2, m_3, \ldots, m_k\} \), respectively.
2) Two buffers called T_BUF and R_BUF are needed at each ICM. These buffers are required to store the transmitted and received messages, respectively.
3) The sizes of buffers T_BUF and R_BUF are equal to the sizes of ST and SR, respectively. T_BUF is used in the data-compression process at the message transmitting side, and R_BUF is used in the data-decompression process at the message receiving side.
4) An ICM can include both T_BUF and R_BUF simultaneously.
5) For all messages generated by all intelligent control modules, bytes in the message data field are repeated with probabilities shown in Table I.

Assumption 5) is based upon the study of SAE recommendation practice J1939 [13]. This study shows that some data bytes represent the status of some physical parameters. If the status of these parameters remains constant for some considerable amount of time, then the corresponding data bytes will remain constant. For example, gear position, wheel speed, vehicle speed, engine temperature, accelerator pedal position, etc., may remain constant for a certain amount of time. Therefore, data bytes associated with these parameters may be assumed constant with high probability.

B. Data Compression

To implement the data-compression algorithm, the transmitting unit of an ICM keeps a copy of the data field of the transmitted messages in the T_BUF. Each entry in the T_BUF consists of two fields. The ID field holds the message identifier of
the transmitted message, and the data field keeps a copy of the data field corresponding to the transmitted message.

In an automotive multiplexing system, intelligent control modules generate messages at a specific or random rate. Assume that an ICM transmits a J1939 message \( m_i \) after every \( r \) time units. Whenever message \( m_i \) is transmitted, the ICM keeps the copy of the data field of \( m_i \) in T_BUF. When the next \( m_i \) is transmitted after \( r \) time units, the transmitting unit compares its data field with that of \( m_i \) saved in the T_BUF. If two or more data bytes of the new message are found to be the same as in the T_BUF, then the transmitter will understand that some data bytes have been repeated. To reflect this repetition, the transmitter sets the DCB in the PDU to “1” and prepares a compression code (CC). The CC will indicate which bytes are repeated in the recently transmitted message. Each bit in the compression code corresponds to a data byte of the message. A bit with a value of “1” corresponds to a repeated byte in the message, and a bit with a value of “0” corresponds to the byte that is not repeated. The nonrepeated bytes are followed by CC in the data field of the actual data frame transmitted over the multiplexing bus. The data-compression process described above is presented in the algorithm shown in Fig. 1.

**C. Data Decompression**

The data-decompression process is the reconstruction of the original message sent by the transmitting unit. At the receiving side of the system, each ICM keeps a copy of the recently received message in a buffer called R_BUF.

When the ICM receives the same J1939 message at a later message reception time, it will check the DCB of the received message frame. If the DCB is found to be “1,” then the receiver will understand that the data bytes in the message frame have been repeated. In this case, the receiving unit of the ICM will treat the first byte in the data field of the received J1939 message as the compression code. The indexes of the repeated data bytes are determined from those bits in compression code whose values are “1.” The nonrepeated data bytes, if there are any, will follow the compression code. The indexes of these nonrepeated data bytes are determined by those bits in CC whose values are “0.” An exact copy of the message actually transmitted will be prepared by reading repeated bytes from the R_BUF of the receiving ICM and nonrepeated data bytes from the received message. The decompression process is summarized in the algorithm shown in Fig. 2.

In order to illustrate the data-decompression algorithm, we assume that bytes 1, 2, and 3 in a transmitted message have been repeated. To indicate this repetition, bits 1, 2, and 3 of the CC will be “1.” To indicate the positions of nonrepeated data bytes, bits 4 to 8 of the CC will be set to “0.” Nonrepeated data bytes will follow the CC in the data field of the received message. The compression code to indicate data repetition is shown in Fig. 3.

The receiving unit of the ICM will retrieve bytes 1, 2, and 3 from its R_BUF. In this example, six data bytes are transmitted instead of eight data bytes (CC plus five nonrepeated data bytes).
The number of transmitted data bytes decreases as the number of repeated data bytes increases. Consequently, if all data bytes are repeated, then only one byte of compression code is transmitted instead of eight data bytes. If no data bytes are repeated, the compression code is not needed and data transmission remains normal. That is, when no data bytes are repeated, data expansion does not occur either.

