A DIGITAL SIMULATOR DESIGN FOR REAL-TIME AND OPEN-LOOP APPLICATIONS

N.A. Izquierdo Jr.
Commonwealth Edison Company

M.Kezunovic, Z.Galijasevic, F.Ji[1],
A.Gopalakrishnan, J.Domaszewicz
Texas A&M University

U.S.A

***Abstract* – This paper describes new design of a digital simulator for relay testing. The main simulator feature is its capability to operate in both real-time and open-loop modes. In the real-time mode, the modeling software used is the Real Time System (RTS) developed by Texas A&M University. In the open-loop mode, a standard Electromagnetic Transient Program (EMTP) is used. A common Graphical User Interface (GUI) is used for both operating modes. The main design features of the simulator are summarized in this paper.**

***Keywords* – Digital Simulation, Protective Relaying, Relay Testing, EMTP, Real-Time Application**

I. INTRODUCTION

Recent development of digital simulator technology has provided an opportunity for use of digital simulators in testing protective relays both in real-time and open-loop modes. Texas A&M University has developed an advanced real-time simulator under the sponsorship of the Department of Energy–Western Area Power Administration (WAPA). The simulator was delivered to WAPA in the Spring of 1994, and its real-time operation has been demonstrated [1-3]. In another project under EPRI sponsorship, Texas A&M University developed an open-loop simulator aimed at back-to-back testing of two terminal protective relay applications. This design has extensive software capabilities for user support in performing relay testing [4]. This simulator has been developed using hardware and software that is compatible with the ones used in the real-time design. This provides for great flexibility in upgrading from the open-loop design to the real-time one [5].[1]

Recently, Texas A&M University was awarded a contract from Commonwealth Edison Company that included the development of a flexible simulator to operate both in real-time and open-loop modes. Due to the compatibility of the previous designs, it was possible to provide both modes of operation using the same hardware and system software environment. This paper discusses implementation characteristics and major benefits of such a design achieved by providing both operating modes.

II. DESIGN REQUIREMENTS

The simulator was implemented under two sets of requirements. One set was related to the general features as follows:

- Simulator hardware should be, as much as possible, commercially available "off the shelf"
- Simulator system software should be standard solution commercially available "off the shelf"
- Simulator application software should be written in high level programming language and portable
- Simulator should have an elaborate Graphical User Interface implemented using standard X-Windows tools

The other set of requirements was related to the specific application constraints as follows:

- Simulator should operate in both real-time and open-loop modes
- The real-time mode should allow for interactive testing of protective relays to change network topology for such conditions as autoreclosing sequences
- The open-loop mode should allow for interactive testing of protective relays by submitting test files generated through simulations using Electromagnetic Transient Program (EMTP)
- Graphical User Interface should allow for a common access to both the real-time and open-loop modes [6]

To meet these requirements, a new design had to be implemented. Various features of both the real-time [3] and open-loop [7] simulators, previously developed independently by Texas A&M University, were combined in one simulator design.

[1]The work has been done while the author was with Texas A&M

III. IMPLEMENTATION

This section describes various implementation aspects of the new simulator.

A. System Architecture

The major hardware and software building blocks of the simulator architecture are shown in Fig. 1.

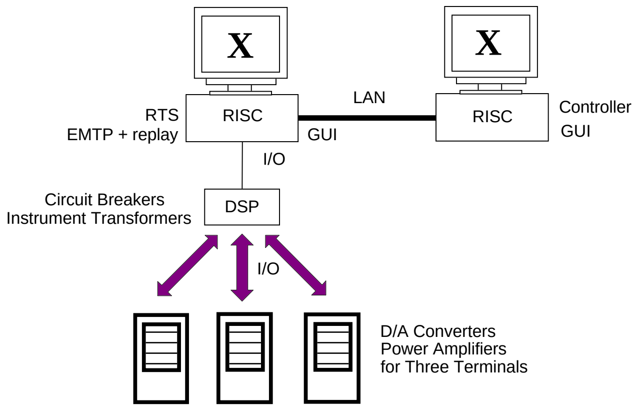

Fig. 1. The architecture of the simulator

In the closed-loop operating mode, transient simulation is carried out on-line so that a real-time change of the network topology controlled by the relay under test is possible. On the other hand, in the open-loop mode transient simulation is done off-line so that there is no possibility to change network topology under the control of the relay. To provide these two modes, unique hardware and software solutions are used.

