NAG9-143
IN-20
217243
1-98

a
report

from the Texas A&M
RESEARCH FOUNDATION
College Station, Texas

(NASA-CR-185330) ADVANCED POWER SYSTEM PROTECTION AND INCIPIENT FAULT DETECTION AND PROTECTION OF SPACEBORNE POWER SYSTEMS Final Report (Texas A&M Univ.) 159 p CSCL 10B

N89-26042

Unclas
G3/20 0217243

FINAL REPORT

Advanced Power System Protection and Incipient Fault Detection

NASA - LBJ Space Center
NAG9-143

and

Protection of Spaceborne Power Systems

NASA - JSC10-86-8483

Prepared by:

Dr. B. Don Russell, Director
Power System Automation Laboratory
Department of Electrical Engineering
Texas A&M University
College Station, TX 77843

May 1989

Introduction

This research~~, funded under two sequential research contracts,~~ concentrated on the application of advanced signal processing, expert system, and digital technologies for the detection and control of low grade, incipient faults on spaceborne power systems. The researchers~~, led by Dr. B. Don Russell and Dr. Karan Watson,~~ have considerable experience in the application of advanced digital technologies and the protection of terrestrial power systems. This experience was used in the current contracts to develop new approaches for protecting the electrical distribution system in spaceborne applications.

The project was divided into three distinct areas:

1. Investigate the applicability of fault detection algorithms developed for terrestrial power systems to the detection of faults in spaceborne systems;
2. Investigate the digital hardware and architectures required to monitor and control spaceborne power systems with full capability to implement new detection and diagnostic algorithms; and
3. Develop a real-time expert operating system for implementing diagnostic and protection algorithms

Significant progress has been made in each of the above areas. Several terrestrial fault detection algorithms were modified to better adapt to spaceborne power system environments. Several digital architectures were developed and evaluated in light of the fault detection algorithms. Also a parallelized rule-based system shell for monitoring and protection applications in real-time was developed. The system was based on CLIPS and has been designated PMCLIPS.

Report on Research Progress

The most efficient means to communicate progress on these projects is through use of the publications generated by the project. The following is a list of the publications generated directly or indirectly as a result of this contract funding. Several of the publications include work funded by other organizations with more direct interest in fault detection in terrestrial power systems. However, it should be noted that most of the work is directly applicable to spaceborne systems. It should further be noted that even though the rule-based system shell was developed specifically for the spaceborne application, it would have application to terrestrial power system monitoring and control.

Research Publications

The following is a list of publications which fully describe the research activities of these contracts. The publications are attached and follow in the order they are cited.

1. Russell, B. D., Hackler, I. M. "Incipient Fault Detection and Power System Protection for Spaceborne Systems" (Proceedings, Intersociety Engineering Conference on Energy Conversion, Philadelphia, August 1987).

2. Russell, B. D., Watson, K. "Power Substation Automation Using a Knowledge Based System-Justification and Preliminary Field Experiments" (IEEE Transactions on Power Delivery, vol. 2, no. 4, pp. 1090-1097, October 1987).

3. Russell, B. D., Chinchali, R. P. "A Digital Signal Processing Algorithm for Detecting Arcing Faults on Power Distribution Feeder" (Presented at IEEE/PES 1988 Winter Meeting, New York, Paper No. 88 WM 123-2, 1988).

4. Watson, K., Russell, B. D., Hackler, I. "Expert System Structures for Fault Detection in Spaceborne Power Systems" (Proceedings, Intersociety Engineering Conference on Energy Conversion, Denver, August 1988).

5. Russell, B. D., Watson, K. "Knowlege Base Expert Systems for Improved Power System Protection and Diagnostics" (Proceedings, Symposium on Expert Systems Application to Power Systems, Stockholm-Helsinki, Norway-Finland, August 1988).

6. Russell, B. D., Mehta, K., Chinchali, R. P. "An Arcing Fault Detection Technique Using Low Frequency Current Components-Performance Evaluation Using Recorded Field Data" (IEEE Transactions on Power Delivery, vol. 3, no. 4, pp. 1493-1500, October 1988).

7. Russell, B. D., Chinchali, R. P., Kim, C. J. "Behavior of Low Frequency Spectra During Arcing Fault and Switching Events" (IEEE Transactions on Power Delivery, vol. 3, no. 4, pp. 1485-1492, October 1988).

8. Russell, B. D., Watson, K. "A Digital Protection System Incorporating Real Time Expert Systems Methodology" (CIGRE, Bournemouth, June 1989).

9. Watson, K., Russell, B. D., McCall, K. "A Digital Protection System Incorporating Knowledge Based Learning" (Proceedings, Intersociety Engineering Conference on Energy Conversion, Washington, D.C., August 1989).

10. Cook, G. "A Parallelized Rule-Based System Shell for Monitoring Applications" (Texas A&M University, Ph.D. Dissertation, December 1987).

Conclusion

It has been definitively demonstrated that the protection of spaceborne power systems can be improved using advanced digital hardware and software techniques. The algorithms which have been successfully used to detect incipient faults in terrestrial power systems can, with few modifications, be used to detect low-grade faults in spaceborne electrical distribution systems. Advanced digital architectures are needed to implement signal processing algorithms capable of distinguishing low-grade faults from normal system activity. The harmonic and noise activity generated by these low-grade faults can be detected well in advance of catastrophic failure. This is a direct improvement over existing protection systems which can only detect the overcurrents generated after catastrophic failure.

The findings of this research should yield improved reliability and serviceability of spaceborne power systems. By warning of incipient fault conditions, soft failures can be orchestrated, resulting in schedule maintenance before destructive failures occur. The digital architectures developed would be capable of monitoring and providing information concerning the ongoing operation of the power system, resulting in better human interaction and knowledge concerning the health of the system.

In summary, it is recommended that spaceborne power system protection be converted to advanced digital algorithms and architectures using signal processing and adaptive, expert programming techniques. The result will be an improved and safer protection system with higher availability.

Incipient Fault Detection and Power System Protection for Spaceborne Systems

B. Don Russell
Texas A&M University
Department of Electrical Engineering
College Station, Texas 77843

Irene M. Hackler
Propulsion & Power Division
NASA - Johnson Space Center
Houston, Texas 77058

ABSTRACT

Techniques are now available and being developed for terrestrial power systems that have the capability of detecting low current high impedance arcing faults. These techniques have the potential for use in spacecraft power systems to increase reliability, safety and maintainability. A study of the application of these techniques is underway to verify the applicability and feasibility of incipient fault detection for spacecraft power systems.

Power systems for manned spacecraft have been traditionally monitored from the ground and reconfiguration commands sent to the crew in the event of a serious fault or failure. This is not possible with the Space Station and future manned space programs as it would require many man-hours on either a ground system or consume many hours of crew time. Detection of impending failures, such as insulation breakdown, low current faults in loads, and corona, can be used to schedule repairs, perform load scheduling and adjust bus loads to reduce the stress on the involved components. Without this capability, repairs can only be done after a catastrophic failure has already occurred, and emergency system reconfiguration will be required with its potential interruptions in service and disruption of user power.

Techniques and hardware have been under research and development over the last ten years for earth-based utility systems that have the capability of detecting incipient faults. These techniques can be combined with the traditional methods of overcurrent, overvoltage, surge and transient protection to provide a more complete protection scheme for power systems. These same techniques, adapted to 20 kHz power distribution systems, can provide a level of protection not traditionally available in spaceborne systems.

Results of recent research indicate that by incorporating these methods of incipient fault detection, along with the traditional methods, into the Space Station remote power controller (RPC), it will be possible to design a safer, more reliable, and easier to maintain power system.

INTRODUCTION

NASA has several motivations for research in advanced power system protection. Spacecraft design issues which must be addressed at all levels include safety, reliability and maintainability. There are also the added complications of low gravity and vacuum operation. These environmental conditions affect all aspects of spacecraft design. Any system, whether it is operated on the ground or in space must be designed to be as safe as possible; however, manned spacecraft have very stringent requirements dealing with the safety of the crew and of the vehicle itself. Even simple accidents in the space environment can have major implications. For instance, surgery in low gravity is not yet considered to be feasible. So systems are specifically designed to prevent harm to the crew and vehicle.

Reliability of the power system is critical, because in the event of a power shortage or outage, there may not be sufficient backup power available during the time that repairs must be made to keep the crew's environment at a survivable level. It is always desirable to be able to schedule maintenance instead of having to do emergency repairs. Accurate and efficient fault detection and diagnostics in addition to the traditional power system protection techniques of overcurrent, overvoltage and transient protection will enable a safer, more reliable and easier to maintain power system.

NASA is planning to build larger and more complicated spacecraft that require more power to operate. These larger power systems will have to operate in a more autonomous mode than spacecraft have in the past. This is especially true of manned vehicles. The long range planetary explorers have operated autonomously, but the power level is very low. In large manned operations, such as the Space Station, the crew manning the installation will be spending a great deal of their time with experiments and space research, spending as little time as possible with maintenance and repair. Crew time is considered to be a valuable and limited resource, as is extravehicular activity time (EVA). The more autonomous and trouble free a system is, the more crew time can be spent with the experiments and less time spent on inside or outside repairs. With the long life envisioned for this program, ground operations will also be kept to a minimum. This leads to systems that operate without the constant supervision of the crew or ground controllers. Fault detection, diagnostics and in-place redundancy become very important in this situation.

The current preliminary design of the NASA space station requires 87.5KW of power initially, with 37.5KW devoted to housekeeping and 50KW available for experiments and payloads. The power distribution system uses 20 kHz, single phase 440 Vac outside of the manned habitat and 208 Vac inside the manned elements. The power to operate the Station comes from sunlight that is converted to electricity using either solar collectors that run a dynamic heat engine or photovoltaic arrays.

Protection philosophy to date has centered on ground monitoring of spaceborne systems with manual initiation of load reconfiguration and isolation. While certain surge and overcurrent systems have been used, the majority of the "protection intelligence" has resided in ground operators who are monitoring information from spacecraft. Fault detection and diagnostics for any part of the power

Presented at the Intersociety Engineering Conference on Energy Conversion, Philadelphia, August, 1987.

system will enable the crew to spend more of their available time with payloads and experiments and will allow the ground operations to be kept to a minimum. Safety is enhanced because accurate fault detection and diagnostics will enable procedures and techniques to prevent harm to the crew and vehicle. With this large and complicated a system, failures that do occur must not cause harm to either the power system, the users, the vehicle, or the crew. Maintenance is also very important. The life of the Station is designed to be at least 10 years and probably be actively used for 30 years. Over that long a time period, failures are bound to occur and if failing components can be detected and replaced before an actual failure occurs, maintenance can be scheduled, as opposed to using emergency procedures. Detecting a potentially catastrophic failure before it can cause harm or serious power outages would be a very useful capability.

A program was initiated with Texas A&M University to study the feasibility of using advanced terrestrial power system protection techniques for spacecraft power systems. These systems, in research and experimental phases, have enabled the detection of high impedance, low current arcing faults and have given advanced notification to the users of insulation breakdown that can lead to more serious problems. These techniques have been combined with the traditional methods of overcurrent, overvoltage, surge and transient protection to provide a more complete protection scheme for power systems. This program was started to enhance and automate spacecraft power distribution systems in the areas of safety, reliability and maintenance.

PROPOSED POWER MANAGEMENT/DISTRIBUTION SYSTEM

The Power Management and Distribution (PMAD) system for the Space Station will resemble a terrestrial utility power system in many respects. The central system will serve various loads which may be operated randomly and periodically. The energy levels are relatively high by space craft standards and the duty cycle will vary significantly as a function of the type of load and various operating scenarios. The randomness of this system allows for a degree of load diversity which is an advantage. However, it introduces unknowns into the power distribution system which makes protection more difficult. The proposed PMAD system is designed to distribute and manage electrical energy in light of the large number of random loads which could potentially be operated in the Space Station. [1] PMAD extends from the Main Power Conditioner (MPC), which converts electrical power, from the power source and storage, to a form which is distributed throughout the station to the user ports or interfaces.

Functional Requirements of PMAD

1. Provide electrical power source conditioning and the transmission and distribution of utility power to the user interface.
2. Protect against open circuits, overloads and short circuits in the distribution system and against overloads and short circuits in the housekeeping and user loads.
3. Provide system reconfiguration to isolate failed components and/or switch in redundant paths between sources and loads.
4. Meet desired failure tolerance criteria.

Power Distribution Hierarchy of PMAD

Electrical energy from solar arrays and storage cells is conditioned to a usable form by MPC and distributed via two ring type primary feeders. Primary distribution Remote Bus Isolators (RBI) are used to reconfigure power distribution during fault conditions. Secondary distribution of power is achieved by the Power Distribution and Control Assembly (PDCA). PDCA distributes power to the Load Converters (LC), if required, via load RBIs and Remote Power Controllers (RPC). The Load Converter converts power to a form usable by the load. The power distribution hierarchy is depicted in Figure 1.

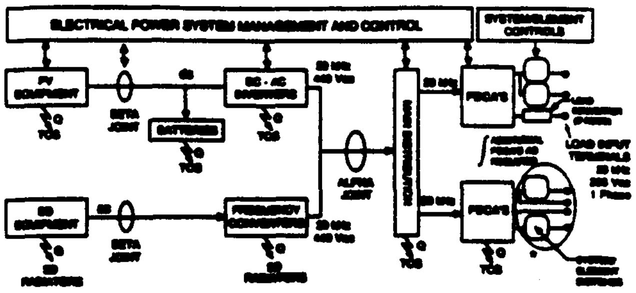


Figure 1. Electrical Power System Architecture

Data/Communication Hierarchy of PMAD

Data/Communication between Main Bus Controller (MBC) and PDCA is achieved using two PMAD data buses. The primary distribution RBI's communicate directly with the MBCA while the load RBIs and RPCs communicate with two Power Distribution and Control Unit (PDCU) controllers located within the PDCA. The PDCU controllers can also communicate mutually via an interconnected full duplex link. The Data/Communication hierarchy of the PMAD is depicted in Figure 2.

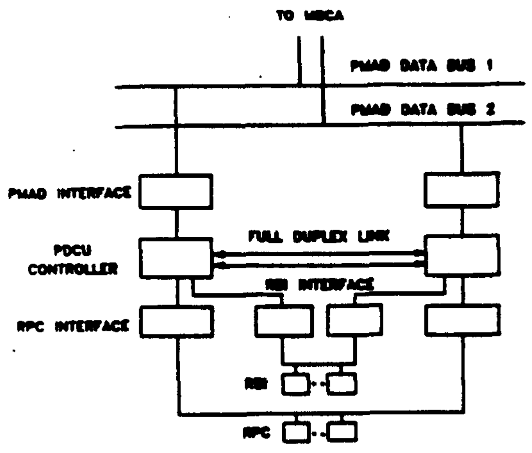


Figure 2. Data/Communication Hierarchy

Power Distribution and Control Assembly (PDCA)

PDCAs are used to distribute power at the load areas. Each PDCA consists of two Power Distribution and Control Units (PDCU) that contain:

1. Primary feeder Remote Bus Isolators (RBI)
2. Load RBIs and Remote Power Controllers (RPC)
3. Sensors to provide necessary monitoring and protection information
4. PDCU controller, an embedded processor that interfaces with RBIs, RPCs, Sensors and data buses.

RPC is a solid state switch used to control secondary distribution of power. The RPC can be used to turn power off to a user and rapidly remove power within microseconds to inhibit fault propagation. It also provides data acquisition of voltage and current quantities.

The RBI is an electromechanical switch used to isolate faults and reconfigure primary and secondary distribution over electrical power. It also can provide status and analog information similar to an RPC.

The RPC is limited to providing protection against overloads and faults using I^2t and fast trip and minimal status information. RPC is primarily a dedicated switch with limited intelligence or ability to communicate with other devices on an interactive basis. Due to these limitations in RPC and RBI intelligence, very little or no facility exists in the PMAD for data acquisition and processing, advanced protection, diagnostics, etc.

The RPC/RBI design should be enhanced to support the functions of intelligent remote, digital relaying, and knowledge based diagnostics.

PROTECTION REQUIREMENTS: SECURITY ASSESSMENT AND CONTROL

The security of the power system has to do with its ability to withstand disturbances such as electrical short circuits or the unanticipated loss of system components. There are several levels of security assessment and the subsequent control which attempts to reconfigure the distribution system for optimal operation under disturbance conditions.

The space craft power system should have multiple layers of security assessment and control which could be classified as follows.

Level 1: The lowest level of load interface should be through devices capable of providing fast response to catastrophic load failure or short circuit. At this level, stand alone protection devices continually monitor the electrical parameters of loads for abnormal conditions. Conventional overcurrent devices fall in this category guarding against thermal overloads, surges, and short circuit conditions.

Level 2: Multiple loads and feeders can be monitored to determine that all power system components are operating within established load limits and within voltage ratings. This is conventionally called static security assessment and monitors the system to determine that steady state operations are acceptable. Load management and allocation of energy between loads is included at this functional level. On-line load analysis and state estimation can be used to create a comparative data base for assessment under normal operating conditions. Load scheduling and cycling are also included.

Level 3: Level 3 security consists of automatic, system wide monitoring. It includes reconfiguration under disturbance conditions and deals with the power system under unstable or dynamic conditions. Dynamic security can involve transient stability and voltage-var security. New concepts include online historical trending of system operations and monitoring of loads to determine unacceptable "patterns" pointing to system degeneration or points of failure. Incipient fault detection is included at this level along with knowledge based diagnostic analysis.

In the conventional spaceborne power system, level 1 security has been provided by overcurrent devices which connect loads to power supply. Level 2 security has been provided by monitoring and manual reconfiguration of loads after failures have been detected. Level 3 security has essentially not existed in conventional systems.

It is proposed that a multiple level security assessment and control philosophy be adopted for the space station using hierarchical processors each providing appropriate data for various levels of security assessment. It is this philosophy that should guide the functional design for the space station protection system.

INCIPIENT AND LOW CURRENT FAULT DETECTION

A significant problem for terrestrial power systems is the detection of very low current or incipient fault conditions in power apparatus and distribution lines. While the problem has not been totally solved, significant research has resulted in several proposed techniques for detecting certain incipient faults in equipment and low current faults on distribution feeders. [2-3]

A low current fault is defined as a fault sufficiently low in magnitude such that it cannot be detected by conventional overcurrent protection devices. Overcurrent devices must, of necessity, be designed to operate for currents which exceed normal operating load current limits. However, it is both possible and common to have short circuit conditions through a sufficiently high impedance to limit the current flow below normal load levels. This type of fault is very dangerous in its potential consequences and cannot be detected by conventional devices. Such faults are common in distribution feeders with distributed loading and in insulated cable.

Research to date has concentrated on the investigation of abnormal signal patterns associated with low current faults as compared to normal load signals. An intelligent protection device should be capable of monitoring feeder currents and voltages to detect abnormal patterns resulting from high impedance short circuits and incipient fault conditions. [4]

Detection of incipient, arcing faults is based on several characteristics of these faults suitable for use as detection parameters. Some of these are as follows.

Arcing faults exhibit an increase in nonfundamental frequencies. Typical loads exhibit relatively slow changes in feeder noise levels over time.

Arcing phenomena exhibit a random/transient (burst) nature.

Arcing phenomena often persist over relatively long periods of time.

Nonfundamental frequency components propagate over power distribution feeders and their presence is an indicator of abnormal activity.

Transients from switching activity are typically limited in time to a few cycles and do not persist.

By using the above criteria in an "intelligent" software package, it has been possible to detect incipient and other low current arcing faults with a degree of selectivity with respect to normal feeder events. The key to effective detection is the identification of signal patterns and parameters which change substantially between faulted and unfaulted conditions. The techniques used to date differentiate by neglecting the random changes and synchronous power system frequencies, using nonsynchronous signals for detection.