IV. RESULTS AND DISCUSSION

This section presents a performance analysis of the proposed data-reduction algorithm. The performance of the algorithm has been measured in terms of the following parameters: compression ratio, average bus utilization, average bus queue length in terms of number of messages, average bus queue length in terms of number of bits, average message delay, maximum latency, and bus queuing probability. A simulation program has been written to simulate the proposed data-reduction algorithm. For the purpose of comparison, the same simulation program is executed with and without data-compression algorithms. First, the simulation program was run to collect results for 30 min of real-time operation. Then it was run again to collect results for 80 min of real-time operation. In the following sections, the results for 80 min of real-time operation are shown under the column entitled \textit{Long-Term Average}. No significant differences have been observed between the results of 30 and 80 min of real-time operation. In most of the cases, the results are very close—within 5% of each other. Thus, we believe that the results shown in this paper indicate a very realistic performance of the proposed data-compression algorithm.

The example data consist of 44 typical messages generated by a set of seven processors attached to the system bus. Fig. 4 shows the simplified automotive multiplexing considered in the simulation. Each processor consists of buffers called T_BUF and R_BUF to store the copies of the transmitted and received messages, respectively. Table II shows the simulation input data. The column labeled “MID” represents the message identification numbers. Similarly, the column labeled “PID” gives the processor ID number, which generates the corresponding message. Table III shows the message distribution among seven processors. The first column shows the PID and the second column gives the MID generated by the processor.

A. Compression Ratio

In order to evaluate the compression ratio for the proposed algorithm, we consider all repetition possibilities in a typical message. In this typical message, if all bytes are repeated, then only one compression code byte will be sent. On the other hand, if seven out of eight bytes of a message are not repeated, then one compression code byte is sent, followed by seven nonrepeated bytes. Table IV lists all the repetition possibilities in a typical message.

The last column in Table IV shows the compression ratio corresponding to the repetition level. We find that in the best case, the compression ratio is 87%, and in the worst case, when one or no bytes are repeated, the compression ratio is 0%. The proposed algorithm does not put any overhead on the system. In the case of no repetition, in the generated data, the transmitter simply resets the DCB bit to “0” to indicate that no compression has been performed on the data. In this situation, the receiver will consider the received message as a normal J1939 message and, therefore, will not try to interpret the very first byte in the data field as the compression code.

B. Bus Utilization

Bus utilization is the ratio of utilized bus time to the total time. In other words, it is the percentage of time the bus remains in use. This is an important parameter for the evaluation.
Table V shows the bus utilization results obtained from the simulation of the proposed data-reduction scheme. Fig. 5 shows that as traffic over the system bus increases, bus utilization increases.

Fig. 5 shows that when the data-compression scheme is not applied, bus utilization reaches 100% with only 20 messages in the system. But, when the DR algorithm is applied, bus utilization reaches 100% with 35 messages. This means that with the application of proposed data-reduction scheme, the system can handle more data traffic. In other words, the DR algorithm allows us to introduce new messages corresponding to new features with the same characteristics of the automotive multiplexing bus.

C. Average Message Delay (AMDL)

Average message delay is another important parameter for the performance evaluation of a multiplexing system. It is defined as the average amount of time spent by a message in the bus queue. The numerical results generated from this simulation show that AMDL was significantly lower when the proposed data-reduction algorithm was applied. Table VI compares AMDL as a function of source messages with and without the data-reduction algorithm. Fig. 6 shows a graphical relation of this comparison. This comparison shows that the application of the proposed DR algorithm significantly reduces the message delay time: from 32.6% to 43.3% with high and low data traffic over the bus, respectively.

D. Average Bus Queue Length

Bus queue length is one of the important parameters used for the performance evaluation of an automotive multiplexing system. This parameter is related to the number of messages waiting in the bus queue.

The processors in the system generate messages at regular or random intervals of time. Assume that a message \( m_k \) is generated at time 0. If the bus is free, then this message is sent over to the bus. After the successful message transmission, message \( m_k \) is scheduled for its next transmission. At the time when the bus is busy by message \( m_k \), another processor may generate a message \( m_j \). This message will find the bus busy. When the bus is busy, message \( m_j \) is placed in a bus queue. By applying the data-reduction algorithm, we observe that the message queue is less congested. In other words, when the data-compression algorithm is applied to a multiplexing system with the same set of transmission parameters, we find relatively better performance of average bus queue length.

Table VII compares the situation in which messages are placed in the bus queue. The results compare the effect on data congested in the bus queue with and without the data-reduction scheme. We find that without the data-reduction scheme, on average there are approximately six messages waiting in the bus queue. With the data-reduction scheme, however, approximately four messages are in the bus queue with the same bus characteristics. This means that on an average, there are 33% fewer messages waiting in the queue when the DR algorithm is applied versus when no data reduction is applied. The second and third columns in Table VII show the bus queue size in terms of number of data bits. Fig. 7 shows the graphical relationship between average bus queue length in bits as a function of the number of source messages.