The hardware basis of the simulator consists of two RISC computers, DSP subsystem, two sets of I/O boards and amplifiers subsystem. One computer serves as a user front-end machine for both open-loop and closed-loop simulators. Graphical user interface software *(GUI)* runs primarily on that computer. The second computer, the faster one, is dedicated to run real-time transient simulation as well as real-time replay of EMTP output file. One set of I/O boards connects that computer to the DSP subsystem where the real-time simulation of instrument transformers and circuit breakers takes place. Another set of I/O boards serves as a connection between the DSP subsystem and amplifiers' section of the simulator.

The simulator software consists of the graphical user interface software, conversion software, software for file management and resource control, as well as real-time simulation and playback software. Both operating modes of the simulator are provided to the user by means of the common graphical user interface (*GUI*). Several programs that perform conversion between the different file formats accompany the user interface software. In the closed-loop mode, network simulation is performed in real-time by the *Real-Time System* (*RTS*) software, while DSP boards-based software simulates instrument transformers and circuit-breakers. In the open-loop mode, entire network simulation is done in preparatory phase by EMTP, while only the replay of the output file takes place in real-time. File transfer from the graphical user interface to network simulation software and back is managed by the program called *Sequencer*. Finally, *Controller* software is used to ensure that I/O hardware of the simulator is used on scheduled basis.

B. Hardware Implementation

The simulator hardware is shown in Fig. 2. It can support up to 3 terminals for relay testing. The system hardware consists of two RISC machines, each of which can be equipped with the Graphical User Interface (GUI). However, all connection to the external I/O hardware is made through the RISC System/590 only. This is to ensure maximum computational speed for the real-time simulation.

The DSP subsystem is composed of a single 'central' DSP and a number of 'peripheral' DSPs (TI's TMS320C40 chips). The central processor is connected to the host and to the peripheral DSPs.

The I/O subsystem is divided into a communication interface to receive/send serial data from/to the DSP subsystem, a D/A subsystem for reconstruction of the analog signals, a digital I/O subsystem to monitor contact status changes of the device under test, and a hardware mechanism for clock synchronization between the Master and Slave I/O terminals. Clock Synchronization between the terminals is achieved by using Phase Locked Loop ICs, which ties the Slave Terminal clocks to the Master Terminal clock. The I/O subsystem and the amplifier subsystem are packaged into a custom designed high-precision cabinet. The interfaces are designed so that both open-loop and real-time testings are possible using the same hardware.

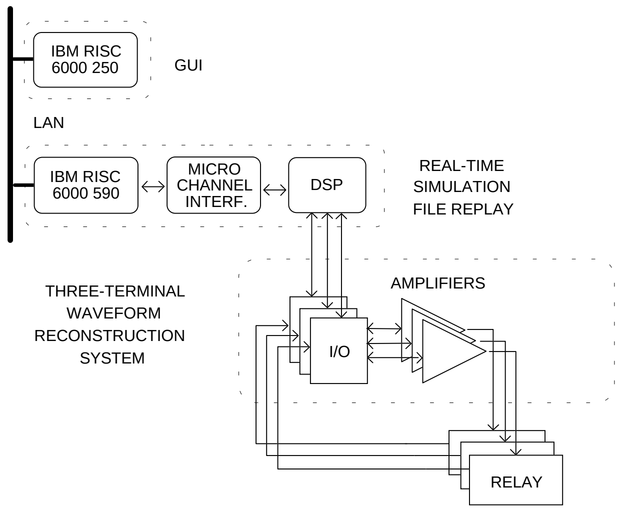

Fig. 2. Simulator Hardware

C. Software implementation

The software architecture of the simulator and interactions between its major components are shown in Fig. 3.