Figure 3 shows a "snapshot" of current on a faulted feeder. The lower trace shows 60Hz load current. The top trace shows burst noise above 2kHz due to the presence of a low current fault. Figure 4 shows a single expanded fundamental cycle processed to eliminate the fundamental and lower harmonics. It can been seen that significant information is contained in the signal indicating the presence of the arcing fault even though the magnitude of the fundamental has not significantly increased. In short, such a fault would not be detected by conventional systems which anticipate an overcurrent condition for catastrophic failure.

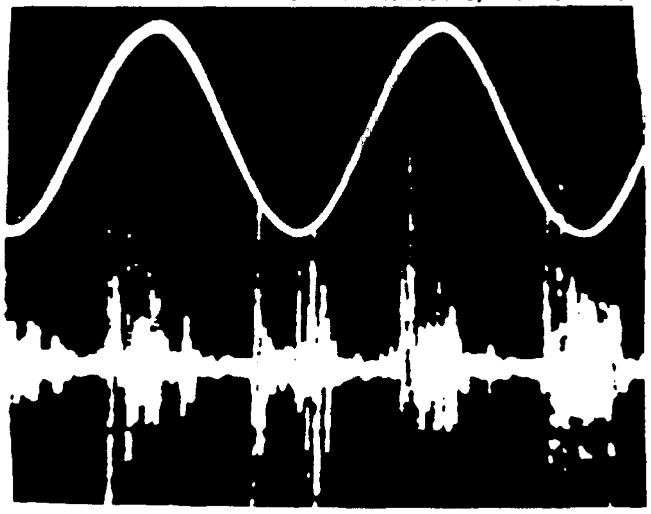


Figure 3. Faulted Feeder Current

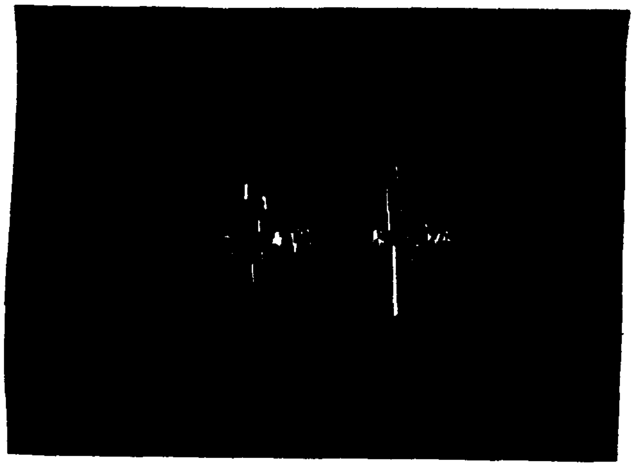


Figure 4. Feeder Burst Noise

By recognizing the abnormal signal patterns using intelligent processing, such faults can be detected possibly even before catastrophic failure.

Detection Technique

One fault detection technique which holds significant potential is based on the detection of energy in nonfundamental and nonharmonic current frequencies serving the device in question. It is possible for detectors to identify burst noise energy, caused by arcing faults, at frequencies higher and lower than fundamental. Generally, harmonics of the fundamental power frequency are not considered in the detection due to the influence of normal load changes and switching parameters on these frequencies. However, previous work has shown that significant sensitivity to arcing faults and ground faults can be achieved using these other frequency components.

The system architecture for an arcing fault detector using a burst noise energy evaluation can be described as follows. The input signal coming from the single phase feeder is appropriatly filtered to attenuate fundamental and harmonic frequencies. The output signal from this filter bank is passed to an analog converter. The sample waveform data is then presented to a software detection algorithm resident in a dedicated protection processor. This processor performs appropriate detection evaluations and determines whether a trip should occur or an alarm condition is indicated.

Detection Scheme

A very simplied flow chart is shown in Figure 5 to indicate the logic incorporated in a burst energy detection algorithm. The detection technique utilizes the summation of the square of the filtered frequency data samples over an entire cycle of fundamental frequency. The detection algorithm does not consider individual impulses in the filtered signal, but attaches importance to their cumulative effect over a predefined time. Characteristics of the software detection algorithm include its adaptability, hierarchical nature, and "expertness".

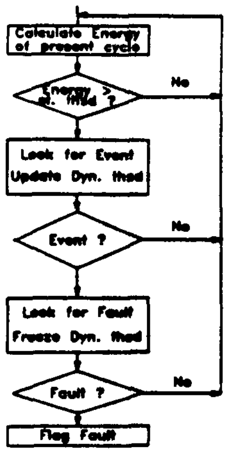


Figure 5. Detection Algorithm

It is imperative for the algorithm to track variations and the low frequency signal which are not associated with faults. Each feeder may have a different normal noise level which itself may be dependent on and vary with the load. A typical distribution system exhibits a periodic load cycle. Hence the detection algorithm must be adaptive to these changes in the load and at the same time avoid a difficult procedure for calibrating arbitrary pick up levels for fault detection. Hence, the algorithm used to detect arcing faults factor was used in the experimental program. Typically, five cycles would be tested where 3 of 5 showing the requisite increase might dictate a necessity to proceed onto the next hierarchical level of fault detection. By varying the number of test cycles, the sensitivity of detection can be varied.

Fault: Once an event is recognized, control is transferred to the fault identification routine in the detection algorithm. The dynamic threshold is frozen in the fault identification routine. This action is necessary because percentage (relative) change in the signal magnitude is used to detect faults as opposed to absolute changes from predefined fixed thresholds. One choice involved is the number of cycles after an event, required to show an increase in energy density, before a fault is specified. The compromise is between correct identification of intermittent or very low grade faults and the possibility of identifying a normal event as a fault. Thus, an important parameter involved is the choice of the length of time after an event that is allowed for evaluating the non-fundamental current to make the trip/notrip decision.

The various parameters involved are adjusted so as to obtain an "optimal" detection scheme which takes a reasonable length of time for making decisions and is sensitive enough to make distinctions between arcing faults and normal operations.

Validation of Detection Techniques

The detection technique has been validated using data taken from low current faults, arcing faults and normal switching data on terrestrial electric power systems. Data analysis on arcing fault and normal switching data indicated that several harmonic and sub-harmonic frequencies could be used to distinguish between incipient faults and normal switching events.

The technique described is only one of several approaches to detecting abnormal, incipient, and fault conditions on feeders using nonfundamental voltage and current parameters. Other detection techniques include identification of abnormal "patterns" caused by events other than normal load activity. It is anticipated that a combination of these several techniques along with conventional digital relaying for overcurrent protection would result in a secure and sensitive protection system. in a typical distribution system should have strong adaptability to these periodic variations in the load to maintain reliability of detection. The feature of adaptability has been incorporated in the detection algorithm by making use of two thresholds - a dynamic threshold and a static threshold. The thresholds are used to adjust changes in the load on a continuous basis.

A hierarchical nature is bred into the algorithm by making use of three levels in the detection process before signaling a fault. The system starts by recognizing a "disturbance" and on the occurrance of a disturbance, the system devotes its attention to trying to verify if the disturbance qualifies as an "event". An event recognition is followed by an attempt to classify the episode into either a "fault" or a normal occurence. The progression of the detection scheme from one level to another and the updating of all values through this progression is automatic. The progression also implies the use of time as a discriminatory factor. The definitions for these hierarchical levels are as follows.

Disturbance: A cycle of data showing a certain percent increase of energy over the average energy per cycle, the average being calculated over some previous period of time, constitutes a disturbance. Thus, if a cycle shows a certain percentage (e.g. 25 percent) increase in energy over the previous average, a disturbance is said to have occurred. If the energy present in the present cycle is reasonably equal to the previous average, then a new average is calculated and disturbance detection is begun again. The purpose of the disturbance detection routine is to identify changes in the non-fundamental current on the feeder. Such an occurrence could be one of any number of events such as load drop or addition, switching event, bolted fault, or high impedance fault.

Event: Once a disturbance is detected, a preselected series of cycles of data are tested. If a set percentage of these cycles show a certain percentage increase of energy per cycle over the average energy per cycle (the average being calculated over some previous period of time), then an event is said to have occurred. A point of interest here is the fact that the dynamic threshold is updated even after the recognition of a disturbance. Statistical analysis has shown that a 75 percent increase in component energy is reasonable for event identification, and this It is further anticipated that the techniques could apply in a modified fashion to spaceborne systems though fundamental frequencies of operation are higher than those used by terrestrial electric utilities. Experiments are currently underway to extend these techniques to proposed spaceborne power system designs. A simulated power system is currently being tested at 20 kHz operation to validate the previously used detection techniques and determine their performance.

PROPOSED SPACEBORNE PROTECTION SYSTEM

In light of the security philosophy previously described and the research performed on the detection of low current faults on terrestrial power systems, a more comprehensive protection system is proposed for the space station. It is proposed that the protection system have a security hierarchy with intelligent processors at each level for both protection and data acquisition purposes. This is similar to recently demonstrated systems for terrestrial substation use. [5] The higher level computers would receive data from the lowest level Intelligent Remote Power Controller (IRPC). The higher level computers would run online contingency analysis programs and security assessment programs to determine optimal reconfiguration patterns under various operating scenarios. Upon receipt of information from lower level devices that a device is failing or has failed, these higher level programs would determine reconfiguration options, schedule maintenance, reduce loading on failed or degenerated components, and provide security assessment and warning to space craft personnel. Online load management, load flow, and state estimation programming could be performed at this level.

At the lowest level of the protection hierarchy, an IRPC is proposed which combines the features of an intelligent remote terminal unit and an adaptive processor based protection device.

Functional Requirements - IRPC

The IRPC must perform those functions previously proposed for the RPC. The intent is to provide the basic function of connecting user loads to the power system while providing level 1, level 2 and level 3 security.

The IRPC must provide the "intelligent switch" capability allowing power to be turned on or off under either automatic or manual positions. It must implement the conventional overcurrent trip functions and should have remote setting capability for these trip levels. It should also serve as the first level of data acquisition for the hierarchy computer system. Voltage, current, and status information should be provided from the IRPC based on sampled data analysis. The same inputs can be used for local digital overcurrent relaying functions as well as for inputs to higher level data bases.

Once the decision is made to create a unified data base in IRPC from sample data inputs, numerous other functions become feasible. The IRPC should be viewed as an intelligent remote terminal unit and an adaptive digital relay. From a unified data base, it is possible to support numerous algorithms for protection purposes including current, overvoltage, undervoltage, and frequency variation detection. Where appropriate, impedance/distance trip could be implemented. Additionally, sequence of events, time tagging, and fault recording functions could be implemented at selected IRPCs where these functions are desirable.

While these advantages of an IRPC are significant, possibly the greatest advantage lies in the detection of incipient faults or abnormal flow trends. Using an IRPC data base, it is considered possible to detect, using knowledge based system approaches, abnormal load trends which indicate pending failures of equipment or conditions which need specific attention. Incipient cable faults, equipment degeneration or failure, could also be detected prior to the occurrence of catastrophic overcurrent events.

Functionally, the IRPC should be a standalone unit capable of performing most functions without connection to the computer hierarchy. However, its operation as a remote terminal unit allows for much of the local information from the IRPC to be selectively passed to higher level systems for the purposes of security analysis and control.

Proposed Operating Scenario For IRPC

Under normal conditions an IRPC would provide status, analog, and load information to higher level computers on demand or as required. Such information could be dynamically used by higher level processors to support state estimation, load flow, load management, and other programs.

When an unpredicted catastrophic fault or load failure occurs, an IRPC would react using one or more protection algorithms to detect the abnormality and immediately break the connection to prevent fault propagation and equipment damage. This load loss would be reported along with information concerning the type of failure.

On a continual basis, the IRPC will serve as an on-line diagnostic unit monitoring loading trends, load characteristics, waveform patterns, etc. When abnormal conditions are detected based on historical data files and knowledge based algorithms, crew members would be informed through the computer hierarchy as managed by higher level processors. First stage warnings would call for maintenance or investigation of an impending or potential problem. If the condition becomes more severe, alarm conditions can be indicated and, when necessary, conventional protection action can occur.

In summary, the IRPC would have three parallel areas of activity.

(A) Intelligent remote terminal unit.
(B) Conventional protection functions from sampled data inputs.
(C) On-line diagnostics, evaluation, and trending.

The difference in proposed RPC designs and the IRPC proposed here can be summarized by the term "intelligence". Programmability and flexibility would result from a design based on sampled data inputs to a digital processing environment. Digital relaying could allow for easily set protective parameters and adaptive algorithms.

The digital processing capability inherent in the IRPC would also allow for implementation of diagnostic routines and creation of various data bases for purposes of load evaluation, load management, and long term trending. The intelligence of the device yields flexibility which in turn should yield improved protection and information concerning system operations.

Conclusion

The proposed PMAD system performs many valuable functions and is logically divided into a hierarchy of equipment modules. However, the performance of the system can be significantly improved by the proposed IRPC design. The IRPC would be an intelligent data acquisition detection device with load switching capability. Such a device would allow for implementation of digital relaying algorithms with both adaptive and programmable characteristics. The data acquisition system inherent in the IRPC would allow for the creation of data bases to improve load management and diagnostic functions. Using sample data input, signal processing algorithms with knowledge based system features could be used for incipient fault detection and to detect abnormal trends.

Research is needed in the development of specific protection algorithms and knowledge based diagnostic systems which take into account the physical configuration, characteristics, and operation of the space station. It is anticipated that certain elements of terrestrial protection can be used to improve previous spaceborne power system protection practices.

Current investigations include determination of the effects of using a high fundamental power frequency on the characteristics and nature of faults. Also being investigated is an improved rule-based monitoring software system which, in fast realtime, can support the diagnostic functions proposed here in. Experiments are underway to test the appropriateness of techniques used in terrestrial fault detection for spaceborne applications. Results are expected soon.

References

[1] Space Station Architectural Control Document - Electrical Power System, JSC-30263, Jan. 1987.

[2] "Detection of High Impedance Faults," Electric Power Research Institute Report EL-2413, Power Technologies, Inc., June 1982.

[3] "Detection of Arcing Faults on Distribution Feeders", Electric Power Research Institute Report EL-2757, Texas A&M University, December, 1982.

[4] Aucoin, B. M. and Russell, B. D., "Distribution High Impedance Fault Detection Utilizing High Frequency Current Components", IEEE Transactions on Power Apparatus and Systems, Vol. PAS-101, No. 6, June 1982, pp. 1596-1606.

[5] Aucoin, M. Zeigler, J. and Russell, B. D., "Feeder Protection and Monitoring System, Part I: Design, Implementation, and Testing", IEEE Trans. on Power Apparatus and Systems, Vol. PAS-104, No. 4, April 1985, pp. 873-880.

Irene M. Hackler is a space engineer with NASA-Johnson Space Center in Houston, Texas. She received a B.S. degree in Electrical Engineering from the University of Wisconsin and a M.Sc. in Physical Sciences from the University of Houston.

Ms. Hackler's professional interests are in the area of power systems for space vehicles including the generation, conversion, storage, and distribution of energy. Ms. Hackler's current position is with the Power Branch of the Propulsion and Power Division of JSC.

B. Don Russell is currently a Principal Investigator with the Texas Engineering Experiment Station at Texas A&M University. He received his B.S. and M.E. degrees in Electrical Engineering at Texas A&M and holds a Ph.D. from the University of Oklahoma in power systems engineering.

Dr. Russell directs the activities of the Power System Automation Laboratory of the Electric Power Institute, Texas A&M University. His research centers on the use of advanced technologies to solve problems in power system control, protection, and monitoring. Dr. Russell's current interests lie in the detection of low current, incipient faults on power distribution systems using "intelligent" microcomputer-based systems.

Dr. Russell is a member of the IEEE Power Engineering Society and chairs several subcommittees and working groups. He is a registered professional engineer and a Director of the Texas Society of Professional Engineers.

Power Substation Automation Using A Knowledge Based System - Justification and Preliminary Field Experiments

B. Don Russell
Senior Member, IEEE

Karan Watson
Member, IEEE

Texas A&M University
College Station, Texas

Abstract

Conventional automation and control devices are passive and responsive. These devices typically "respond" to changes in the power system as measured against fixed thresholds and preset limits. These systems are designed to prevent catastrophic failures and incorrect operations and work well for general data acquisition and remote supervisory control. However, feedback control, system diagnostics, advanced protection, and contingency control are difficult to implement on the ever changing power system with these conventional approaches. Knowledge based systems have the potential for following the changes in the power system and adjusting decision criteria accordingly. Decisions can be made on a more complete data base which is constantly adjusted to changes in system parameters and operation.

This paper describes the various functions where knowledge based systems could ideally be used. The use of a knowledge based, adaptive system approach for diagnosing distribution system disturbances and equipment failures is presented. Two field experiments are described.

Introduction

Substation automation can be conventionally segmented into the functions of data acquisition, control, protection, diagnostics, and monitoring. While there is considerable overlap between these categories, each has come to represent a set of automation functions within the power substation.

An excellent description of the numerous substation functions and how they are interrelated has been prepared by the Application of New Technologies Working Group of the Automatic and Supervisory Systems Subcommittee of the Substation Committee.[1] These functions are grouped as follows.

86 T&D 600-1 A paper recommended and approved by the IEEE Substations Committee of the IEEE Power Engineering Society for presentation at the IEEE/PES 1986 Transmission and Distribution Conference, Anaheim, California, September 14 - 19, 1986. Manuscript submitted March 20, 1986; made available for printing July 30, 1986.

Table 1

Substation Automation Functions

Supervisory Control Functions
- Trip Close
- Off On
- Automatic/Supervisory

Data Acquisition
- Analog
- Indication with Memory
- Accumulator
- Sequence of Events
- Analog Data Freeze
- Demand Data Retrieval
- Pre/Post Fault Recording

Data By Exception
- Status
- Analog - Variable Dead Band
- Analog - Limit Violations
- Analog - Change Alarms
- Operations Counting

Function Check
- Self Diagnosis
- Analog Calibration

Automatic Control
- Circuit Recloser
- Line Sectionalizer
- Load Throwover
- Reclosing Equipment
- Opening Equipment

Automatic Control Functions
- Voltage Var Control
- Automatic Transformer
- Load Matching
- Capacitor Switching
- Load Reduction
- Load Shedding
- Automatic Start/Stop Sequence

Calculation Of Control Parameter Or Data
- Local Load Flow
- Load Survey Computations
- Fault Locator

Data Recorded On Local Output Media
- Analog Parameters
- Event Recording
- Sequence of Events
- Fault Data
- Digital Sampling of Analog Waveforms
- Other Functions:
  - Local Analog Logging

Protection Functions
- Breaker Failure
- Consistency Checks
- Instantaneous Relaying
- Time Overcurrent Relaying
- Adaptive Relay Curves
- Arcing Fault Detection
- Line Noise Indication
- Transformer Temperature

RTU Functions Initiated By Master Station
- Down Line Functions
  - Loading of Data
  - Load Curtailment
    - Voltage Reduction
- Coordination Functions
  - Time Synchronization
- Access by Multiple Master Stations
- Bulk Data Transfer
  - Analog Waveforms
- Analog & Status Data
- Historical Data

Data Transfer
- Analog Output
- Digital Output
- Output to Line Printer
- Computer to Computer Data
- AGC Pulse
- Binary Output to Local Digital Meters

Careful review of these functions will show that many can be performed by stand-alone systems which have satisfactorily performed these operations for years. However, other functions require more sophisticated equipments not yet typically used by electric utilities. The subject of substation automation encompasses all of these functions as they are implemented in the power substation.