This result shows that there is less data accumulation in the bus queue when data reduction is applied. The fourth and fifth columns of Table VII compare bus queue length in terms of
number of messages with and without the application of the proposed data-reduction algorithm. Fig. 8 shows the graphical relationship between average message queue length and number of source messages.
The numerical results obtained from the simulation confirm that the application of the proposed data-reduction algorithm reduced data congestion on the bus queue. From Table VII, it is observed that when an automotive multiplexing supports five messages, we get a 66% reduction in bus queue length in terms of data bits by applying the data-reduction algorithm. On the other hand, when there are 44 messages, we get 51% reduction in bus queue length in terms of data bits by applying the data-reduction algorithm. Similarly, message queue length is reduced by 31–44% in case of high and low traffic, respectively. The significance of a reduced queue length is that a smaller buffer is necessary to keep the queued messages.

### E. Maximum Latency Time

Latency time is defined as the time interval from the instant a message is generated to the instant when the message’s last bit is received by the receiver. Certain messages in the system must reach their destination before a predefined deadline. It is therefore important to determine maximum latency time when there is a certain amount of data traffic on the bus. The simulation results show that the proposed data-reduction algorithm decreases the message transmission time. Table VIII presents a comparison of the results of applying the data-reduction algorithm. Fig. 9 shows the graphical relation between the number of messages and maximum latency time. It is obvious that maximum latency time increases as the number of messages increases in the system. However, we get better performance by applying the proposed data-reduction algorithm. In other words, with the data-reduction algorithm, the messages reach their destination in a relatively shorter period of time.

### F. Bus Queued Up Probability

At the instant a message is generated, the bus may or may not be busy. When the bus is busy, the generated message is placed in the bus queue. We define “bus queued up probability” (BQP) as the number of transmission failures divided by the number of transmissions attempted over a simulation period. The simulation results are shown in Table IX. This result shows that when the data-reduction algorithm is applied, fewer messages are queued at the bus. In other words, with the proposed data-reduction algorithm, messages get more chances of obtaining a free bus.

### G. Message Throughput

With the proposed data-reduction scheme, the messages occupy the bus for a relatively short period of time. This allows more messages to pass over the system bus. To determine message throughput, another simulation run was executed in which some messages of random interval were considered. Table X lists input simulation data with random messages. In this example’s data, messages 5, 14, 24, and 35 are generated randomly with a mean time interval of 0.005 s. Numerical results obtained from the simulation show that with the application of the proposed data-reduction algorithm, more messages access the bus without queuing onto the system bus. The results are shown in Table XI.

We define a parameter called “throughput gain ratio” as the ratio of the total number of messages generated with and without compression. This ratio tells how many more messages are passed through the bus when the data-reduction algorithm is applied. The throughput gain ratio is presented in Table XII.
Studies on the automotive multiplex system have revealed that there are a limited number of messages inside the system. Each intelligent unit connected to the multiplexing system transmits or receives a fixed set of messages. It has been observed that some data bytes in the data fields do not change often. Considering this observation, we introduce a data-reduction technique for the automotive multiplexing system. This algorithm is implemented using the J1939 protocol. However, the algorithm is general and can be applied to any of the automotive multiplexing protocols. The proposed algorithm generates compression code only if some data bytes in the messages are repeated. The compression code is not sent when data bytes are not repeated. Therefore, the algorithm does not add any overhead in case of nonrepeated data bytes. A discrete event simulation has been performed to verify the functionality of the proposed algorithm. The numerical results show that the proposed algorithm gives improved results in terms of compression ratio, bus utilization, average message delay, maximum latency time, average bus queue length, and throughput gain ratio.

### Table XII

<table>
<thead>
<tr>
<th>No. of Messages</th>
<th>Throughput Gain Ratio</th>
<th>Throughput Gain Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Long-term Average</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1.012</td>
<td>1.010</td>
</tr>
<tr>
<td>10</td>
<td>1.091</td>
<td>1.087</td>
</tr>
<tr>
<td>15</td>
<td>1.110</td>
<td>1.095</td>
</tr>
<tr>
<td>20</td>
<td>1.051</td>
<td>1.047</td>
</tr>
<tr>
<td>25</td>
<td>1.045</td>
<td>1.042</td>
</tr>
<tr>
<td>30</td>
<td>1.039</td>
<td>1.039</td>
</tr>
<tr>
<td>35</td>
<td>1.025</td>
<td>1.024</td>
</tr>
<tr>
<td>40</td>
<td>1.019</td>
<td>1.023</td>
</tr>
</tbody>
</table>

### Table XIII

<table>
<thead>
<tr>
<th>Priority</th>
<th>DCB</th>
<th>DP</th>
<th>PDU Format</th>
<th>Destination Address</th>
<th>Source Address</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>0-64</td>
</tr>
</tbody>
</table>

**DP**: Data Page  
**DCB**: Data compression bit

**V. CONCLUSION**

Electronics has been introduced in automobiles to enhance the functionality of the system. In order to include more features inside the automobile, more electronic control units are required. An increase in the number of electronic control units will increase data traffic over the multiplexing bus.