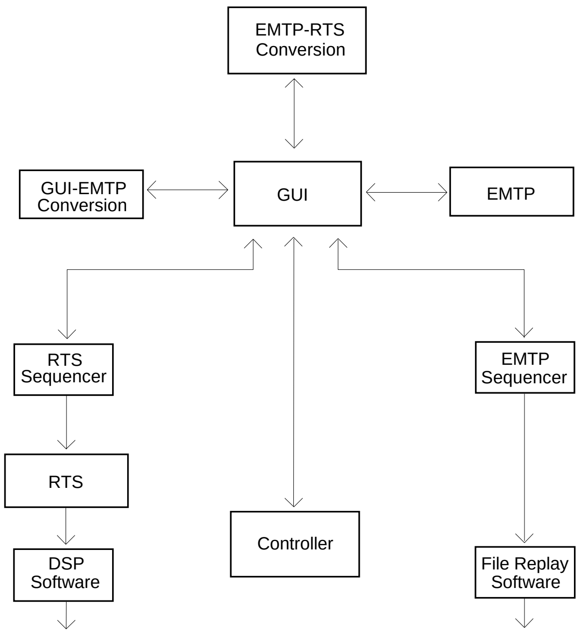

Fig. 3. The software architecture of the simulator

D. Graphical User Interface

The user builds the network in one-line diagram form and initiates the simulation in both operating modes of the simulator by using a common graphical user interface. The network is built in a paint-brush-program way; the user selects with the cursor an icon in the toolbox representing a network component, inserts it to the required location in the drawing area, and connects it to the rest of the network using editing tools such as move, cut, link, delete, undo, etc. [6]. The parameters of the components are entered through a menu-driven interface. Ease of use is further enhanced by the existence of the libraries of predefined models for some of the complex network elements.

As far as the *GUI* is concerned, there is no difference between the two operating modes until the very start of the simulation. In both modes the same graphical interface is used: the same components and tools are utilized to build the network, and component parameters are defined through the same menus. The two operating modes are forked by selecting different option provided by the *GUI* menu before the simulation is about to start. This transparency, along with the unique output file format, is ensured by using the appropriate file conversion prior to the simulation.

In order to support concurrent use of the simulator by multiple users (in closed-loop and open-loop mode), *GUI* software is written as a client program. It means it relies on certain services provided by the server program (in this case called *Controller*). The service can be either real-time network simulation or real-time waveform file replay. Multiple real-time operations are not allowed at the same time because they attempt to use the same hardware, each on the exclusive basis. Therefore, *GUI* negotiates with the *Controller* when to start such an operation.

E. Conversion software

After the user creates a network using one-line network editor feature of the *GUI*, it can be saved in the output file in the specific format (referred as ".gui"). This file keeps all information that enables one to redraw the network at a later time. However, in order to run the simulation, this file must be first converted into another file whose format depends on the simulation program used in the specific operating mode.

For both possible types of the simulations, ".gui" file needs to be converted into the *EMTP* input file format. However, since *RTS* based simulation uses some complex component models that do not exist in *EMTP*, the output file can not be the same as for the *EMTP* based simulation. Therefore, specific 'hybrid' *EMTP* format is adopted for the *RTS* based simulation in order to provide additional information. Likewise, since ".gui" file itself does not bear all the information needed for *EMTP* based simulation, a special mechanism for building input file from predefined "include" file modules has been developed.

Since *RTS* requires its input in the specific format, an additional file conversion is needed in closed-loop operating mode. First, 'hybrid' *EMTP* input file has to be processed so that all the network components are given in the specific order. Then, in order to shift as much of the computational burden as possible into the preparatory phase, the network admittance matrix and the forcing function vectors are computed.