In general, each of these functions could be performed by a functionally independent device in segregated hardware. However, careful study shows that the data base required for most of these functions is very similar and a level of integration is certainly advisable. The integration of various control, monitoring, and protection features has been proposed for 30 years with serious investigation during the last 10 years. [2,3] Several research efforts have led to integrated systems demonstrating monitoring, control, and protection features. [4,5] These developments which have been field tested show the practicality of performing many functions from one piece of equipment with a unified data base. Distinct advantages are observed including a reduction in functional redundancy, a reduction in hardware complexity, potential cost savings and improved reliability and diagnostics.

However, it has also become obvious from these demonstrations that more "intelligent" systems must be designed if the true benefits of integrated substation automation are to be achieved. It is unacceptable to simply repeat conventional functions in an integrated equipment design. While this is a step forward it does not take full advantage of the significant improvement in operations which can occur using the power of computer based systems. A need exists for knowledge based systems which not only "respond" to changes in the power system against preset parameter thresholds and setpoints, but which can change and adapt their operation to meet long term trends and changing conditions.

Potential For Knowledge Based Systems

While there are numerous automation functions which could be improved in their implementation using expert systems, several are very obvious and can be used to demonstrate the need.

One proposed use for expert systems is in the identification and location of fault sections after the operation of protective relays. The intent is that a tool be available to dispatchers in emergency situations to assist in restoration procedures. This approach proposed by Fukui and Kawakami of Hitachi in Japan is an excellent example of how a knowledge based inference system can be used to diagnose sequence of events following a disturbance. [6]. This system has been implemented using the Prolog computer language. Its data base consists of information concerning protective relays which have operated and circuit breakers which have tripped. From this information, inferences are made to estimate the fault section using a knowledge based approach. The intent is to emulate the mental process used by an expert dispatcher to achieve the same conclusions.

This approach can be applied to many other functions in the substation. The conventional equipments used in substations and most of the recently proposed automation approaches do little to "diagnose" the integrity and health of the substation and power system. If conditions deteriorate to the point that a catastrophic failure results, then protective devices and other control devices initiate action to reduce or isolate the catastrophic effect. However, many of these failures may have resulted from conditions which developed over hours or days eventually resulting in a catastrophic breakdown.

Careful analysis of Table 1 shows several functions which are better accomplished by a system capable of calculations and logic based not only on present values of data, but historical data and trends. For example, transformer loading can be optimized only if historical loading data and current information are simultaneously available. The need for transformer maintenance and inspection after faults is not only a function of the severity of the last fault but the cumulative effects of fault duty experienced by the apparatus. The calculation of fault location on a line can be improved in terms of accuracy by including feedback as to the error in previous calculations. In each of these cases inferences from historical data or other calculations are used to improve performance of the automation system. Other functions such as fault detection, equipment deterioration, and data validation can also be significantly improved using knowledge based system techniques.

Field Experiments

It is proposed that an expert or knowledge based system be used to constantly monitor the integrity of the substation, its equipment, and operations and provide advanced information and warning of impending problems. The potential for this technique has been demonstrated in two experiments by researchers at Texas A&M University. Experiment 1 related to the detection of incipient fault conditions which were insufficient in their severity to be detected by conventional protection and monitoring systems. Experiment 2 related to the detection of equipment breakdown over a long period of time. These experiments are described as follows.

Experiment 1-Incipient Fault Detection

It has long been known that many distribution faults are not severe enough to be detected by conventional protection and monitoring devices. Field experiments, staged fault tests, and fault statistics have shown that many ground faults begin and remain below conventional detection thresholds.

In response to this characteristic of distribution faults, work has been performed for many years to improve the overall sensitivity of fault detection devices to include these low current incipient faults. This work has provided several techniques including those developed by Power Technologies, Inc., Pennsylvania Power & Light, and Texas A&M University. [7,8,9]

These research investigations have shown that it is generally not sufficient to simply monitor changes in 60Hz fault current and voltage components to determine that these incipient conditions exist or that low current faults have occured. Other approaches including a much broader data base and the use of historical data are indicated.

In experiments by Texas A&M University, a broad data base including high frequency information above 2kHz was used to detect these low current faults. Simply stated, since a ground fault typically generates an arcing condition which modulates current, high frequency noise is generated which propagates to the substation. This noise has characteristic patterns which can be detected and used as fault indicators.

However, in the development of this technique, it became obvious that normal system changes including feeder noise levels were dramatic. If fixed protection thresholds are used, it is highly probable that many false trips will occur due to normal system variations.

Figure 1 shows the high frequency current noise on a feeder before and after a ground fault. It is obvious from this figure that the faulted section has considerable noise components which could be used as a fault indicator. However, the selection of an arbitrary threshold of detection between pre and post fault levels is difficult due to the fact that the normal noise patterns on the feeder change precipitously over a broad dynamic range. In short, the conventional approach of determining an acceptable vs. an unacceptable level of current at a given frequency and using this as a fault detector is insufficient and will yield an insecure protective device.

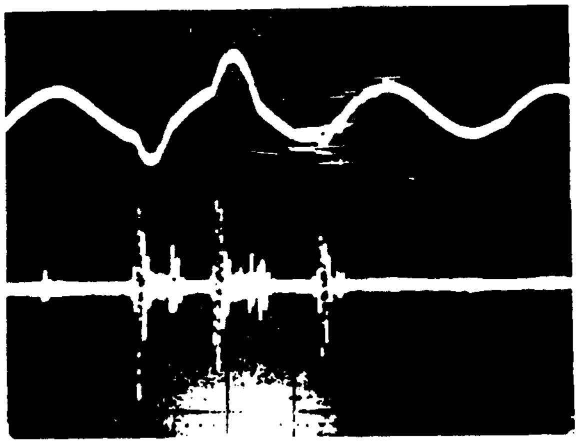


Figure 1 - Pre/Post Fault Feeder Noise

Through considerable analysis of recorded data, it was determined that the high frequency on the distribution feeder tended to change over the short and long term related to such factors as the level and type of loading on the distribution feeder. As an example, noise levels under heavy loading during the day might be very high, whereas corresponding measurements taken at night under light load conditions might show little noise activity above 2kHz. The need exists for a "variable sensitivity" device which can adapt not only to daily changes but to long term, seasonal load changes or increased loads due to circuit reconfiguration.

Techniques were developed which compared the present value of high frequency current of the faulted section, not to a preset threshold, but to a calculated value which was a function of historical measurements of the parameter. Simply stated, the threshold was changed dynamically and adapted to the changes in the power system providing a detection threshold which could rachet up or down based on normal persistent system changes.

Additionally, certain noise generating equipments on the distribution feeder have patterns which can be identified as a function of their repetitive nature and how they occur with reference to the 60Hz waveform. Other switching activity also has specific patterns of behavior which are unique and identifiable.

Experimentation has shown that these features are ideal candidates for processing by an expert system resulting in a high probability of fault detection in spite of dynamic, long term and short term changes in normal system activity.

During field tests of the arcing fault detection technique it was shown that a secure discrimination between a dynamic normal system and a faulted system could only be made using historical data as opposed to instantaneous measurements of parameters.

Experiment 2-Equipment Failure Diagnosis

During the numerous field tests and measurements made over several years it was determined that equipment breakdown and deterioration may occur over long periods of time. For example, an incipient transformer fault due to insulation breakdown may occur very slowly before resulting is a catastrophic failure. Other apparatus such as insulators on distribution feeders may have intermittent breakdown due to incipient mechanical failure which persists for weeks prior to causing a high current fault. During experiments with Public Service Company of New mexico, changes in current waveform frequency components were detected on a specific feeder over many weeks of monitoring. The changes in high frequency activity were easily measured and at times were precipitous resulting from insulation breakdown. However, since the fault was not mechanically sustained, system integrity was restored and all indications of the presence of the breakdown were lost.

By careful study of this feeder it was determined that certain arrestors and insulators were failing and repairs were needed. Effecting these repairs, the current waveforms on the feeder changed accordingly resulting in a reduction of the noise patterns previously measured. Careful analysis of these noise patterns as correlated to other data predicted to the line problems.

This experiment indicates the potential for diagnosing the need for equipment repair and line maintenance. A carefully designed expert system looking at numerous parameters can, through inferences, determine the most probable cause of incipient conditions and prioritize the actions taken to restore the system to 100% integrity.

Characteristics of Knowledge Based Systems

Certain classes of knowledge based systems are appropriate for use in addressing power system problems. It is important to understand the characteristics of these systems and how they can be adapted to a specific situation.

ORIGINAL PAGE
BLACK AND WHITE PHOTOGRAPH

Knowledge based systems are a subfield of artificial intelligence. These systems incorporate the strengths of advanced programming techniques in order to solve specific problems. If the information necessary for the solution of a problem can be encoded in tractable algorithms, conventional computing systems should be used. However, since many significant problems cannot be described by a reasonable set of algorithms, a different strategy is needed to incorporate computing machines.

The observation that human experts often solve problems without focusing on a formal reasoning method lead to the development of knowledge based systems. The key to these systems is the recognition that a human expert requires a strong domain-specific knowledge base in order to achieve outstanding performance. This self-contained knowledge base is evident since a trainee, educated in problem solving methods, may still spend years of internship under the guidance of more senior level "experts". The transfer of knowledge is the crucial operation during the trainee's internship. The goal of a knowledge based system is to encode the complex and sometimes changing knowledge of the human expert into a system, and then use the knowledge as the expert would.

Knowledge based systems have advantages over conventional data processing systems because they utilize symbolic representtion, symbolic inference, and heuristic searches. Thus, instead of requiring a specific sequence of algorithms, the knowledge base system can, for example, include a set of rules (if-then conditions) which govern the system behavior. The advantages of a set of rules over algorithms are:

1) the information is in a more natural language

2) new information can be added with greater ease

3) the solution can be understood more readily since the specific rules used to come to a conclusion can be traced.

Knowledge based systems do not have to represent their knowledge in a rule form. Some knowledge is represented as a fact and some is represented by frame-like relationships. Regardless of the specific encoding form used for the knowledge. the knowledge systems can handle uncertain or unspecified information more readily than the conventional processing system. Therefore. even if the ultimate goal of a project may be to develop a specific algorithmic controller, a knowledge based system allows many flexibilities which help transform human knowledge into a computable form.

Even though research in knowledge based systems has been in progress for many years, the last five years have seen the emergence of the systems from the research lab into the world. These systems, seen in the medical, scientific, management and engineering fields, are being used to solve many types of problems: interpretation, prediction, diagnostics, design, planning, monitoring, debugging, repair, instruction and control.[10] Many hardware and software tools are being developed for knowledge based systems, but these tools are not prerequisites for development of a knowledge based systems.

The tools which have been developed include computer languages, canned programs for inferring from a knowledge base, and computing machinery which operates the software languages and programs more efficiently. The use of LISP or PROLOG rather than BASIC, FORTRAN or PASCAL in knowledge based systems is common. LISP and PROLOG handle symbolic representation and manipulation more easily than the more conventional languages. The canned programs for inferring from knowledge, referred to as inference engines, are knowledge independent. Each program has a unique way of sorting and responding to the knowledge it is given. The user must find the program which provides the robustness and friendliness he requires. A few of the existing software tools and one of the many applications each has been used for is mentioned:

1) EMYCIN - used in diagnostic systems for internal medicine

2) OPS - used in design and verification systems for VLSI circuits

3) HEARSAY III - used in spill management systems

4) KEE - used in chemical processing controls

5) ART - used in flight planning systems.

This is meant as an example only. It is not implied that EMYCIN is the best at diagnostics, or OPS at design. etc.

The design of a knowledge based system is an evolutionary process. The listing of specific steps for the design of a system is somewhat ambiguous in that often steps overlap or can be resequenced. Basically the designer must:

1) Identify the characteristics of the problem to be solved

2) Identify the sources of knowledge to be used

3) Identify a method for representing the knowledge

4) Determine the inference engine necessary

5) Select or develop software and hardware tools

6) Encode the systems knowledge base

7) Evaluate the knowledge base and inference engines performance

8) Edit the knowledge base until system performance is satisfactory.

In identifying the problem characteristic we determine the limits of the knowledge domain. The type of solution (diagnosis, prediction, control, etc.) required is specified. The constraints on equipment, signals, or timing for the system should be identified early.

The sources of knowledge include written materials, field observations, and most importantly human experts. Since the knowledge being encoded is often based on empirical experience rather than proven theories, a scheme for resolving any conflicting knowledge should be determined.

Many knowledge bases utilize rules to represent knowledge; however some knowledge is not easily stated in an if-then conditional statement. Knowledge representation in the form of object (fact) programming and/or frame (relational) systems must also be considered.

ORIGINAL PAGE IS
OF POOR QUALITY

Inference engines may be developed or existing ones acquired. There are many strategies for how situational data should be compared to a knowledge base in order to solve a problem. Decisions such as: will our system begin with a desired goal and find any or all solutions which can obtain the goal (referred to as backward chaining): or will our system begin with a set of data and arrive at the conclusion or goals indicated by the data (forward chaining), must be made. The inference engine must be able to resolve conflicts created when there is insufficient information or two or more contradicting actions or possibilities indicated.

The selection of existing software and hardware tools should be made whenever possible. If money or system specifications do not allow the selection of existing software tools, a great deal of time may be required for developing new tools.

Knowledge acquistion, encoding the knowledge into the systems knowledge base may require an engineer trained in knowledge representation (again an expert in his domain). This involves taking the text or verbal descriptions of pertinent information and encoding it into a working-nonconflicting knowledge base.

The evaluation of the system begins at the subexpert levels so that system accuracy is easily checked. Then the performance of the system at the expert level should be compared with human expert performance. The knowledge based system does not need to outperform a human expert to be a useful tool. The system can be a time-saver and non-tiring aid to the experts. The system may also be a significant aid for training sub-expert level workers.

Decision Criteria For Diagnostic Application

We have studied and tested certain of the required steps previously described for designing a knowledge based system. The purpose of our system is the diagnosis of feeder insulation degeneration and incipient faults. We have major equipment constraints in that economics dictate that the computing equipment costs per feeder be kept at a minimum. This implies that our final system will be a microprocessing system which may not support many of the existing knowledge base tools. We decided to begin the initial investigations of a knowledge based system utilizing existing tools. Once a system with a good knowledge data base is developed, we will begin looking at the steps required to develop the system on a microprocessing system.

Our knowledge source has been from the researchers at Texas A&M and the Public Service Company of New Mexico. An example of the kind of expertise we are encoding is given below.

The human expert diagnosing the health of a particular feeder goes through the following steps:

1. Based on a recorded and mental data base of a feeder under normal, healthy conditions, he must develop impressions of what the feeder is supposed to look like.

2. He must continually calculate and receive data from the feeder to provide a current picture of the electrical condition of the feeder.

3. Comparison of current feeder data to the data stored when the feeder was known to be healthy (or at least healthier) must be made.

4. The expert begins to recognize a changing pattern more than a threshold detection in determining that feeder maintenance is needed.

5. After maintenance is performed comparison of the feeder data to the waveforms of the feeder data before degeneration of performance occured is made. If the current pattern of data more closely resembles the pattern of a healthy feeder, it is concluded and learned that the maintenance and repair operation was at least partially successful.

These steps which human experts perform are not easily encoded in conventional data processing because specific algorithms and numerical threshold values must be determined. These algorithms may not easily track time variations and other patterns which the human expert easily tracks. The knowledge based system does not need algorithms and it can introduce its own threshold values. The system knowledge base is composed of the same pattern detection ideas which the human expert has. For example:

The frequency and amplitude of certain noise levels can be monitored and recorded. This data can be compared with levels when the feeder was relatively healthy. Trends of change rather than absolute difference may be monitored.

Based on the previous example it can be seen why we selected an inference engine which has forward and backward chaining capabilities. Given a set of data we forward chain to the conclusion as to whether the feeder appears normal or not. If the feeder does not appear normal, based on present data and recorded data, we forward chain to probable causes for the abnormalities. Once we have a guess of probable cause for system degeneration, we backward chain to see if we can find the data necessary to support the diagnosis.

Based on these requirements we are investigating the use of the OPS and ART software tools, both of which can operate on a VAX system or a symbolic processor. These tools have helped us to test the collection and organization of the knowledge the experts have provided. Eventually, we can drop some of the features provided by these tools and develop a streamlined inference engine and knowledge base for our problem.

Justification For Knowledge Based Systems

It is obvious from a review of Table 1 functions that conventional protection and supervisory equipment cannot implement all of these operations. Furthermore, the specific functions of incipient fault detection and equipment failure diagnosis cannot be performed by existing commercial equipment when one considers that these would be very important functions and would allow for the very early detection of faults and the scheduling of maintenance, the importance of advanced power substation automation concepts including knowledge based systems becomes apparent.

One could argue that existing substation apparatus and designs are "adequate". However, to take this position is to ignore advances in technology which offer us many advantages and improvements in operation and control. Instead of simply reacting to catastrophic faults as they occur, it is easily recognized that a system which can detect incipient conditions prior to catastrophic failure has definite value by incorporating this and other nontraditional functions from Table 1 in integrated. computer based hardware we can expect significant improvements in our ability to control and operate the power substation. In essence, it should be recognized that when technological advances occur providing us new tools we should reevaluate our system designs and functions and, where possible, incorporate this new technology. In the case of integrated common knowledge based systems it should be expected that we can reduce the hardware complexity in substation while at the same time providing more powerful diagnostics resulting in increased reliability and a reduction in catastrophic failures.

It is with the above objective that research in this area will be expanded. The next steps are to investigate specific expert system tools which can be used with integrated system architecture to best implement the automation functions shown in Table 1.

Summary

It has been shown that certain functions of substation automation can best be performed by knowledge based systems. The increased use of integrated designs for substation automation provides the possibility for incorporating knowledge based systems approaches resulting in better, more "fine tuned" decision making.

Two field experiments have shown that the use of historical data and inferences can improve the detection of incipient faults and can diagnose certain progressive degeneration conditions such as insulator deterioration. By transferring the approach of a human expert to a real time computer, it is possible to significantly improve substation protection and diagnostic functions.

Considerable work remains to optimize the use of knowledge based system approaches in small, microprocessor based equipments. Successful implementation of these approaches will dramatically change the design of substation automation systems.

REFERENCES

[1] "Utilities Survey of Automation (SCADA-RTU) Functions" Application of New Technologies Working Group, PES Substations Committee, W. Block, Chairman

[2] "Recent Field Experiences With the Automation of Substation Control" Automatic and Supervisor Systems Subcommittee. IEEE/PES Paper A 78 102-6

[3] "Microprocessor Applications of Interest to Substation and Protection Engineers" Joint Committee Report, Automatic and Supervisor Systems Subcommittee, Substations Committee and Substation Computer W.G. of PSRC, IEEE/PES Paper A 77 746-1

[4] "Feeder Protection and Monitoring System, Part 1: Design, Implementation and Testing", M. Aucoin, J. Zeigler, B. Don Russell, IEEE Transactions on Power Apparatus and Systems, Vol PAS-105, No 4, pp 873-80

[5] "Feeder Protection and Monitoring System, Part 2: Staged Fault Testing Demonstration", M. Aucoin, J. Zeigler, B. Don Russell, Transaction on Power Apparatus and Systems, Vol PAS-104, No 6, pp 1456-1462

[6] Information From Protective Relays and Circuit Breakers" C. Fukui, J. Kawakami, Transactions On Power Apparatus and Systems, Paper 86 W M 112-7

[7] "A Microprocessor-Based Technique For Detection of High Impedence", S. Balser, K. Klements, D. Lawrence, IEEE Transactions On Power Apparatus and Systems, Paper 86 W M 155-6

[8] "A Comparison of Measured High Impedence Fault Data to Digital Computer Modeling Results" R. Lee, M. Bishop. IEEE Transactions On Power Apparatus and Systems. Paper 85 W M 232-4

[9] "Distribution High Impedence Fault Detection Utilizing High Frequency Current Components", B. Aucoin, B. D. Russell, Transactions On Power Apparatus and Systems, Vol PAS-101, No 6, pp. 1596-1606

[10] "An Overview of Expert Systems" F. Hayesroth, D. Waterman, D. Lenat, Building Expert Systems, Addison-Wesley Publishing Co., Inc. Don Mills, Ontario, 1983.