Studies on the automotive multiplexing system have revealed that there are a limited number of messages inside the system. Each intelligent unit connected to the multiplexing system transmits or receives a fixed set of messages. It has been observed that some data bytes in the data fields do not change often. Considering this observation, we introduce a data-reduction technique for the automotive multiplexing system. This algorithm is implemented using the J1939 protocol. However, the algorithm is general and can be applied to any of the automotive multiplexing protocols. The proposed algorithm generates compression code only if some data bytes in the messages are repeated. The compression code is not sent when data bytes are not repeated. Therefore, the algorithm does not add any overhead in case of nonrepeated data bytes. A discrete event simulation has been performed to verify the functionality of the proposed algorithm. The numerical results show that the proposed algorithm gives improved results in terms of compression ratio, bus utilization, average message delay, maximum latency time, average bus queue length, and throughput gain ratio.

### REFERENCES

[10] “Recommended practice for serial control and communications vehicle network (Class C),” SAE draft J1939, 1939.

Syed Masud Mahmud received the B.S. degree in electrical engineering from Bangladesh University of Engineering and Technology, Dhaka, in 1978 and the Ph.D. degree from the University of Washington, Seattle, in 1984.

Currently, he is an Associate Professor in the Department of Electrical and Computer Engineering, Wayne State University, Detroit, MI. His areas of interest are computer architecture, parallel processing, performance analysis, simulation, and multiprocessor-based system design. He has published more than 50 technical papers in refereed journals and conference proceedings.

Dr. Mahmud was awarded the University Gold Medal for securing the highest marks among all the recipients of the bachelor’s degree in the Bangladesh University of Engineering and Technology during 1978. He received a Physio Control fellowship from the University of Washington during the 1982–1983 and 1983–1984. He was nominated for the President’s Teaching Excellence Award at Wayne State University in 1991. He received the Outstanding Teaching Award of the College of Engineering at Wayne State University in 1994.

Syed Misbahuddin (S’95–M’97) received the B.E. degree in electrical engineering from Dawood College of Engineering and Technology, Karachi, Pakistan, in 1983, the M.S. degree in electrical and computer engineering from King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia, in 1988, and the Dr.Eng. degree in electrical engineering from the University of Detroit, Mercy, MI, in 1998.

From 1988 to 1992, he was an Assistant Professor in the Electrical and Computer Engineering Department of NED University of Engineering and Technology, Karachi, where he was engaged in a variety of computer engineering research. Currently, he is a Faculty Member in the Computer Systems Department of King Fahd University of Petroleum and Minerals, Hail, Saudi Arabia. His research interests are in the areas of distributed computing, data reduction algorithms, and microprocessor engineering.

Dr. Misbahuddin is a member of the Pakistan Engineering Council.

Nizar Al-Holou (SM’94) received the Bachelor’s degree from Damascus University, Syria, the M.Sc. degree from The Ohio State University, Columbus, and the Ph.D. degree from the University of Dayton, Dayton, OH, all in electrical engineering.

He is a Professor and Chairman of electrical and computer engineering at the University of Detroit Mercy, Detroit, MI, where he teaches courses on digital logic, computer networks, microprocessors, computer architecture, and electrical circuits. His area of expertise is in digital systems, microprocessors, fuzzy logic, GPS, real-time systems, computer-based instruction, and computer architecture. He has published in the areas of parallel processing, real-time systems, fuzzy logic, GPS, and computer-based instruction. He has received funding from the National Science Foundation and industry. He has published more than 50 technical papers in refereed journals and conference proceedings.

Dr. Al-Holou has received numerous awards, such as the IEEE/SEM Outstanding Chapter Involvement Award, IEEE-Eit 2000 Award, and FIE 98 Best Paper Award. He is in Who’s Who in the Midwest 1994 and Who’s Who Among America’s Teachers. 5th edition, 1998. He has served on the board for IEEE/SEM, ASEE/NCS, the Rocky Mountain Bioengineering Symposium, ASACA, and YUFORIC. He has served as Chairman and Vice Chair of the Computer Chapter of the Southeastern Michigan IEEE Section since July 1994.

He served as the ASEE-NCS Conference Chair in 1998.