F. Real-Time System (RTS)

The network simulation in the closed-loop operating mode has to be performed in real-time. Ordinary programs for power system transients simulation are not suitable for this purpose. Therefore, a program specially developed for the real-time network transient simulation is used in the TAMU's simulator.

The *RTS* is a new program specially tailored for the real-time applications [1,2]. The main objective in its development was to meet very stringent time and memory requirements. It is, therefore, written in C with the extensive use of the speed optimizing techniques. To meet the other requirements, such as accuracy and stability, *RTS* uses *EMTP* modeling and solution techniques.

All the main elements existing in *EMTP* are also included in *RTS*: simple branches, coupled branches, overhead transmission lines, voltage sources, MOVs, voltage arresters and time switches. Additionally, there is a number of complex network elements such as relay controlled circuit breakers, series capacitors with MOV protection, voltage and current instrument transformers and faults. Common to all of them is that they are handled in alternative ways so that the real-time nature of the simulation is not jeopardized. For example, series capacitors with MOV protection are modeled and solved efficiently as a single component [2]. A special solution technique is used to represent switching operation. Circuit breaker and instrument transformer models are implemented using DSP boards.

G. File replay

The file replay software enables replaying of the waveforms generated by EMTP and collecting of the relay response data. The file replay software described here is a major upgrade of the version incorporated in the previous open-loop simulator. In both cases the software is composed of two modules: the host (RISC) module and the DSP module. In the previous simulator design, waveform data is downloaded by the host to the local DSP memory prior to actual replaying. Even though such an approach greatly simplifies the overall design (the host is not directly involved in replaying), it has an obvious drawback: the length of a waveform is limited by the capacity of the DSP memory. The new version enables real-time transfer of waveform data from a disk file system to the I/O cabinets. Thus, any size of the waveform data can be accommodated.

Organization of the file replay software is shown in Fig. 4. The first step (not done in real-time) is the conversion of the *EMTP* output file to the one in which waveform samples are represented using the D/A converter format. The second step is the actual replaying. The host module is implemented as a single UNIX process. The main task of the module is to read waveform data from a disk file to a buffer located in the main memory of the host. A chunk of data of a predefined size is read at a time. After a disk read operation, the host module checks the status of a buffer located in the local memory of the DSP subsystem. Chunks of data are transferred between the host buffer and the DSP buffer as long as they are available in the former one and there is space in the latter one; after that another disk read is initiated. Host-to-DSP transfers take only about 10 % of the overall time budget. Hence, on the host side, virtually all the time is devoted to the most time-consuming operation: reading from the disk.

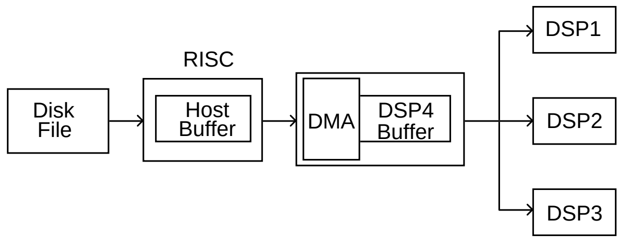

Fig. 4. Organization Of Replaying Software

The DSP buffer is located in the local memory of the central DSP. The central DSP manages the DSP buffer and distributes waveform data to the peripheral DSPs. The data is received from the host using an on-chip DMA controller. The peripheral DSPs receive data from the central one and present them to the waveform reconstruction system contained in the I/O cabinets. The peripheral DSPs also collect the relay response data from the cabinets and store the data in their local memory. After the replaying has been completed, the relay response data is transferred to the central DSP and, in turn, to the RISC host, which stores them in the capture files.

H. Controller

To make it possible for multiple users to concurrently use the simulator, a program called *Controller* is used. It acts as a server, providing two services (real-time simulation and real-time file replay) to the clients (*GUIs*). Its task is to resolve possible collisions of the concurrent requests from different clients (*GUIs)*, since certain simulator's hardware must be used on exclusive basis.