B. Don Russell (SM) received B.S. and M.E. degrees in Electrical Engineering at Texas A&M University. He holds a Ph.D. from the University of Oklahoma in power systems engineering.

Dr. Russell is an Associate Professor of Electrical Engineering and directs the activities of the Power System Automation Laboratory of the Electric Power Institute, Texas A&M University. His research centers on the use of advanced technologies to solve problems in power system control, protection, and monitoring. He is the inventor of several microcomputer based automation systems and is the recipient of several awards for advanced technology applications.

Dr. Russell is a member of the Substations Committee and Power Engineering Education Committee of PES. He is a registered professional engineer and a director of the Texas Society of Professional Engineers.

Karan Watson (M) received her B.S. and M.S. degrees in Electrical Engineering at Texas Tech University. She holds a Ph.D. from Texas Tech University in digital control systems.

Dr. Watson is an Assistant Professor of Electrical Engineering at Texas A&M University. Her research centers on knowledge base systems and the design of digital VLSI devices for enhancing knowledge base systems.

A DIGITAL SIGNAL PROCESSING ALGORITHM FOR DETECTING ARCING FAULTS ON POWER DISTRIBUTION FEEDERS

Dr. B.Don Russell
Senior Member

Ram P. Chinchali
Student Member

Texas A&M University
College-Station, Texas

ABSTRACT :- Signal processing hardware and software can be used to significantly improve the detection of certain power system faults using computer relays. Integrated systems and architectures for monitoring several fault sensitive parameters have been investigated. A suggested architecture is presented utilizing several processors.

Several fault sensitive parameters for the detection of arcing faults are presented. A detection methodology based on these parameters is described and a partial solution to the problem of directionality is discussed. The use of knowledge base environment to modify protection criteria is also suggested.

INTRODUCTION

Some downed distribution feeder faults exhibit a low magnitude of fault current and cannot be detected by conventional overcurrent protection. There is a desire in the utility industry to detect these faults for operational reasons and to improve public safety. Often, arcing is associated with these faults which poses a potential fire hazard and property damage. The arcing phenomenon exhibits a random burst nature resulting in a spectrally-rich current waveform at frequencies both above and below the fundamental power frequency.

Earlier work has indicated that no single frequency can be used as a parameter to identify the presence of these faults because transients from switching activity can sometimes exhibit a similar burst nature and characteristic frequencies [1,2]. In this paper, a vectorial approach to fault detection is presented wherein several parameters are observed simultaneously. By attaching certain confidence factors with each parameter, a probability estimate can be made as to the presence of these faults.

Presented at IEEE/PES 1988 Winter Meeting, New York, Paper No. 88 WM 123-2, 1988.

Modern day microcomputers in signal processing integrated circuits make this approach feasible. Inexpensive hardware configurations and system architectures make possible powerful computer relays capable of performing the numerous, complicated calculations required for a multi parameter fault detection algorithm. This allows researchers to address the single most significant practical problem with sensitive, high impedance fault detection devices, namely, the need for discrimination between fault vs. normal events on the protected feeder.

It is expected that the vectorial approach using multi parameter algorithms will eliminate practical limitations and bring the utility industry closer to a solution to this long standing problem.

SYSTEM ARCHITECTURE

The random/transient (burst) nature exhibited by the arcing phenomena dictates that, for effective detection, it is not only imperative to track variations in the signal pattern, but also equally important to identify the periodicity of burst duration on a subcycle basis during the fault itself. Each feeder may have a different normal noise level which itself may be dependent on and vary with the load. A typical distribution feeder exhibits a periodic load cycle.

The detector must be adaptive to these load variations and at the same time avoid a complicated procedure for calibrating "pickup" levels for fault detection. Hence, the need for a protective system that associates some form of "intelligence" is desirable. The intelligence can be bred into a knowledge-based system that can communicate/interact with a microprocessor based digital signal processor. A system architecture such as the one shown in Figure 1 has been adopted by researchers at Texas A&M. Input signals from current and voltage transformers are appropriately conditioned and low-pass filtered. A separate current signal is also derived by notching the fundamental power frequency and band-pass filtering it between 2-6 KHz. This signal is exclusively used for monitoring the high frequency content in the arc generated noise. The input signals are then digitized and processed by digital filters for extracting the various parameters. The TMS 32020 is a digital signal processing chip capable of filtering signals efficiently in the digital domain. The TMS 32020 interacts with a micro-processor based system in which the main software resides.

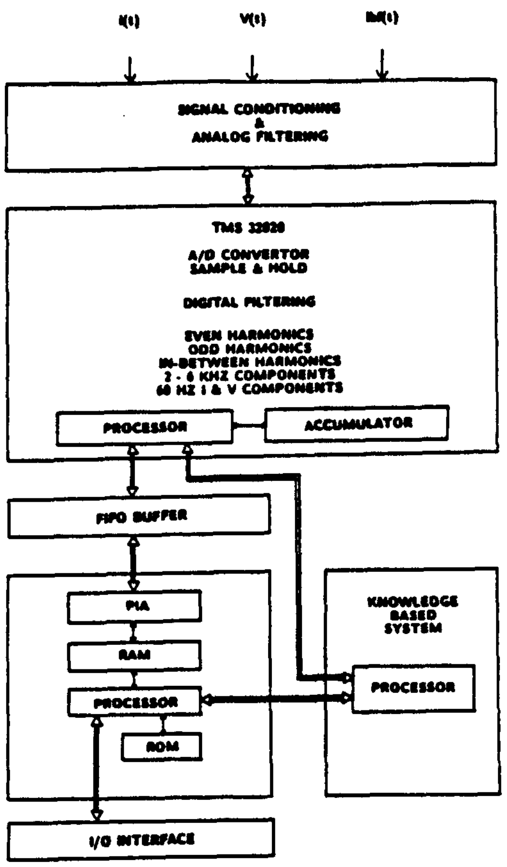


Figure 1. System Architecture

A First-In-First-Out (FIFO) buffer is interfaced between the TMS 32020 and the micro-processor in order to alleviate timing burden on the processor and thereby prevent the possibility of data overrun. Interface exists between the micro-processor and the Knowledge-Based System (KBS) and also between the KBS and TMS 32020.

The TMS 32020 processor's modified Harvard architecture emphasizes speed and increased throughput for real-time signal processing applications such as digital filtering. The device incorporates internal hardware for single-cycle 16 X 16 - bit multiplication, data shifting and accumulation. This hardware - intensive approach provides computing power for high speed time - domain convolution that other processors typically perform in software or microcode. Increased flexibility is also provided by two 256 word on-chip data RAM and an additional 128 K words of external memory.

The function of the microprocessor based system is to incorporate feeder monitoring, fault detection, and diagnostic capabilities using Eprom - resident software. Logic is provided to update thresholds dynamically and also to identify significant increase in the energy level of detection parameters. Other functional features such as updating software counters for time discriminatory decisions, look-up tables for weighting constants and probability estimates are also included.

The knowledge - based system provides an "expert" environment that combines algorithmic and heuristic approaches in narrowing the search space for problem identification and solution. The "expert" system utilizes three major building blocks to enhance decision making capabilities :

1. **Knowledge Base** that incorporates a data base of heuristics and facts used by an "expert".
2. **Working Memory** that provides facility for dynamic storage of facts asserted during program execution.
3. **Inference Engine** that processes knowledge from the data base and external facts to provide answers and derive new approaches.

DETECTION SCHEME

The parameters that are monitored by the detector comprise : 1) "in-between" harmonic frequencies, 2) even harmonic frequencies, 3) odd harmonic frequencies, 4) 2-6 KHz high band frequencies, 5) zero sequence component at fundamental frequency, 6) negative sequence component at fundamental frequency, and 7) positive sequence component at fundamental power frequency. These parameters are derived by filtering the input signals digitally in the signal processor.

For example, Figure 2 depicts the frequency response of a digital filter for extracting the "in-between" harmonic frequencies from the input signal. The filter length is 120 samples which translates into a filter transient response of one power frequency cycle at the given sampling rate. Figure 2 illustrates only a portion of the frequency response for the sake of clarity.The filter response is identical for other "in-between" frequencies extending upto the sampling frequency. Similar "Comb" type digital filters are also used to derive the even and odd harmonic frequencies. The digital filter used to derive the high frequency (2-6 KHz) components is of band-pass type.

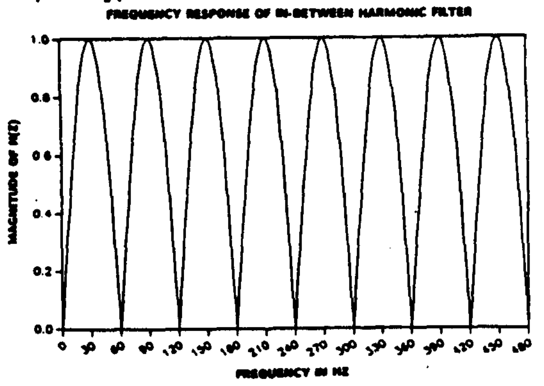


Figure 2. In-between Harmonic Digital Filter

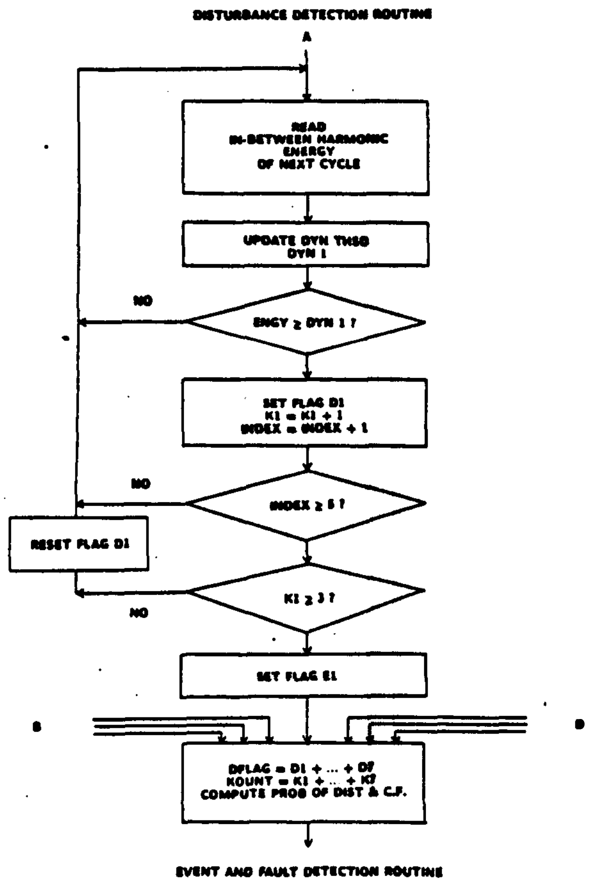
The detection algorithm does not consider individual samples in the filtered signals, but attaches importance to their cumulative effect over a predefined time. The technique utilizes the summation of the square of the filtered current data samples, referred to as the energy, over an entire cycle of the fundamental power frequency. This approach is preferred as it greatly alleviates the timing constraints on the micro-processor because now, instead of having to access every sample of filtered data, it can fetch one data value for each parameter that is monitored once every fundamental cycle. The energy calculation is carried out efficiently in an accumulator within the TMS 32020.

Detection of incipient, arcing faults is based on several characteristics of these faults some of which are stated below :

1. Arcing faults exhibit an increase in non-fundamental frequencies.
2. Arcing phenomena exhibit a random burst nature.
3. Arcing phenomena often persist over relatively long periods of time.
4. Transients from switching operations do not persist and are usually duration limited to a few cycles.

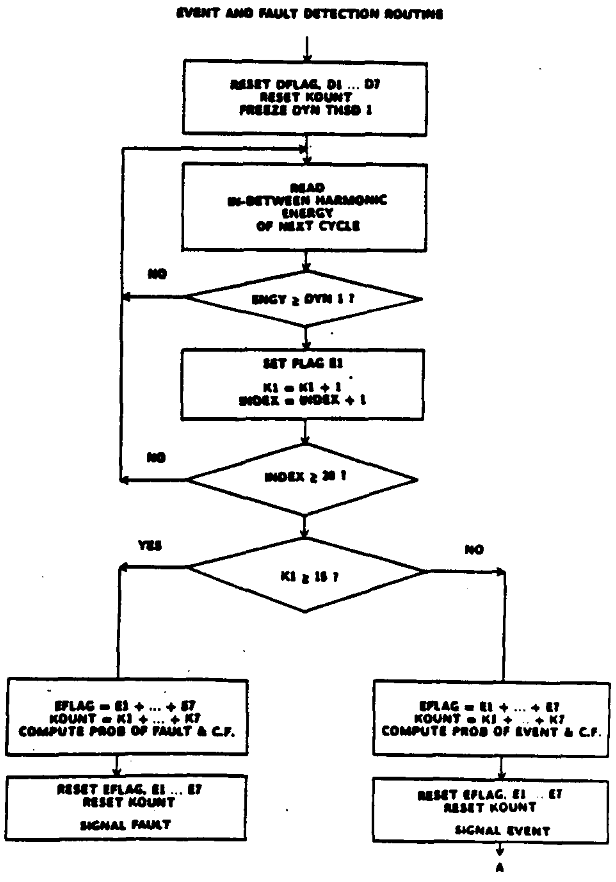
By using the above criteria in an "intelligent" environment, it is possible to detect incipient and other low current arcing faults with a degree of selectivity that can discriminate normal switching events. The key to effective detection is the identification of signal patterns that change from pre-fault to fault conditions. A hierarchical nature is bred into the algorithm by making use of three levels in the detection process before signalling a fault. The system starts by recognizing a "disturbance" and on occurance of one, the system devotes its attention to trying to verify if the disturbance qualifies as an "event". An event recognition is followed by an attempt to classify the episode into either a "fault" or a normal switching activity. The progression of detection scheme from one level to another is automatic, irrespective of whether all parameters register abnormality. Decision is finally based on the confidence levels and associated probability factors with which an episode qualifies as a disturbance, event or a fault.

The progression also implies use of time as a discriminatory factor. In a typical distribution environment, a strong adaptibility to periodic variations in the load is desirable for the detection scheme. The feature of adaptibility has been incorporated in the algorithm by making use of two thresholds - a static threshold and a dynamic threshold. The static threshold is used to adjust to changes in the load on a continual basis whereas the dynamic threshold is used to adjust to the feeder environment on a continuous basis. The dynamic threshold is defined as the weighted average energy per cycle calculated over a moving time window of predefined length wherein energy in present cycles are weighted more heavily compared to previous cycles. A typical weighting scheme would be one where the factors exhibit an exponential distribution. A flowchart is depicted in Figure 3 for the disturbance detection routine and in Figure 4 for the event and fault detection level.



DISTURBANCE DETECTION ROUTINE
A
READ IN-BETWEEN HARMONIC ENERGY OF NEXT CYCLE
UPDATE DYN THSD DYN 1
NO
ENGY ≥ DYN 1 ?
SET FLAG D1 K1 = K1 + 1 INDEX = INDEX + 1
NO
INDEX ≥ 5 ?
RESET FLAG D1
K1 ≥ 3 ?
NO
SET FLAG E1
B
D
DFLAG = D1 + ... + D7 KOUNT = K1 + ... + K7 COMPUTE PROB OF DIST & C.F.
EVENT AND FAULT DETECTION ROUTINE

Figure 3. Disturbance Detection Routine



EVENT AND FAULT DETECTION ROUTINE
RESET DFLAG, D1 ... D7 RESET KOUNT FREEZE DYN THSD 1
READ IN-BETWEEN HARMONIC ENERGY OF NEXT CYCLE
NO
ENGY ≥ DYN 1 ?
SET FLAG E1 K1 = K1 + 1 INDEX = INDEX + 1
NO
INDEX ≥ 20 ?
YES
NO
K1 ≥ 15 ?
EFLAG = E1 + ... + E7 KOUNT = K1 + ... + K7 COMPUTE PROB OF FAULT & C.F.
EFLAG = E1 + ... + E7 KOUNT = K1 + ... + K7 COMPUTE PROB OF EVENT & C.F.
RESET EFLAG, E1 ... E7 RESET KOUNT SIGNAL FAULT
RESET EFLAG, E1 ... E7 RESET KOUNT SIGNAL EVENT
A

Figure 4. Event and Fault Detection Routine

DATA ANALYSIS

Data analysis was performed to verify the efficacy of the various detection parameters. Test data from three different staged fault locations were chosen and processed by the algorithm. Figure 5 shows the time domain signal for an arcing fault conducted on wet soil conditions. The short duration nature of arcing bursts, as is typical on this type of soil can be clearly observed [2]. Figure 6 shows the variation of the "in-between" harmonic energy for this fault. As is evident, these frequencies are very sensitive to short duration bursts. Figure 7 shows the variation of even harmonics for the same fault. It is observed that the even harmonic frequencies are also very sensitive to short arcing bursts. The odd harmonic frequencies however did not register such changes. This was due to the level of odd harmonics being relatively high even under normal conditions. Other parameters such as 2-6 KHz components, negative sequence component, etc. were not found to register significant dynamic changes. Figure 8 shows the current waveform during an arcing fault that was conducted on dry soil. On dry soil, the arcing bursts are of relatively long duration. For a fault of this type, it was observed that the even and odd harmonic frequencies register a large dynamic change as can be seen from Figure 9 which depicts the variation of odd harmonic energy. Also, the negative sequence component indicated a considerable increase in magnitude as illustrated in Figure 10.