The logic on which *Controller* operates is rather simple as shown in Fig. 5. Before any real-time operation, *GUI* asks *Controller* for permission. Since *Controller* keeps track of the status of all *GUIs* and hardware resources, it checks if the requested resource is available and grants or denies permission. The implementation of the *Controller* and client part of the *GUI* is done using standard UNIX mechanisms for interprocess communication. All the requests coming from *GUIs* to the *Controller* are put into the input queue and then processed on the first come first serve basis. Backward communication is based on the same mechanism.

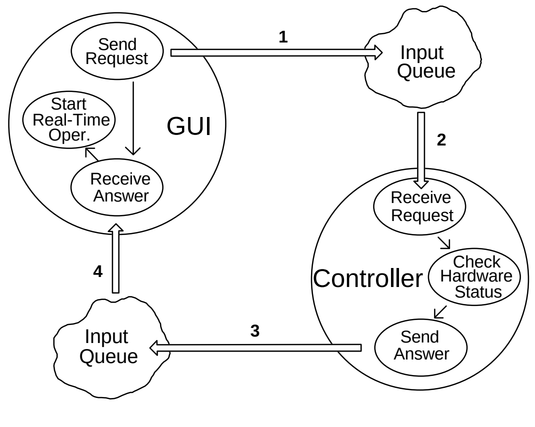

Fig. 5. GUI–Controller Communication

IV. EXAMPLES OF SIMULATOR OPERATING MODES

This section explains in more detail each of the operating modes of the simulator.

A. Closed Loop Operating Mode

In both operating modes the user first builds network by using *Graphical User Interface* (*GUI*). Network description is contained in the file in ".gui" format. In closed loop mode, described in Fig. 6, *GUI* then invokes *GUI-EMTP* software. It converts ".gui" file into ".rts" file whose format closely resembles format of an EMTP input file. After that, *GUI* starts *EMTP-RTS* software that produces two files: ".inp" and ".ss". First file contains network data in the specific format requested by *RTS*, and second file keeps steady state solution data. To create ".ss" file, *EMTP-RTS* invokes *EMTP*.

Prior to the start of the real-time simulation, *GUI* sends a request to the *Controller*. *Controller* checks if the hardware that is to be used is available, and sends an answer. If the permission is granted, *GUI* transfers necessary files and invokes *Sequencer*. *Sequencer* starts *RTS* which outputs its results through the DSP subsystem as soon as they are calculated. Relay responses are used to change the network topology interactively by the operation of the respective relay terminal(s). At the end of the simulation, *Sequencer* copies back the waveform (".wav")and message files (".msg").

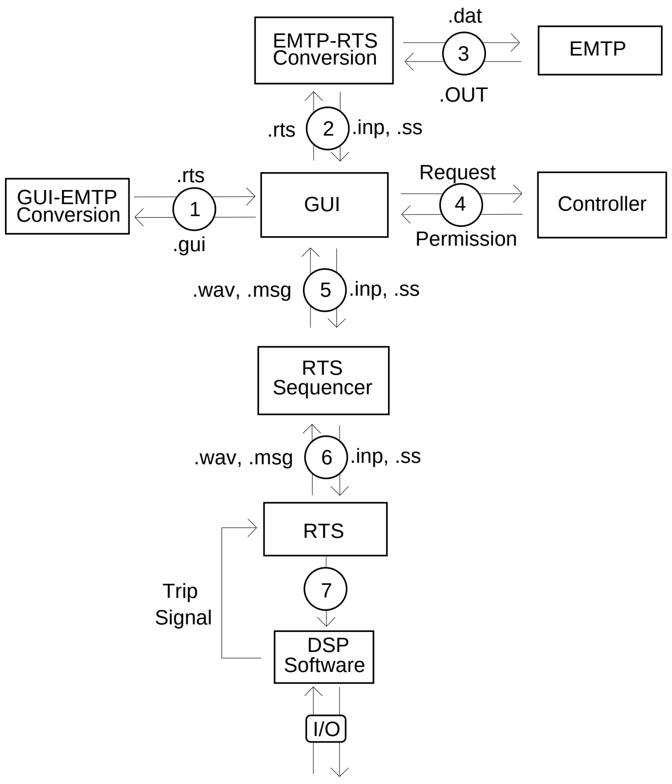

Fig. 6. Closed-Loop Operating Mode

B. Open Loop Operating Mode

Fig. 7 shows the involvement of particular software modules in the open loop operating mode of the simulator.

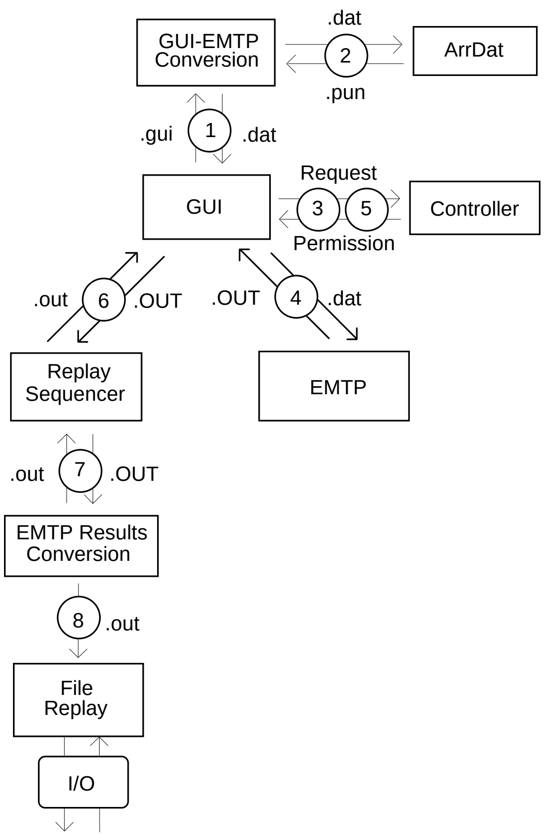

Fig. 7. Open-Loop Operating Mode

Again, the first step is to build the network by using *Graphical User Interface* (*GUI*). In the next step *GUI* invokes *GUI-EMTP* conversion software. It converts ".gui" file into ".dat" file whose format is identical to the format of an EMTP input file. In addition to the conversion used in the closed loop mode, this involves processing of the "include" files and component reordering. Also, during this conversion, EMTP auxiliary program *ArrDat* might be called to process surge arrester and metal-oxide varistor data.

Since in this operating mode only file replay happens in real-time, *GUI* starts *EMTP* separately. However, it still communicates with the *Controller* to find out where to run *EMTP* most efficiently. After the *EMTP* simulation is done, *GUI* requests permission from the *Controller* to start real-time replay. If the replaying hardware is available, the permission is granted. *GUI* then transfers EMTP output file and invokes Replay Sequencer. *Sequencer* first invokes software that processes raw EMTP results, and then replaying software. At the end of the replay, *Sequence*r copies back the waveform and relay response file.

V. CONCLUSIONS

Simulator description given in this paper leads to the following conclusions:

- It is possible to to implement digital simulator design that is capable of both real-time and open-loop operation
- Simulator hardware and system software can be selected as "off the shelf" products
- Graphical User Interface (GUI) can be implemented to serve as a common interface for both operating modes

VI. ACKNOWLEDGMENTS

The simulator development reported in this paper is funded by the Commonwealth Edison Company of Chicago under contract # 811856.

VII. REFERENCES

[1] M. Kezunovic, et. al. "Transient Computation for Relay Testing in Real-Time", IEEE Transaction on Power Delivery, vol. 9, no. 3, July 1994, pp. 1298-1307.

[2] M. Kezunovic, et. al. "Computing Responses of Series Compensation Capacitors with MOV Protection in Real-Time", IEEE PES Summer Meeting, paper no. 94 SM 400-2 PWRD, San Francisco, July 1994.