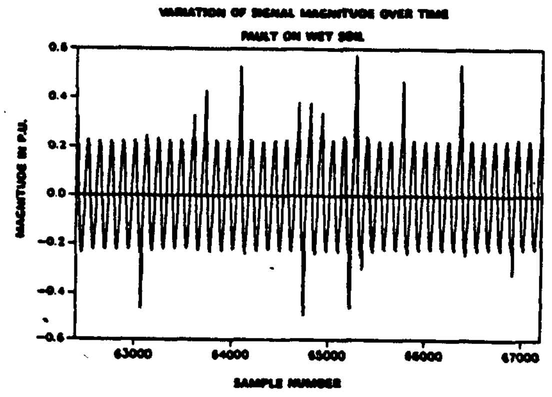


Figure 5. Arcing Fault on Wet Soil

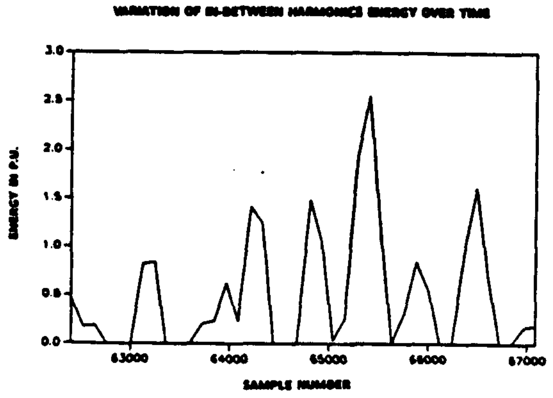


Figure 6. Variation of In-between Harmonic Energy

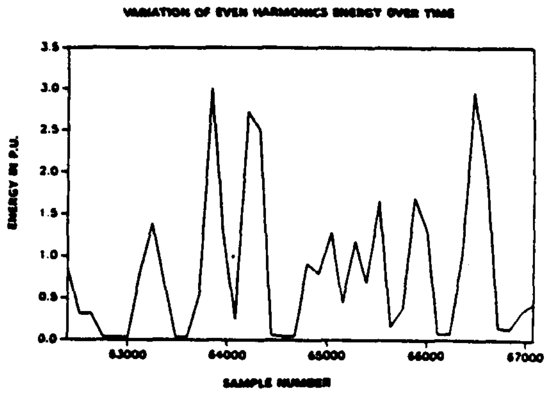


Figure 7. Variation of Even Harmonic Energy

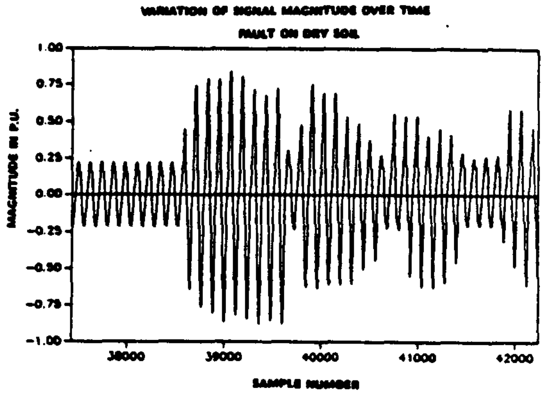


Figure 8. Arcing Fault on Dry Soil

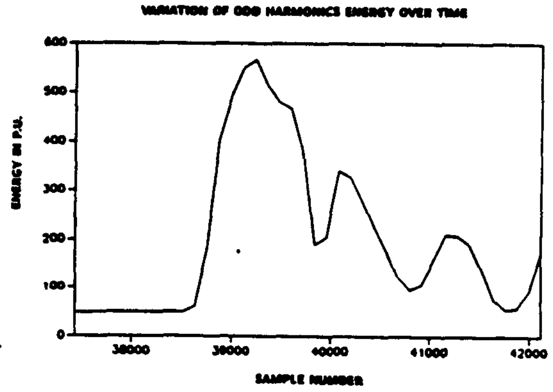


Figure 9. Variation of Odd Harmonic Energy

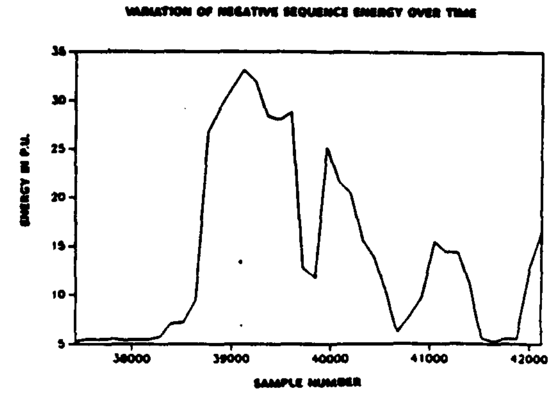


Figure 10. Variation of Negative Sequence Component

Similar analysis was carried out for faults conducted on sandy soil. Here, the burst duration is considerably long as is depicted in the time domain signal of the 2-6 KHz components in Figure 11. The dynamic increase in energy of these broad band frequencies is depicted in Figure 12.

It is evident from the above analysis that the random burst characteristics of arcing faults together with their dependency on soil type dictates that no single parameter is equally sensitive to detect arcing faults in a secure manner. Another factor compounding the complexity of detection arises from the necessity to discriminate switching transients from arcing bursts. Earlier work has indicated that switching activity such as capacitor bank and load tap changer operations cause considerable increase in the level of odd harmonics and several even harmonic frequencies [2]. Capacitor banks also attenuate high frequency components by providing a low impedance shunt path to ground. The inrush current during the energization of transformers causes an increase in the level of harmonic frequencies. In view of these constraints it is concluded that, for effective fault detection, simultaneous monitoring of several parameters is justified.

DIRECTIONALITY CRITERIA

The direction to the fault is many times needed to prevent a source side relay/detector from tripping for a line side fault. A technique based on power flow direction, similar to the operation of voltage restrained directional overcurrent relays has been investigated [3]. This method involves determining the coefficients of the fundamental frequency current and voltage components from the discrete fourier transform of the sampled data in a recursive manner [4]. The coefficients are then used to calculate a directionality factor in the direction of "maximum torque". By comparing with a predefined threshold level, a fault is considered to be in the direction seen by the relay if the directionality factor lies in the positive torque region.

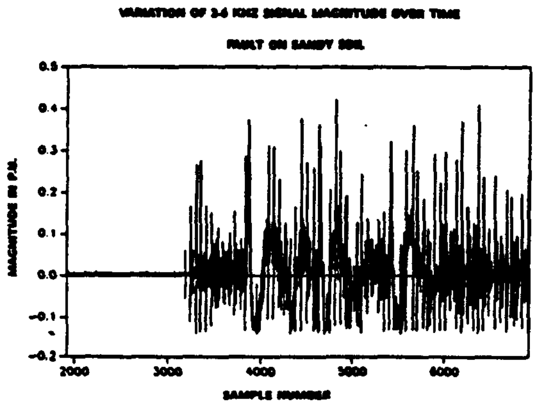


Figure 11. Arcing Fault on Sandy Soil

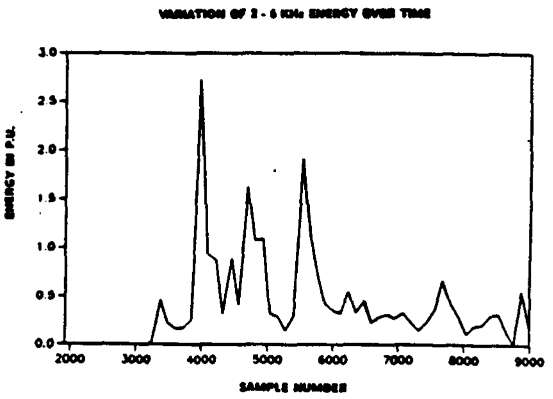


Figure 12. Variation of 2-6 KHz Energy

Assuming that the input current and voltage waveforms are sampled N times per cycle of the fundamental frequency to produce sample sets x_{ik} and x_{vk}, the discrete fourier transform of x_k contains a filtered fundamental frequency component given by,

$$\begin{aligned} X_1 &= \frac{2}{N}\sum_{k=0}^{N-1} x_k e^{-j\frac{2\pi}{N}k} \\ &= \frac{2}{N}\sum_{k=0}^{N-1} x_k \cos\frac{2\pi}{N}k - j\frac{2}{N}\sum_{k=0}^{N-1} x_k \sin\frac{2\pi}{N}k \\ &= A_1 - jB_1 \end{aligned}$$

where A_1 and B_1 are the cosine and sine multiplied sums in the expression for X_1. The magnitude and phase of the fundamental component can be calculated with respect to the reference waveforms $\cos(\frac{2\pi}{N}k)$ and $\sin(\frac{2\pi}{N}k)$.

$$C_1 = \sqrt{A_1^2 + B_1^2} \qquad \phi_1 = \tan^{-1}(\frac{B_1}{A_1})$$

The torque relation for a current-voltage directional relay can be expressed as,

$$T = K_1 VI\cos(\theta - \tau) \tag{1}$$

This operating characteristic is seen to be a straight line offset from the origin and perpendicular to the maximum positive torque position of the current as illustrated in Figure 13. The voltage and current phasors are computed from the coefficients C_{1v} and C_{1i} and the directionality factor T determined in the direction of maximum torque angle τ using (1), for each pair of current and voltage data samples over a system cycle.

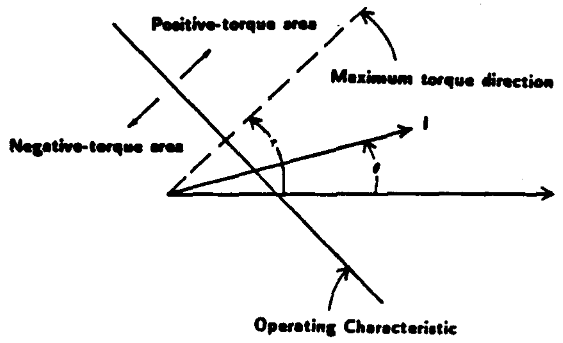


Figure 13. Operating Characteristic of Voltage-Current Relay

Figure 14 shows the variation of instantaneous power as seen by a down stream relay for an arcing fault in its direction. Figure 15 shows the variation of the directionality factor as seen by this relay. It is evident that an increase in level of T occurs for a fault in the direction seen by the relay. On the contrary, a relay looking in the reverse direction will register a decrease in level of T for the same fault. Figure 16 depicts the power flow for a different arcing fault and Figure 17 the corresponding variation in the directionality factor. The normal level of T is monitored and compared with predetermined pickup thresholds to determine the direction when a fault is registered by one of the parameters. Time coordination is then utilized to block a source side device from tripping out before a load side device picks up a fault.

Current research is centered on improving the sensitivity of this approach by "zero-based" referencing of the fault current without the load component included in the calculation. This would allow sensitive directionality evaluation of even very low level faults.

In order to identify the faulted conductor phase, a different approach can be used. It is observed that the 2-6 KHz signal indicates burst phenomena only during certain periods of a power cycle. There are typically two such bursts occuring every half cycle as shown in Figure 18 where the faulted phase voltage was superimposed on the high frequency signal to illustrate this phenomenon.

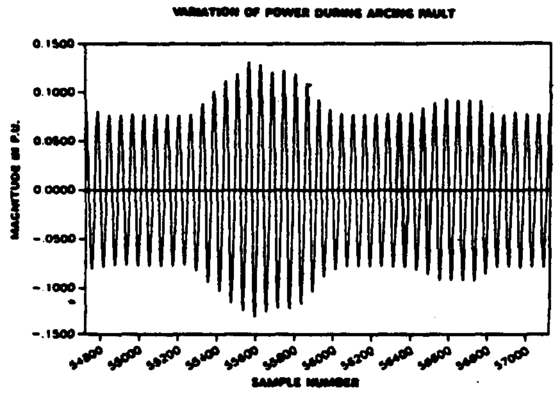


Figure 14. Variation of Power seen by Forward Relay

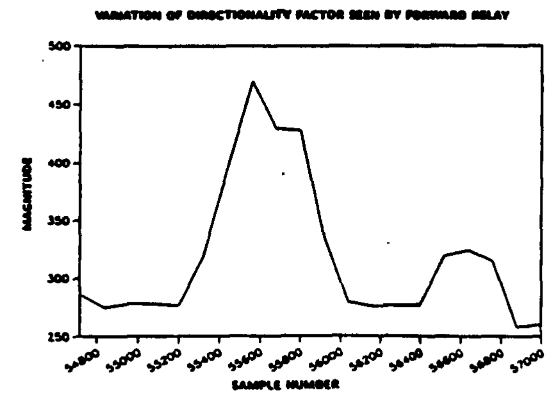


Figure 15. Variation of Directionality Factor seen by Forward Relay

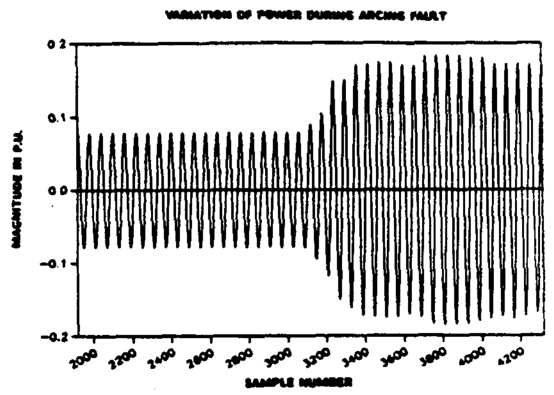


Figure 16. Variation of Power seen by Forward Relay

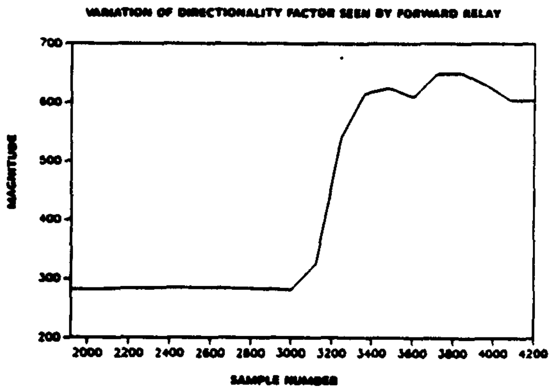


Figure 17. Variation of Directionality Factor seen by Forward Relay

The phase voltage waveform is taken as reference as it does not register any perturbations during arcing fault conditions. These bursts occur when the system voltage equals the restrike voltage. The arc voltage then drops to a constant e_{arc} and current begins to flow. Current reaches a maximum when the system voltage equals the arc voltage. After this time the current decreases till the arc restrikes again [5]. An unfaulted phase, on the other hand, will not register such burst activity in the high frequency components. It is thus anticipated that by monitoring the variations in the 2-6 KHz signal on a sub-cycle basis and latching these bursts with certain positions on the voltage cycle, the faulted phase can be identified.

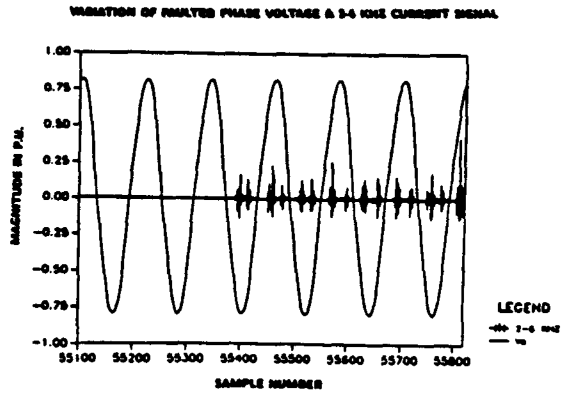


Figure 18. Relationship between Faulted Phase Voltage and 2-6 KHz Signal

KNOWLEDGE BASE ENVIRONMENT

The knowledge base interface is to provide rule-based "intelligence" environment to the signal processor. Numerous functions can be incorporated that would result in a secure and sensitive protection system. Functionally, it could monitor the sensitivity of various detection parameters, adjusting their levels in case of over/under sensitivity to provide overriding features in decision making. Other capabilities such as periodic adjustment of various weighting factors, time discriminatory counters and decision logic can also be incorporated. On a continual basis, it can serve as a database for monitoring load trends, load characteristics, waveform patterns, etc. Other characteristics such as arc burst duration and burst repetition rate can be monitored on a sub-cycle basis to provide discriminatory decisions such as faulted phase identification. Abnormal conditions can be compared with historical data bases using knowledge base algorithms to provide reliable decisions. Probabilities and confidence levels can be accurately determined by verifying trends in past history. Such an interface also facilitates additional support for numerous other protection applications such as conventional overcurrent, overvoltage, frequency variation, etc. Adaptive protection concepts which permit real time modification of relay settings, characteristics, or logic functions can be efficiently incorporated for improved system reliability and performance.

CONCLUSIONS

An intelligent computer relay for detecting arcing faults can be constructed using advanced signal processing hardware and software. The use of a vectorial approach monitoring numerous fault sensitive parameters allows for improved discrimination between low current faults and normal events on the protected feeder. By using a knowledge based system, it is possible to dynamically adjust protection weighting factors and improve protection performance within the relay itself.

Additional advantages of the proposed design include the ability to interface with higher level protection systems, the ability to provide detailed fault and feeder evaluation information, and the ability to implement sophisticated protection routines without hardware changes.

Continuing research at Texas A&M University has focussed on performance evaluation of this proposed approach and improvements to the individual protection parameters.

REFERENCES

[1] Aucoin,M. ,Russell,B.D., "Detection of Distribution High Impedance Faults Using Burst Noise Signals near 60 Hz", IEEE Transactions on Power Delivery, 86 T&D 546-6.

[2] Russell,B.D., Ram Chinchali, Kim, C.J., "Behaviour of Low Frequency Spectra During Arcing Fault and Switching Events", Presented at IEEE/PES 1987 Summer Meeting, San Francisco, California, 87 SM 633-1.

[3] Singh, J., Sachdev M.S., Fleming R.J., Krause, A.E., " Digital IDMT Directional Overcurrent Relays", IEE Conference Proceedings on Developments in Power System Protection, IEE Publication No. 185, pp 84-87.

[4] Phadke,A.G., Thorp,J.S. , Adamiak,M.G., "A New Measurement Technique for Tracking Voltage Phasors, Local System Frequency, and Rate of Change of Frequency", IEEE Transactions on Power Apparatus and Systems, vol.PAS-102, No.5, May 1983.

[5] Dunki-Jacobs,J.R., "The Effects of Arcing Ground Faults on Low Voltage System Design", IEEE Transactions on Industry Applications, vol. IA-8, No.3, May/June 1972.

Biography

B. Don Russell, (SM) received B.S. and M.E. degrees in Electrical Engineering at Texas A&M University. He holds a Ph.D. from the University of Oklahoma in power systems engineering.
Dr. Russell is Associate Professor of Electrical Engineering and directs the activities of the Power Systems Automation Laboratory of the Electric Power Institute, Texas A&M University. His research centers on the use of advanced technologies to solve problems in power system control, protection, and monitoring. He is the recipient of several awards for advanced technology applications. Dr. Russell chairs several working groups and subcommittees of PES.
Dr. Russell is a member of the Substation Committee and Power Engineering Education Committees of PES. He is a registered professional engineer and a director of the Texas Society of Professional Engineers.

Ram Chinchali, (S'84) received B.E. (Hons) degree from the University of Madras, India in 1981 and a M.E. in Electrical Engineering from Texas A&M University in 1984. He is currently working toward his Ph.D. degree.
Mr. Chinchali's research interests include microprocessor and knowledge based system applications in power system control and protection. During the brief period between 1981 and 1983, he was a Relay Engineer at English Electric Co., India where he developed protection schemes for 220 KV and 400 KV substations.

EXPERT SYSTEM STRUCTURES FOR FAULT DETECTION IN SPACEBORNE POWER SYSTEMS

Dr. Karan Watson Dr. B. Don Russell Ms. Irene Hackler
Texas A&M University NASA-JSC

ABSTRACT

This paper presents an architecture for an expert system structure suitable for use with power system fault detection algorithms. The system described is not for the purpose of reacting to faults which have occurred, but rather for the purpose of performing on-line diagnostics and parameter evaluation to determine potential or incipient fault conditions. The system is also designed to detect high impedance or arcing faults which cannot be detected by conventional protection devices.

This system is part of an overall monitoring computer hierarchy which would provide a full evaluation of the status of the power system and react to both incipient and catastrophic faults. An approximate hardware structure is suggested and software requirements are discussed. Modifications to CLIPS software, to capitalize on features offered by expert systems, are presented. It is suggested that such a system would have significant advantages over existing protection philosophy.

INTRODUCTION

This paper describes a software shell and an architecture for an expert system which provides a more comprehensive tool for the diagnosis of the health of a power distribution system. The intent of this diagnosis and protection expert system (DAPES) is to use the sensor data from the power system for on-line diagnostics and to provide detection of and protection from faults whether they create overcurrents or not. Some of the functions which DAPES includes are:

* detection and response to overcurrent faults on a feeder
* determination of directionality of currents in the event of a fault
* compression of data for more efficient communication and more effective on-line presentation to real-time operators
* distribution of monitoring and diagnostic functions to remote computational units
* detection and response to high impedance and arcing faults which do not cause overcurrents
* detection of incipient faults for the provision of better maintenance scheduling
* self-checking, or test on request, abilities within the detection system
* fault tolerance so that DAPES functions can be shed, down to overcurrent detection, in the event of delayed maintenance on a degraded system
* the flexibility to add a wide variety of protection functions with little to no hardware changes
* a tool which can not only perform the required protection features in the field, but can concurrently be used to investigate new and developing algorithms.