[3] M. Kezunovic, et. al. "Design, Implementation and Validation of a Real-Time Digital Simulator for Protection Relay Testing", IEEE PES Winter Meeting, Paper no. 95 WM 034-9 PWRD, New York, February 1995.

[4] M. Kezunovic, et. al. "Advanced Signal Processing and File Management Software for Relay Testing Using Digital Simulators", 11th. PSCC, Avignon, France, September 1993.

[5] M. Kezunovic, et. al. "New Digital Simulator Design for Protective Relay Testing", Texas A&M Relay Conference, College Station, Texas, March 1995.

[6] M. Kezunovic, et. al. "Extensible Graphical User Interface for Digital Simulators", First International Conference on Digital Simulators, College Station, Texas, April 1995.

[7] M. Kezunovic, et. al. "Design Characteristics of and Advanced Two-Terminal Digital Simulator for Relay Testing", First International Conference on Digital Simulators, College Station, Texas, April 1995.

N. A. Izquierdo Jr. (M '76) earned his B.S.E.E. at the Illinois Institute of Technology, in 1978. He also completed post-degree master's level courses. Mr. Izquierdo began his career with Commonwealth Edison (ComEd) in 1975 in the Relay Section of the Operational Analysis Department. In 1978 as a Relay Settings and application Engineer in the System Planning Department, he was responsible for relay setting and fault calculation for two Chicago divisions and three generating stations. He also taught various ComEd after-hours protective relay courses. Since 1982, in the Relay Section of the Operational Analysis Department, he has been responsible for acceptance testing, including design and manufacture of new protective relays, developing relay test equipment, investigation and analysis of oscillograms after system disturbances, assisting the Regional Operational Analysis Department, General Office Engineering Departments, and laboratory personnel with special investigations and solutions of relay and power system problems. He currently holds the title of Staff Engineer in the Operational Analysis Department, Technical Services-Relay Group. He is a member of the Institute of Electrical and Electronic Engineers.

M. Kezunovic (S'77, M'80, SM'85) received his Dipl. Ing. degree in electrical engineering in 1974, and the M.S. and Ph.D. degree from the University of Kansas, in electrical engineering in 1977 and 1980 respectively. His industrial experience is with Westinghouse Electric Corporation in the U.S.A., and the Energoinvest Company in Sarajevo. His academic experience is with the University of Sarajevo and Washington State University. He has been with Texas A&M University since 1987 where he is an Associate Professor. He is member of the IEEE PSRC, member of CIGRE and a registered Professional Engineer in the State of Texas. Dr. Kezunovic is the chairman of the PSRC working group F-8 on " Digital Simulator Performance Requirements".

Z. Galijasevic received his B.S. and M.S. degrees from the University of Sarajevo, Bosnia and Herzegovina, both in electrical engineering, in 1985 and 1992 respectively. From 1985 to 1992 he was with the Power Electric Institute of the Energoinvest Company in Sarajevo. In 1993 Mr. Galijasevic joined Texas A&M University as a research engineer. His main research interests are in the area of digital simulation of electromagnetic transients and computer aided measurement.

F. Ji was born in Henan, China on February 11, 1965. She received her B.S. and M.S. degrees in electrical engineering from Tianjin University, Tianjin, China in 1985 and 1988 respectively. Ms Ji had been employed with Tianjin University from 1988 to 1991 as a teaching and research assistant. In 1994 she received M.S. degree in electrical engineering from Texas A&M University. Currently she is with the Texas Instruments in Houston.

A. Gopalakrishnan received his B.E. degree in Electrical & Electronics Engineering from BITS, Pilani, India in 1989. From 1989 to 1992, he was working in The English Electric Co. of India Ltd., in Madras, India. Mr. Gopalakrishnan is currently a graduate student in EE at Texas A & M University.

J. Domaszewicz earned his M.Sc.E.E. degree from Warsaw Technical University, Poland, in 1986. From 1986 to 1989 he was with the Department of Electronics at Warsaw Technical University. Mr. Domaszewicz is currently a graduate student at Texas A&M University.