DAPES should reside in computing facilities as close to the loads and/or protection mechanisms as feasible.

The monitoring of power delivery systems can be described as a hierarchy of automation systems. The top level is concerned with broad system monitoring functions and interfaces with the operators. This level is typified by current Supervisory Control and Data Acquisition systems and is generally housed in a powerful mini-computer in a clean and controlled environment. The middle level of the hierarchy is often located at the same site as the SCADA but is dedicated to special functions. These functions are often implemented on small mini or even micro-computers. Functions such as energy management, post fault location and recovery systems, data acquisition and data communications are some examples of the distributed computing tools found at this level of the monitoring system. The low end of the hierarchy tends to be based in machinery which is located remotely from the rest of the hierarchy. This level performs the initial data acquisition of sensor data and is the only level where appropriate responsiveness for protection functions can be guaranteed. We attempt, with DAPES,

Presented at the Intersociety Engineering Conference on Energy Conversion, Denver, August, 1988.

to provide as much computational "intelligence" as possible in the remote low-level machinery in order to relieve the processing requirements in the top and middle levels of the hierarchy. Developments in available, and inexpensive, microprocessors makes the enhanced intelligence of remote units feasible.

Care must be taken when adding features and functions to DAPES in order to assure adequate responsiveness is maintained. Previously reported research supports the conclusion that expert systems provide beneficial structures for the required functions (1). For example, in the detection of arcing faults, certain, but varying, patterns of energy levels in the non-fundamental power frequency can indicate a fault. Figure 1 shows an oscillographic recording of a faulted distribution feeder. The upper trace represents the 60Hz phase current waveform carrying both the load and fault current. The lower trace represents the waveform with the 60 Hz fundamental frequency eliminated. Similar patterns have also been found in 20KHz systems. Human experts can view the information provided and, after a time period of a few seconds to a few minutes, can determine whether a low grade fault exists. In terrestrial systems different feeders, load conditions, weather conditions, soil types and faults present various patterns at different frequencies. Many algorithms are being studied to find a concise method of detecting non-overcurrent faults (2). Our expert system allows for the combined use of various algorithms with expert consideration of the current relative value of each algorithm. The environment strongly supports the introduction or deletion of algorithms into the detection process.

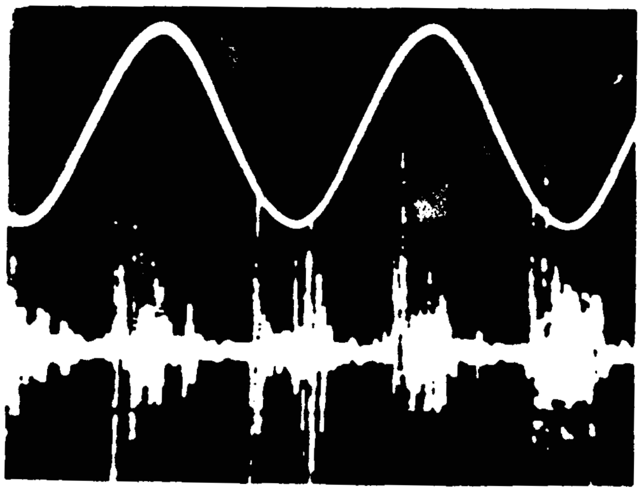


Figure 1. Arcing Fault Waveforms

A primary consideration for DAPES success is the provision of an expert system shell which provides real-time responsiveness for the on-line system monitoring. Typical expert systems will slow down drastically in the face of a rapidly changing data base as found when monitoring the power system sensor data. Therefore, we developed an expert system shell tuned for responsiveness. This shell is called PMCLIPS.

PMCLIPS: The Expert System Shell

The search for a real-time expert system lead us to CLIPS (3). CLIPS was developed at NASA and is based on the C language. The compiled C language provides a responsiveness in processing which is more rapid than that is typically achieved by an interpreted language, such as LISP. However, even CLIPS slows down drastically when faced with a rapidly changing data base. The reason for this can be traced back to the assumptions made when the reasoning process, or inference engine, for CLIPS, and most of the available expert system shells, were developed. Within the reasoning process a matching algorithm tries to unite the incoming data with a knowledge base of rules. Most matching algorithms assume the incoming list of data stays fairly constant from one reasoning cycle to the next. Also, most matching algorithms tend to be serial in nature so the use of parallelism for enhanced responsiveness is not supported.

We modified CLIPS so it could operate in a parallelized, monitoring environment, and the result is our PMCLIPS expert system shell. PMCLIPS has a modified matching algorithm (PMA) which can run on a single or multi processing environments.

PMA: A Parallelized Match Algorithm

PMA was influenced by the Rete match algorithm (4) found in CLIPS. Some of the internal data holding structures of the Rete and PMA algorithms are very similar. However, the flow of data through these structures differs greatly. The general flow of the inference process involving PMA is shown in Figure 2 and explained now.

PMA receives two inputs: a list of objects and sets of patterns. The objects represent the power system sensor data and the current diagnostic hypothesis and goals. The sets of patterns represent the conditions (if part of if-then rules) used for diagnostics in the power system. PMA searches for a match between objects and patterns. Initially the list of objects are compiled into a linked list of facts. Then the sets of patterns are formatted into two nets which aid in the efficient search for matches. These nets must be formed only once for a given set of patterns and goals.

The first net formed is referred to as the opnet, or condition net. In this net, usually beginning with the primary goal, each condition which will be checked in the diagnostic process is compiled into an indexing structure. In this net conditions are indexed with no regard to the combined patterns which must be checked. However, the condition node points to a particular point in the second net which maintains the link of conditions for a particular pattern or rule. The purpose of the opnet is to allow all matches between all particular conditions to be considered. A given condition can have zero or more matches. For example, the energy in high frequency information might increase above the recent average by 25% in more than one feeder. The match of increased high frequency energy must be matched with all the appropriate feeders, however, the condition of increased high frequency noise need only be listed in the opnet once. The opnet is structured so depth first searches may be attempted. Each object begins at the head of the opnet and when a mismatch is found a redirection in the search is made to effectively cut off branches which are unfruitful.

The second net formed is called the join net. This net represents the conditional patterns of each rule in the knowledge base in a linked format. Figure 3 illustrates a join net of three rules, two with four conditional clauses and one with three conditional clauses. Each condition, which can occur in more than

ORIGINAL PAGE
BLACK AND WHITE PHOTOGRAPH

ORIGINAL PAGE IS
OF POOR QUALITY

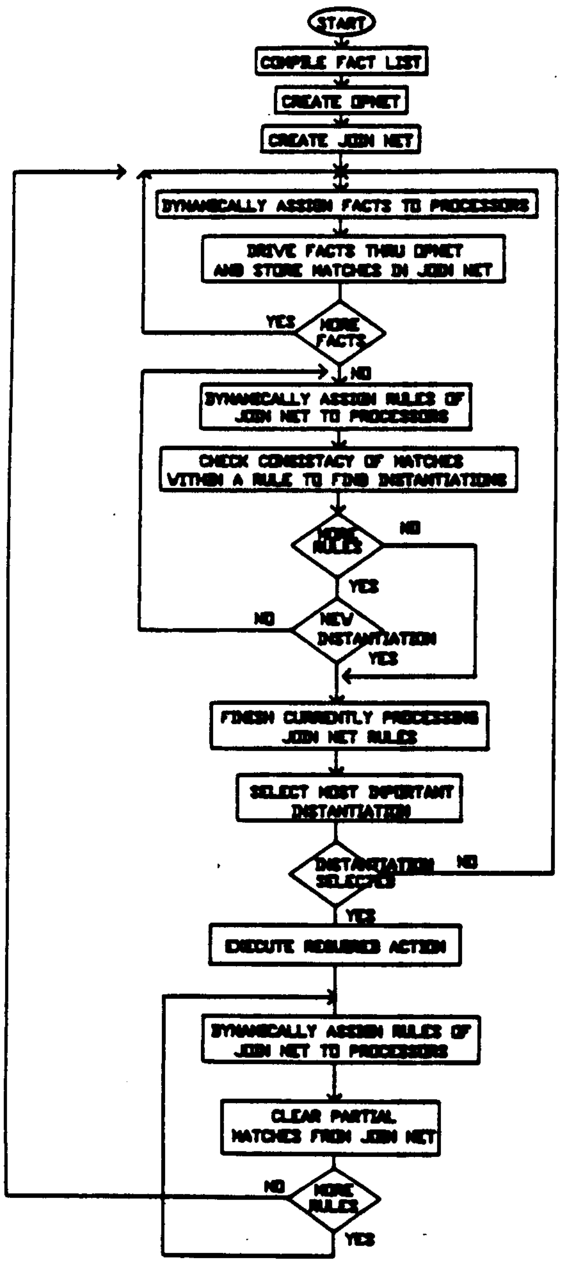

START
COMPILE FACT LIST
CREATE OPNET
CREATE JOIN NET
DYNAMICALLY ASSIGN FACTS TO PROCESSORS
DRIVE FACTS THRU OPNET AND STORE MATCHES IN JOIN NET
MORE FACTS
YES
NO
DYNAMICALLY ASSIGN RULES OF JOIN NET TO PROCESSORS
CHECK CONSISTACY OF MATCHES WITHIN A RULE TO FIND INSTANTIATIONS
MORE RULES
NO
YES
NEW INSTANTIATION
NO
YES
FINISH CURRENTLY PROCESSING JOIN NET RULES
SELECT MOST IMPORTANT INSTANTIATION
INSTANTIATION SELECTED
NO
YES
EXECUTE REQUIRED ACTION
DYNAMICALLY ASSIGN RULES OF JOIN NET TO PROCESSORS
CLEAR PARTIAL MATCHES FROM JOIN NET
MORE RULES
NO
YES

Figure 2. PMA Flowgraph

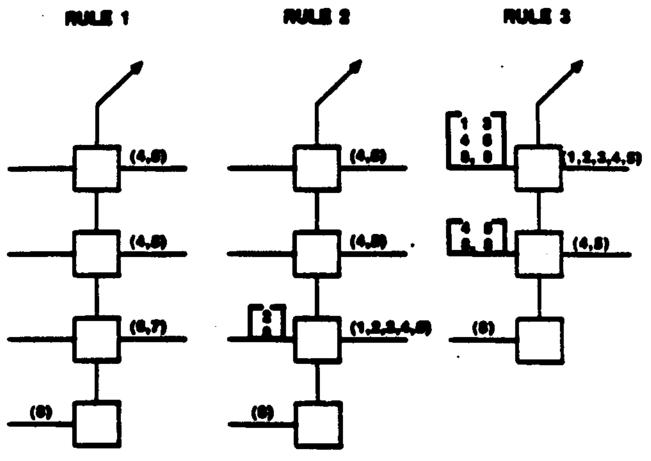


Figure 3. Example Join Net

one rule, is represented by the same point in the opnet. The join net is used to temporarily store all the matches of objects with conditions found in the opnet. In addition the join net is used to check the consistency of matches within a rule in determining if all conditions have been met. That is, if a rule for detection contains clauses such as: If the high frequency energy in a feeder rises by more than 25% above average and if the increased energy in a feeder persists for more than five power cycles, then an arcing fault is suspected; we must assure we are discussing the same feeder before we believe all the conditions have been met. Associated with each node in the join net is a right hand side (RHS) and left hand side (LHS). The RHS holds information from the opnet concerning the match of facts with conditions. The LHS holds the set of variable bindings which are consistent up to that point. If consistency in binding information on the RHS and LHS of a node can be created, the combined consistent set is sent up to the next node. How this is done will be expounded further, however keep in mind that if a set of consistent bindings reaches the top node, then that rule has been instantiated.

How PMA uses the opnet and join net begins with the driving of all objects in the fact list through the opnet. The advantage of PMA is that it has been structured so objects can be driven through in parallel. This is done with dynamically scheduled processors. Each object is assigned to a free processor. When the object has been driven through the opnet and all matches stored in the join net the processor is free to receive a new object.

Once all the objects have been driven through the opnet, all the possible matches are stored in the RHS of the join net. Next the processors are dynamically allocated a rule from the join net. The rules with the highest importance, salience, are allocated first. When a processor receives a rule to work on, it begins by looking at the bottom node, which corresponds to the first condition in a rule. If no variable bindings, conditions matched to facts, are found, the processor is finished with this rule and waits for a new rule. If any bindings do exist then one is extracted and moved up to the LHS of the next node. At this point consistency checking is performed on the variable bindings on the LHS and RHS of that node to see if a new consistent set of variable bindings may be constructed. If so, then they are shipped up to the next node for a similar operation to be performed. This process continues until the top node is reached, which means an instantiation has been found, or until no new set of consistent variable bindings can be generated for shipping up to the next node. If the top node cannot be reached with a consistent set of bindings, the control goes back to the bottom node so that the process can start over with a new set of variable bindings which are attached to the node. If no more variable bindings exist, the processor is finished working on that rule and returns to the main process to be reassigned to a new rule.

If an instantiation is discovered, then a special check is made to determine what to do next. PMA contains a structure which holds history information about the first n instantiations which were used to evaluate the list of objects. The value of n is

ORIGINAL PAGE IS
OF POOR QUALITY

variable and is set by the user to any desired value. If the newly discovered instantiation is identical to one of the instantiations in the history structure, then the new instantiation is ignored and not used. Processing continues until another instantiation is found which is new, or until all of the variable binding sets at the bottom node are exhausted. The history feature is useful in guarding against undesired looping errors during run time operation.

If the discovered instantiation has not been used during the last n cycles, then it is stored. At this point, the processor is finished working on the rule. Since only instantiations of lower importance could possibly be found in the rest of the sets of patterns, no more processors are assigned to process rules. This observation saves a lot of processing time in general.

The dynamic scheduling employed allows several rules to be checked for instantiations concurrently. One special consideration which must be insured is that whenever a processor is assigned to a rule it must not be interrupted until it finishes by finding a new instantiation or by running out of variable bindings to work with. This is important because the rules are fed out in descending order of importance. If a processor working on a rule with a relatively low level of importance finds a new instantiation, another processor still working on a set of patterns with a higher level of importance must be allowed to finish or else an instantiation of lower importance may be picked during the select step. It is assumed that the most important instantiation will be selected on each cycle, and in PMA it is.

At this point, PMA steps aside momentarily to allow the select operation to be performed. This job can be done very quickly since PMA sets up the data in a convenient form. The top nodes in the sets of patterns in the join net are checked in descending order of importance. The first set of patterns found which has an instantiation stored in it is the one chosen since it is the most important one. If no instantiation is found, the program is complete for this cycle.

After the appropriate rule has been selected we must perform the act step, (i.e. the then part of the if, then rule). The actions to be taken are dictated by the selected instantiation. Typically a series of assertions and retractions are made. In the Rete match algorithm, an assertion involves a long sequential series of steps which entails going through the opnet, join net, and agenda as far as possible. However, in PMA, an assertion involves simply the addition of a new object to the object list, which takes very little time.

In the Rete match algorithm, a retraction takes a lot of time since every pattern which matches the object being retracted must be located. Once these patterns are located, a linear search through the join net and agenda must be made so that all partial matches and instantiations which use the object may be deleted from the system. When several retractions are required per cycle, as in monitoring problems, much time is consumed. This is the place where most of the time savings of PMA come from. For retractions, PMA simply deletes the object from the object list. The price it pays in reasserting each object on each cycle is not as severe as the price paid by the Rete match algorithm in retracting large numbers of objects per cycle.

After the act step is finished, the final portion of PMA is executed. This is the third and final distinct parallel section of code which is run. The job that is performed is essentially a cleanup operation. The entire join net is cleared of all structures that were placed on it during the other two parallel execution phases. Clearing of structures refers to returning to dynamical scheduling since the time needed to clear one set of patterns may be very different than the time needed to clear another set of patterns. As in the second phase of parallel execution, free processors are assigned to sets of patterns. When a processor finishes with one set, it goes back to be assigned to a new set to clear. When all join net nodes are clear, control returns to the first phase of parallel execution in which all constant objects are driven through the opnet , and the cycle continues.

PMCLIPS: Test Results

Various sets of rules with varying numbers of conditions were run on CLIPS and PMCLIPS. For one typical test Figure 4 shows a plot of the ratio of CLIPS run time to the fastest PMCLIPS run time versus the average number of conditions in a rule set. The conclusion reached by examining this graph is that the more conditions present, the better PMCLIPS does relative to CLIPS. The improvement tends to level off at high values, but is still present.

Another graph is shown in Figure 5. This plot demonstrates the effects of adding more processors to a PMCLIPS run. In this plot the run time has been normalized to the PMCLIPS run with a single processor. Also shown is the CLIPS run time.

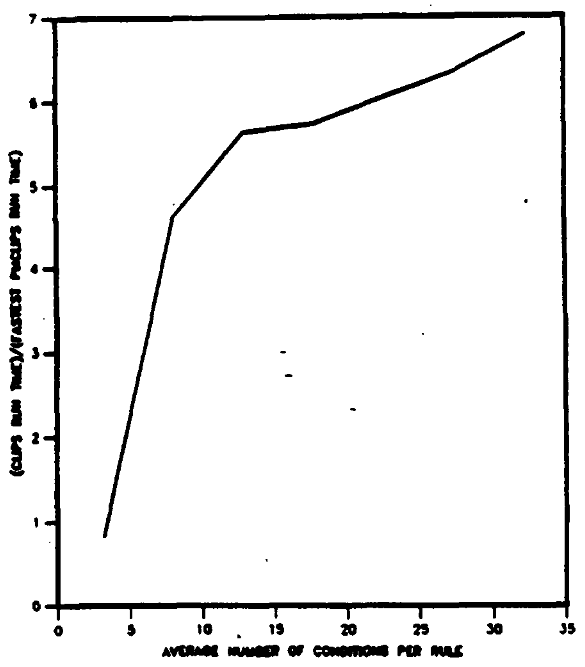


Figure 4. PMCLIPS Relative Performance
Versus Number of Conditions

ORIGINAL PAGE IS
OF POOR QUALITY

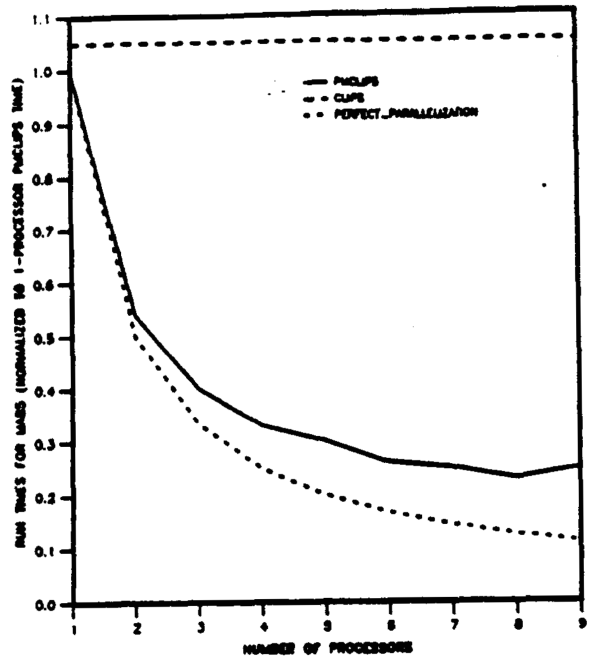


Figure 5. PMCLIPS Performance Versus Number of Processors

As the plot shows, adding a second processor cuts the run time almost in half. And adding a third processor significantly improves performance. However, after four or five processors have been added, the extra processors are almost not worth adding. This result is not surprising since the Sequent Balance 8000, which was used as the machine for test, has only a single bus. When several processors try to access memory, all but one must wait, which wastes time. This is a major problem with having just one bus.

Overall, the test results speak very well for PMCLIPS. Its design goal of improving CLIPS for rule sets with several conditions has been met. PMCLIPS obviously makes use of parallelism, whereas CLIPS does not.

The Computing Environment for DAPES

Figure 6 presents a hardware architecture to implement DAPES. Traditional voltage and current signals from a single feeder line are the inputs. Analog filters shown in this figure illustrate what is used for a 60Hz power system. The low pass filters clean up the 60Hz information, the 60Hz notch filter eliminates the fundamental frequency so harmonics can be seen, and the high pass is used to observe high frequency noise. Similar filtering techniques would be used for higher frequency power systems. This filter block can economically be provided for each line being monitored by this DAPES system.

The number of analog to digital converters can be reduced by using multiplexing. We believe the great deviation in signal levels and dynamic ranges for the different filtered signals suggest the four A/D converters should be the minimum. Multiplexing among different lines is acceptable if practical limitations deem it important. The signal processor shown achieves common parametric computations necessary for some of the non-overcurrent faults. In a non-restricted environment, the entire signal conditioning block would be repeated for every line being diagnosed.

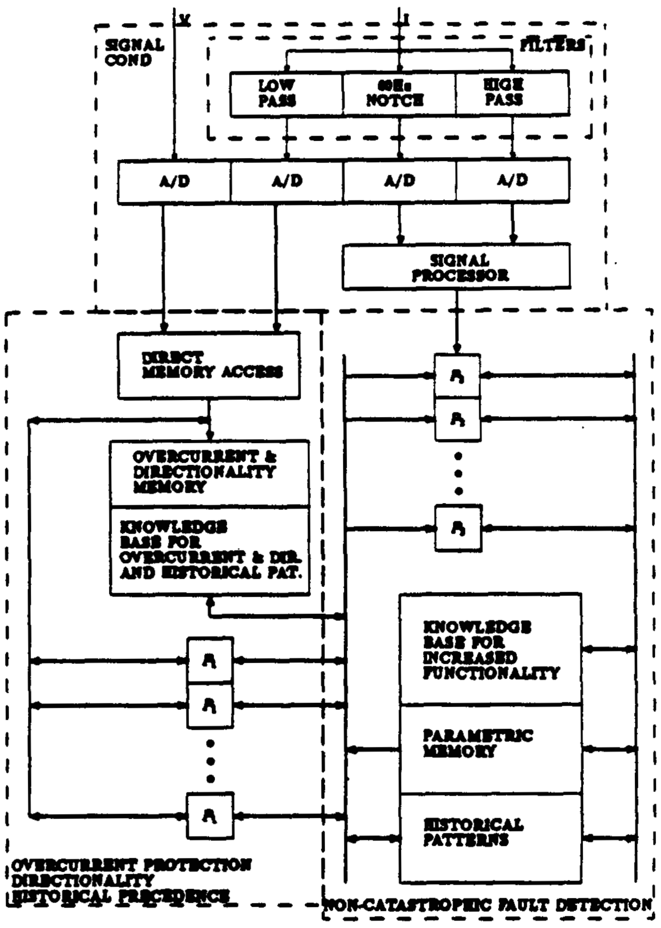

V
I
SIGNAL COND
FILTERS
LOW PASS
60Hz NOTCH
HIGH PASS
A/D
A/D
A/D
A/D
SIGNAL PROCESSOR
DIRECT MEMORY ACCESS
OVERCURRENT & DIRECTIONALITY MEMORY
KNOWLEDGE BASE FOR OVERCURRENT & DIR. AND HISTORICAL PAT.
P1
P1
P1
P2
P2
P2
KNOWLEDGE BASE FOR INCREASED FUNCTIONALITY
PARAMETRIC MEMORY
HISTORICAL PATTERNS
OVERCURRENT PROTECTION DIRECTIONALITY HISTORICAL PRECEDENCE
NON-CATASTROPHIC FAULT DETECTION

Figure 6. Block Diagram

The overcurrent protection, directionality, and historical precedence block is used to provide these functions for all of the lines being protected by this remote unit. Therefore, parallelism in the P1 processors is provided to assure appropriate responsiveness. PMCLIPS, utilizing dynamic allocation for many parallel tasks, is favorable on this system.

Finally, the non-catastrophic fault detection block is utilized to diagnose high impedance and arcing faults. The ability to detect incipient faults is also housed here as well as many of the self-testing procedures desired for the system checks. Parallel P2 processors are shown to again emphasize a need for responsiveness. PMCLIPS can also be utilized in this block.

In addition to responsiveness, multiple P1 and P2 systems enhance fault tolerance. At this time P1 and P2 are actually identical 16 bit processors, although this is not a requirement. However, if multiple P1's are not necessary for responsiveness, a configuration allowing a P2 to become a P1 in case of P1 failure is under investigation. The figure shown is merely intended to show the general plan of the architecture. The details of the architecture are beyond the scope of this paper.

CONCLUSIONS:

We believe, after preliminary evaluation of recorded staged fault data, that the most comprehensive diagnostic and protection system for a power distribution system can be implemented in an expert system environment. Due to the lack of a responsive expert system for on-line monitoring environments, we developed PMCLIPS as an expert system shell. PMCLIPS takes advantage of the nature of monitoring systems, i.e. rapidly changing data, and parallelism in order to enhance the responsiveness of expert system. We observed that in a complex monitoring problem, PMCLIPS provides significant speed up (approximately 7x the responsiveness). Finally after reviewing the algorithms and knowledge base which are currently used to diagnose the health of a power distribution system, we have proposed a hardware architecture for our diagnostic and protection expert system (DAPES). Work is progressing to refine this architecture and the knowledge base so that a prototype system can be tested.

REFERENCES

[1] B. Don Russell and Karan Watson, "Power Substation Automation Using a Knowledge Based System - Justification and Preliminary Field Experiments", IEEE Trans. on Power Delivery, Oct. 1987, pp. 1090-1098.

[2] Ram Chinchali and B. Don Russell, "A Digital Signal Processing Algorithm for Detecting Arcing Faults on Power Distribution Feeders", IEEE Trans. on Power Delivery, 88 WM 123-2, Jan. 1988.

[3] Chris Culbert, CLIPS 3.0 Reference Manual, Clearlake, TX: NASA Artificial Intelligence Section, Johnson Space Center, 1986.

[4] Charles L. Forgy, "Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem," Artificial Intelligence, Vol. 19, pp. 17-37, Sept. 1982.

KNOWLEDGE BASE EXPERT SYSTEMS FOR IMPROVED POWER SYSTEM PROTECTION AND DIAGNOSTICS

Dr. B. Don Russell and Dr. Karan Watson
Texas A&M University
College Station, Texas

Abstract - Current practices of power system protection and diagnostics call for the use of dedicated relaying devices typically using preset parametric thresholds to determine the health and state of the power system. It has long been known that certain types of power system faults and abnormal conditions cannot be detected by these existing relaying and protection methodologies. Recent work has shown that knowledge based systems are capable of significantly improving the detection and diagnosis of certain faults and abnormal conditions.

This paper presents the results of research investigations and field tests which show the advantages of knowledge base systems in the detection of incipient fault conditions and certain low grade faults. Expert system algorithms are presented which are to be used in an integrated fashion with existing protection algorithms implemented in computer relays. The ability of these new techniques to automatically adjust for changing power system conditions is a specific advantage. The other advantages of these systems are given with particular emphasis on the types of faults which cannot be currently detected, but which can be found by expert system techniques.

Comments are included on the impact of systems on protection system architectures and on integration with existing protection techniques. Work on modifying traditional expert system software to improve on-line behavior and speed is also discussed.

INTRODUCTION

Conventional automation, control, and protection equipments are passive and responsive. These devices typically respond to changes in the power system as measured in fixed thresholds and preset limits. These systems are designed to prevent catastrophic failure and incorrect operation and work well for most protection, data acquisition, and supervisory control situations. However, feedback control, system diagnostics, advanced detection, and contingency considerations are difficult to implement on the ever changing power system with these conventional approaches. Knowledge based systems have potential for following the changes in the power system and adjusting the criteria accordingly. This paper describes a knowledge base system methodology for real time detection of a power distribution feeder. The goal of the protection system is to extend schemes which focus predominantly on over current thresholds to a scheme where the harder to diagnose incipient faults (e.g. high impedance and arcing faults) are detected. A secondary objective is to provide for more adaptable overcurrent thresholds so that sensitivity to faults is more consistent. In any protection scheme some situations demand an immediate diagnosis and response while it is acceptable in other situations if the response takes a significant period of time. Like any real time system, we must be able to respond to the fastest time demanded in order to protect the controlled network. Thus, protection system responsiveness is one of the main criteria to be used in judging performance of ours or any proposed protection system.

The idea of developing a time restrained, responsive protection system does not usually lead anyone to the conclusion that expert system or knowledge based methodologies should be used. Expert systems have traditionally been consultant by nature, requiring large memories, long response time, and human input (at the keyboard) of relative data. However, due to the heuristic nature of the existing techniques for detecting incipient faults there are indications that future protection systems are well suited for an expert system environment. [1-4] The following are reasons for using an expert protection system.

1) Because of the heuristic and usually empirical nature of the techniques for detecting incipient faults there is a degree of "tuning" required for various feeders with different loads and in different environments. An expert system can make this tuning easier for an operator (versus the programmer) to accomplish.
2) The strategy for adaptable thresholds also requires tuning and constant parameter adjustment so again an expert system environment may make it easier for operators.
3) The incipient fault detection algorithms are subject to many changes. The programming environment of expert systems, where there is a clear separation between data, task knowledge and inference processes, makes it much easier to update the task knowledge.
4) Having to deal with uncertain or incomplete data, which is common when looking for incipient faults, is handled well by expert systems.
5) The potential for development of a learning tool, which can autonomously tune the protection system and even diagnose certain maintenance requirements before they become faults, is much higher in an expert system environment.

Expert systems seem to be our best programming environment, if we can achieve acceptable responsiveness.

FAULT DETECTION METHODOLOGY

Previously reported research supports the conclusion that the expert system environment can improve both feeder detection and diagnostics [5]. Confidence was gained by analyzing collected field data and applying expert system approaches to establish effective performance. Experiments were run using data from in service distribution feeders which were either faulted or had maintenance problems which needed diagnosis.

Presented at the Symposium on Expert Systems Application to Power Systems, Stockholm-Helsinki, Norway-Finland, August, 1988.

Staged Experiments

Researchers at Texas A&M conducted two experiments. Experiment 1 related to the detection of incipient fault conditions which were insufficient in their severity to be detected by convention protection and monitoring equipments. Experiment 2 related to the detection of equipment breakdown over a long period of time which could not be found or diagnosed until a catastrophic failure or tripping of the distribution feeder occurred. These experiments are described as follows.

Experiment 1-Incipient Fault Detection

It has long been known that many distribution faults are not severe enough to be detected by conventional protection and monitoring devices. Field experiments, staged fault tests, and fault statistics have shown that many ground faults begin and remain below conventional detection thresholds.

In response to this characteristic of certain distribution faults, work has been performed for several years to improve the overall sensitivity of fault detection devices to include these low current incipient faults. This work has provided several techniques including those developed at Texas A&M University [6,7,8].

These research investigations have shown that it is generally not sufficient to simply monitor changes in 60Hz fault current and voltage components to determine that these incipient conditions exist or that low current faults have occurred. Other approaches including a much broader data base and the use of historical data are indicated.

In experiments by Texas A&M University, a broad data base including high frequency information above 2kHz was used to detect these low current faults. Simply stated, since a ground fault typically generates an arcing condition which modulates current, high frequency noise is generated which propagates to the substation. This noise has characteristic patterns which can be detected and used as fault indicators.

However, in the development of this technique, it became obvious that normal system changes including feeder noise levels were dramatic. If fixed protection thresholds are used, it is highly probable that many false trips will occur due to normal system variations.

Figure 1 shows the current waveform for a feeder during a ground fault. The unfiltered current waveform is shown together with the current waveform with the 60Hz component suppressed. It is obvious from this figure that the faulted section has considerable "noise" components which could be used as a fault indicator. However, the selection of an arbitrary threshold of detection between pre and post fault levels is difficult due to the fact that the normal noise patterns on the feeder change precipitously over a broad dynamic range. In short, the conventional approach of determining an acceptable vs. an unacceptable level of current at a given frequency and using this as a fault detector is insufficient and will yield an insecure protective device.

ORIGINAL PAGE
BLACK AND WHITE PHOTOGRAPH

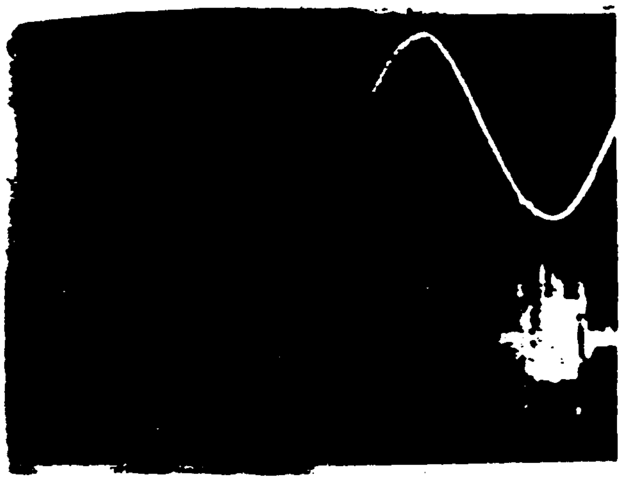


Figure 1 - Faulted Feeder Current Waveforms

Through considerable analysis of recorded data, it was determined that the high frequency on the distribution feeder tended to change over the short and long term related to such factors as the level and type of loading on the distribution feeder. As an example, noise levels under heavy loading during the day might be very high, whereas corresponding measurements taken at night under light load conditions might show little noise activity above 2kHz. The need exists for a "variable sensitivity" device which can adapt not only to daily changes but to long term, seasonal load changes or increased loads due to circuit reconfiguration.

Techniques were developed which compared the present value of high frequency current of the faulted section, not to a preset threshold, but to a calculated value which was a function of historical measurements of the parameter. Simply stated, the threshold was changed dynamically and adapted to the changes in the power system providing a detection threshold which could rachet up or down based on normal, persistent system changes.

Additionally, certain noise generating equipments on the distribution feeder have patterns which can be identified as a function of their repetitive nature and how they occur with reference to the 60Hz waveform. Other switching activity also has specific patterns of behavior which are unique and identifiable.

Experimentation has shown that these features are ideal candidates for processing by an expert system resulting in a high probability of fault detection in spite of dynamic, long term and short term changes in normal system activity.

During field tests of the arcing fault detection technique it was shown that a secure discrimination between a dynamic normal system and a faulted system could only be made using historical data as opposed to instantaneous measurements of parameters. Hence, a knowledge based system with adaptive features is indicated.

It is possible for a human expert to view Figure 1 and determine that a fault or, at least, a very abnormal event is occurring along the distribution feeder. Possibly this is best seen by comparing Figures 2 and 3. Figure 2 shows a "normal" feeder current which has a relatively low high frequency activity. The 60Hz and harmonic components have been suppressed, leaving only the high frequency band from 2-10 kHz. The time period is 1 cycle. Figure 3 shows the same feeder current recorded under during fault. Again, one 60Hz cycle is shown filtered with 2-10 kHz bandpass. When such a figure is compared to a "normal" feeder waveform, several observations can be made. First, it is generally the case for arcing faults that the fundamental waveform magnitude does not increase substantially and therefore conventional protection devices will not react. sensitivity" device which can adapt not only to daily changes but to long term, seasonal load changes or increased loads due to circuit reconfiguration.

Techniques were developed which compared the present value of high frequency current of the faulted section, not to a preset threshold, but to a calculated value which was a function of historical measurements of the parameter. Simply stated, the threshold was changed dynamically and adapted to the changes in the power system providing a detection threshold which could rachet up or down based on normal, persistent system changes.

Additionally, certain noise generating equipments on the distribution feeder have patterns which can be identified as a function of their repetitive nature and how they occur with reference to the 60Hz waveform. Other switching activity also has specific patterns of behavior which are unique and identifiable.

Experimentation has shown that these features are ideal candidates for processing by an expert system resulting in a high probability of fault detection in spite of dynamic, long term and short term changes in normal system activity.

During field tests of the arcing fault detection technique it was shown that a secure discrimination between a dynamic normal system and a faulted system could only be made using historical data as opposed to instantaneous measurements of parameters. Hence, a knowledge based system with adaptive features is indicated.

It is possible for a human expert to view Figure 1 and determine that a fault or, at least, a very abnormal event is occurring along the distribution feeder. Possibly this is best seen by comparing Figures 2 and 3. Figure 2 shows a "normal" feeder current which has a relatively low high frequency activity. The 60Hz and harmonic components have been suppressed, leaving only the high frequency band from 2-10 kHz. The time period is 1 cycle. Figure 3 shows the same feeder current recorded under during fault. Again, one 60Hz cycle is shown filtered with 2-10 kHz bandpass. When such a figure is compared to a "normal" feeder waveform, several observations can be made. First, it is generally the case for arcing faults that the fundamental waveform magnitude does not increase substantially and therefore conventional protection devices will not react. Secondly, the energy levels at all nonfundamental frequencies including harmonics and high frequencies, generally increase substantially and provide a spectrally rich environment with much information for fault detection and characterization.

A human expert viewing a prefault and postfault waveforms would easily conclude that an abnormal event had occurred or an arcing fault was proven. If time were available for the expert to study the waveforms and analyze their behavior over a period of a few seconds to a few minutes, it would be possible to determine whether the event was load switching, capacitor switching, or a low grade fault. The information to make these determinations is generally present in the waveforms, but relaying devices, to date, have not been "intelligent" enough to make the necessary discriminations.

Figure 2. Normal, High Frequency Current Waveform

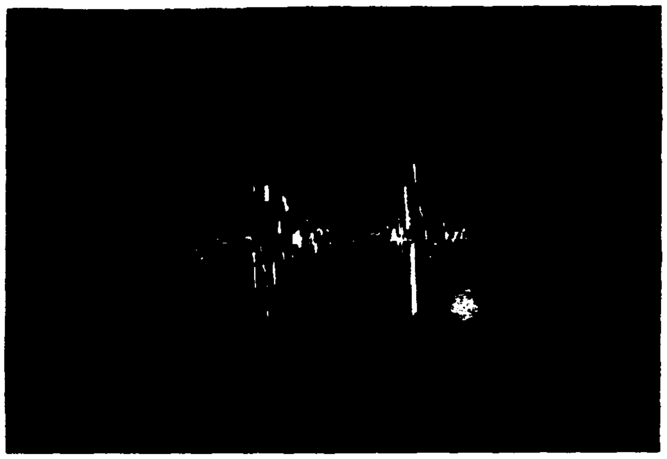


Figure 3. Faulted, High Frequency Current Waveform

Experiment 2-Equipment Failure Diagnosis

During the numerous field tests and measurements made over several years, it was determined that equipment breakdown and deterioration may occur over long periods of time. For example, an incipient transformer fault due to insulation breakdown may occur very slowly before resulting in a catastrophic failure. Other apparatus such as insulators on distribution feeders may have intermittent breakdown due to incipient mechanical failure which persists for weeks or months prior to causing a high current fault. During experiments with Public Service Company of New Mexico, changes in current waveform frequency components were detected on a specific feeder over many weeks of monitoring. The changes in high frequency activity were easily measured and at times were precipitous resulting from insulation breakdown. However, since the fault was not mechanically sustained, system integrity was restored and all indications of the presence of the breakdown were lost.

By careful study of this feeder, it was determined that certain arrestors and insulators were failing and repairs were needed. Effecting these repairs, the current waveforms on the feeder changed accordingly, resulting in a reduction of the noise patterns previously measured. Careful analysis of these noise patterns as correlated to other data predicted the line problems.

This experiment indicates the potential for diagnosing the need for equipment repair and line maintenance. A carefully designed expert system looking at numerous parameters can, through inferences, determine the most probable cause of incipient conditions and prioritize the actions taken to restore the system to 100% integrity.

It has long been desired to transfer the ability of the human expert to the protective relay so that with a very intelligent and flexible yet rigorous system, faults and other abnormal conditions that are presently undetected and undiagnosed could be properly characterised. Several issues must be carefully resolved before such relays can be successfully implemented. A significant problem is the need for "real time" expert systems capable of providing the response necessary for relaying purposes. It is this need that is next discussed.

REAL-TIME EXPERT SYSTEMS

The desire to make expert systems more responsive is not a new concern. Many researchers are currently attempting to find suitable methods for utilizing expert systems in real time applications. One system which has achieved some notable results is the PICON system [9]. This system was developed to run on the LMI Lambda/Plus enhanced LISP machine and to be a user friendly system for program development and a responsive system for real-time operation. Essentially the system provides a standard processor for data manipulation and more numerical tasks and a LOSP processor for the expert system reasoning tasks.

Another real-time expert system environment has been developed in the Hexscon (for Hybrid Expert System Controller) system [10]. This system uses a large machine with its own knowledge base to manage microcomputers which actually interface with the sensors and effectors. The microcomputers contain compiled code for their knowledge base and inference processing. Hexscon chose to use PASCAL instead of LISP because a compiled code by nature runs faster than an interpreted code.

Other application specific examples of real-time expert systems for robotics [11], computer operations [12], nuclear reactor diagnostics [13] as well as others can be found in the literature. B. G. Silverman presents an interesting methodology for real-time supervisory controllers [14]. In his approach Silverman discusses the value of distributed inference engines and knowledge bases in order to improve the responsiveness of the system. He also discusses dual calculus which is an approach where the expert system has two distinct modes of operation. In the first mode, which is most common, the system uses fast reasoning techniques to go from the data to the most likely diagnosis or proposition. In the second mode, the system works to resolve conflicting diagnostics or propositions using fusion algorithms. The fusion algorithms, which can involve Baysian representation [15], combinatorics [16] or the Shafer-Dempater technique [17], work to resolve the conflict when two or more sources have come up with conflicting propositions.

We have approached our protection system trying to capitalize on these other works. We do not intend at this time to use a LISP processor but we are interested, yet not tied to, multiprocessor implementations. We feel that a compiled code will greatly enhance the responsiveness of the system, so we are developing codes in the C language. We are confident that a distributed inference strategy will lead to a more effective protection scheme and a more favorable environment for development. Also, the approach is well suited for dual calculus since some catastrophic faults demand immediate response while other incipient faults require more time to confirm due to the uncertain or incomplete data available. The architecture environment our protection system is to operate in and the programming environment being developed are hereafter discussed.

COMPUTING ARCHITECTURE FOR PROTECTION SCHEMES

We envision a hierarchical computing structure for the protection of power systems. In this structure the top level contains SCADA type systems. This level will provide graphic and reporting capabilities along with high level control techniques. This level can be implemented on a powerful computing facility since the machine can be located in a relatively environmentally friendly area. Much of the operator interface can be handled at this level. On a medium level we envision smaller machines which are more task oriented than operator directed. This level may receive commands and information from higher levels and information from lower levels. At this level such tasks as load management, system reconfiguration, and fault location may be handled. The lower level of the hierarchy is involved in specific diagnostic and protection functions. This level receives information and command from higher levels. This low level acquires data which it formulates into usable information before passing it up the line. The best situation would be to provide as much intelligence as close to the feeder being monitored. The lowest level of the hierarchy is to be discussed.

The development of systems at the low level of the hierarchy indicates that standard microprocessors are potentially the best media for implementation. This is because of the space considerations, environmental robustness, and the economic considerations. We have approached the problem with processors in the Motorola 68000 and Intel iAPX 86/10 families in mind. We are aware of some SISP microprocessors under development, but we do not feel these are required.

On this low level of the computing hierarchy we envision a very distributed system in which individual processing systems have no need for intercommunication. Any communication between processing systems at the low level can only be accomplished by passing information up the hierarchy accomplished by passing information up the hierarchy and then passing it back down. Each of the distributed processing systems will monitor a particular subsection of the distribution system.

Within each of the distributed processing systems we may have a single or multiprocessing system depending upon the timing and complexities required. If multiprocessors are used they will be tightly coupled and have extensive intercommunications. Regardless if one or more processors are used, a distributed inference methodology is utilized.

DISTRIBUTED INFERENCING FOR FAULT DETECTION

In the distributed inference methodology we envision a decentralized expert system. That is, instead of having a centralized knowledge base containing a large number of rules and a sometimes complex resolution methodology we break the expert up into many smaller experts. If time or processing power allows all of the small experts make their analysis and conclusions about the present situations and then conflicts are resolved. In other situations where time is short certain experts will be given more priority than the others, and at time absolute domination of the system. The high priority expert will not be the same in every situation. The generation structure of the environment under development is shown in Figure 4 and each block is discussed below.

Signal and Parameter Conditioning

The functions performed by the signal and parameter conditioning function of the programming include all the processing required to translate sensor outputs into meaningful parameters for diagnosing the status of the power network. Some of the functions call for hardware and some software. Figure 5 shows the generalized steps performed in this area. Sensor data enters the system and is directed to either high resolution or low resolution, fast analog to digital converters. The conversion is handled by hardware, but the decision as to which sensors go to which converter is controlled by an expert system, referred to as the focus expert system.

This expert system gets data from the system characterization block of Figure 4 and decides which signals to convert. The main goal of the expert system is to get as many signals as possible converted with the appropriate resolution. For example say we have four incoming signals and two signals at a time go through the fast A/D converter and one signal at a time goes through the high resolution converter. Now let us assume the conversion rate of the fast converter is three times faster than the high resolution converter. The low resolution converter allows up to a 10% error in the signal value where the high resolution converter allows less than 0.1% error. Suppose we are in a normal (no faults suspected) operating mode where we are well below the overcurrent thresholds and we do not consider any transition in the system less than 20% to be significant. In this case, our focusing expert system would determine the fastest conversion process which is acceptable to sending the four signals, two at a time, through the fast A/D converters. Now suppose that we get a signal from the system characterization block which indicates a suspected fault, but we require more precise data from the first signal to be sure. Therefore, our focusing expert system would cause the first signal to be directed to the high resolution A/D while it passes the other three signals (two at once then the third) through the fast converter. Our expert system also warns the follow-up functions to wait for the appropriate time for high resolution conversion to be complete. Of course, none of this is required if high resolution fast converters are affordable for every sensor signal, but we can also imagine some complex situations where certain signals are temporarily ignored in order to provide as much pertinent data as possible in a minimal time frame.

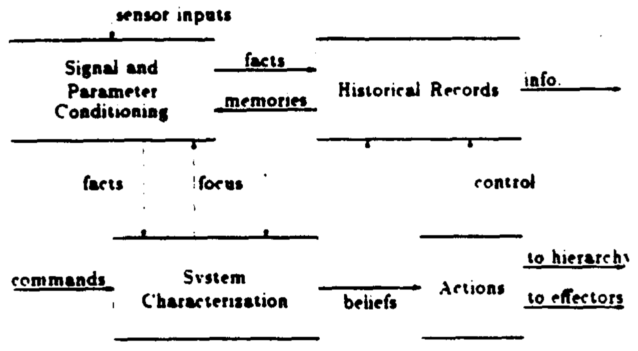


Figure 4. The Distributed Inference System

After digitization of sensor values we usually go through various filtering arrangements. Sometimes we filter noise from the primary frequency signal. Other times we isolate certain harmonics and subharmonics in the system. In this case, another inference process in the focus expert system considers the mode described by the system characterization function, and determines the type of filtering to be done. In a normal, no fault mode, the "filtering" section might not be too concerned about short roll-off in the transitions from pass to nopass areas. However, in certain situations we may choose to neglect one filter while we sharpen the edges of another filter. We could also consider what type of compensation for the spreading of the signal usually caused by digital filtering will be activated at a given time.

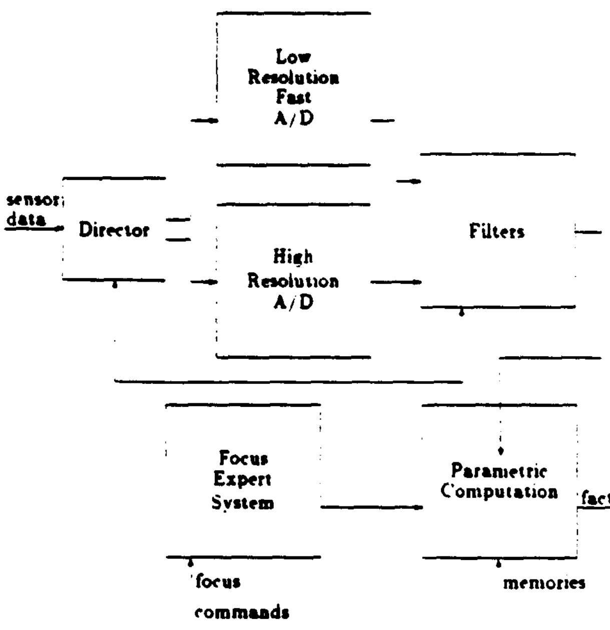


Figure 5. Signal and Parameter Conditioning

Finally, certain parametric values or transforms may be of interest. Information such as sliding averages, Fourier transformations or energy calculations are a few of the parameters of interest. In a noneventful normal mode these parameters may have a certain sequence of calculation. In the mode where diagnostics are being attempted, this regular sequence may be preempted by a new sequence of calculations which are more relevant to the present situation. The outputs of the signal and parameter conditioning blocks are the facts which seem most relevant for the present situation.

Historical Records

The block labeled "historical records" in Figure 4 is predominantly a "memory", however it does have a management expert system which determines what goes into and out of the memory (see Figure 6). In a normal mode of operation (no faults), this system would accept facts and system characterizations into the memory. This data could be continually transmitted a FIFO type fashion or could be buffered and transmitted to a higher point in the computing hierarchy. It is even more likely that the data in memory would be thrown away unless some event indicates a record needs to be saved. The memory management system, MMS, would get requests for certain data in order to make some parametric calculations from the focus expert system in the signal and parameter conditioning block. The MMS would also get request from the system characterization block for certain historical data it requires. The MMS would also handle certain commands from the action block, such as purge all current data, transmit all data in sequence, transmit partial data in memory, etc. The amount of memory being handled by this block would be somewhere between 2 and 10 cycles worth of data. Thus, if your data comes in every millisecond from a 60Hz line, and we are starting 12 facts with 16 bit resolution and 8 system characterizations with 8 bit resolution, we would need

$$(16.6\frac{\text{samples}}{\text{powercycle}} \times (12+8) \text{ bits} = 4260 \text{ bits/cycle}.$$

Therefore, to store 10 cycles of data we would need ~42k bits of memory. This calculation is provided to show the order of memory requirement.

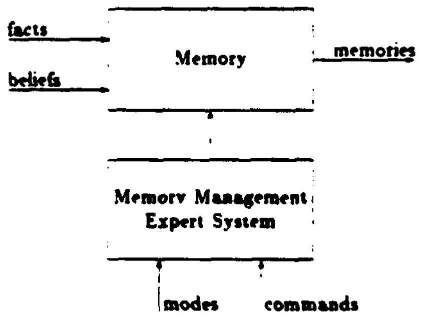


Figure 6. Historical Records

System Characterization

The system characterization block is a combination of several distributed inference systems. These systems are illustrated in Figure 7. Initially, information about present facts, the recent history and various commands are input to the Modus Operandi expert system, MOES. This system uses this information plus the current uncertainties to determine the most effective mode to proceed in. For example, if no facts have undergone a significant transition, no commands from higher up in the hierarchy, and no significant uncertainties about the current state of the system exists, we would enter the normal, no fault mode. This would indicate to other systems to maintain a general monitoring pattern with no intense focus on a particular area. Now, if one of the incoming facts undergoes a significant transition we shift to a mode called the transition mode and investigate what new uncertainties have been generated by this transition. The level of uncertainties would help to begin a focusing process of the relevant area of activity.

The characterization block in Figure 7 represent a section of programming which computes a belief interval. The belief interval for any proposition in the system diagnostics can be computed. However, not every interval must be calculated every computation cycle. The MOES will help determine the mode of operation we are in (e.g. normal, transition, disconnect eminent, evaluating an incipient fault, etc.) and the Inference Strategy Selection System, ISSS, will determine which belief intervals are most pertinent to the given mode of operation. The belief interval is represented by a pair of fractional numbers, each number falling between 0 and 1. The first number represents the amount of evidence we have supporting a given proposition. The second number gives the maximum likelihood that a given proposition is true. Thus as the first increases we have more evidence supporting the proposition and as the second number decreases we have more evidence supporting the inverse of the proposition. The belief intervals for a given proposition are demonstrated in Table 1.

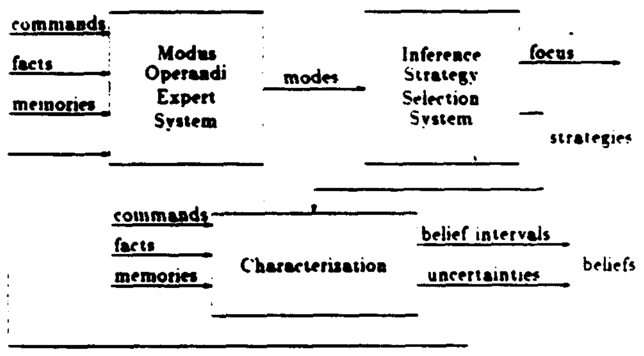


Figure 7. System Characterization

Table I.

Proposition: There is an arcing fault on Feeder #1. columns

Belief Interval	Meaning	Evidence
(0,0)	false	no evidence for strong evidence against
(0,1)	no knowledge	no evidence for or against
(1,1)	true	strong evidence for no evidence against
(4,6)	possible but uncertain	two cycles of data indicated fault but next two cycles showed no fault

Two numbers in the belief interval can be used to determine the uncertainties by subtracting the first number from the second. The confidence level in the proposition is described by the first number in the belief interval.

The MOES operates in a forward chaining mode, i.e. data comes in and a goal or proposition is inferred from the data. The ISSS operates in a backward chaining mode. It receives the most likely propositions from the MOES and determines what facts and belief intervals are most relevant for proving or disproving these propositions. This information is passed on to the focus expert system of the SAPC block as well as the characterization block. Thus the facts will come in with the proper focus and will be used to characterize the most relevant areas of the system.

The Action Block

This block takes the information determined by the characterization block of Figure 7 and determines when and what actions should be taken. The actions can be in the form of effector signals of messages on the hierarchy. The messages can request various maintenance routines be undertaken if a possible fault has not reached a certainty level after a given time period. The action block can also shift the protection system into certain modes of data acquisition or transmission if power disconnect seems eminent. The action block can also put the protection system through certain self diagnostic steps to determine if there are any problems in the computing hardware.

CONCLUSION

It has been shown that the nature and characteristics of low grade faults and other abnormal conditions on distribution feeders are such that detection and characterization is very difficult. The complexity of the electrical signals produced by these events makes it impossible to distinguish them using conventional relay and protection practices or equipment. This detection problem is further complicated by the dynamic and changing nature of the electric power system.

Field studies have shown that an "intelligent" protection system using knowledge base and expert system methodologies can significantly improve the probability of detecting false and abnormal events. This is primarily due to the increased ability of such systems to do detailed signal processing in light of historical signal patterns and distribution feeder behavior. Expert systems offer the possibility of "adaptive relaying" as well as the possibility of probabilistic determination of the existence of the fault. Given the subjective nature of some of the fault data indicators and the numerous parameters to be evaluated, expert systems offer distinct advantages.

Expert systems also offer the possibility of designing self modifying, self tuning hardware systems. Advantages can be obtained in signal conditioning and conversion performance, real time algorithm modification, and statistical evaluation of the results of detection algorithms.

Further work is needed to fully test the suggested architectures and methods presented in this paper. Work is underway to implement the architecture and test it using both recorded

ORIGINAL PAGE IS
OF POOR QUALITY

REFERENCES

1. B. M. Aucoin, J. Zeigler, B. Don Russell, "Feeder Protection and Monitoring Systems, Part I: Design Implementation and Testing", *IEEE Transaction on Power Apparatus & Systems*, Vol. PAS-104, April 1985, pp. 873-80.

2. B. M. Aucoin, J. Zeigler, B. Don Russell, "Feeder Protection and Monitoring Systems, Part II: Staged Fault Testing Demonstration" *IEEE Transactions on Power Apparatus*, Vol. PAS-104, No. 6, pp. 1456-1462.

3. B. Don Russell, Ram Chinchali, C. J. Kim, "Behavior of Low Frequency Spectra During Arcing Fault and Switching Events", presented at IEEE/PES 1987 Summer Meeting, San Francisco, California, 87 SM 633-1.

4. B. Don Russell, K. Metan, Ram Chinchali, "An Arcing Fault Detection Technique Using Low Frequency Current Components - Performance Evaluation Using Recorded Field Data", *IEEE Transactions on Power Delivery*, 87 SM 634-9.

5. B. Don Russell, Karan Watson, "Power Substation Automation Using a Knowledge Based System-Justification and Preliminary Field Experiments", IEEE/PES 1986 T&D Conference, Anaheim, CA, 86T&D, 600-1.

6. S. Balser, K. Klements, D. Lawrence, "Microprocessor-Based Technique For Detection of High Impedence", *IEEE Transactions on Power Apparatus and Systems*, Paper 86 WM 155-6.

7. R. Lee, M. Bishop, "A Comparison of Measured High Impedance Fault Data to Digital Computer Modeling Results", *IEEE Transactions on Power Apparatus and Systems*, Paper 85 WM 232-4.

8. B. M. Aucoin and B. Don Russell, "Distribution High Impedence Fault Detection Utilizing High Frequency Current Components", *IEEE Transactions on Power Apparatus and Systems*, Vol. PAS-101, No. 6, 1982.

9. David Leinweber and John Perry, "Controlling Real-Time Processing on the Space Station with Expert Systems", SPIE *Proceeding Space Station Automation II*, Oct. 1986, pp. 11-39.

10. M. Lattimer Wright, M. W. Green, Gundrun Fiegl and Perry Cross, "An Expert System for Real-Time Control", *IEEE Software*, March 1986, pp. 16-24.

11. B. G. Silverman, et.al., "Expert Systems and Robotics for the Space Station", in *Expert Systems in Government Conf.*, Prof., IEEE Comp. Soc. Press., 1985.

12. R. L. Ennis, J. H. Briesman, S. J. Hong, et al, "A Continuous Real-Time Expert System for Computer Operations", *IBM Journal of Research and Development*, Jan. 1986, Vol. 30, No. 1, pp. 14-27.

13. William R. Nelson, "REACTOR": An Expert System for Diagnosis and Treatment of Nuclear Reactor Accidents". *Proc. of the National Conference on Artificial Intelligence*. Aug. 1982, pp. 296-301.

14. B. G. Silverman, "Distributed Inference and Fusion Algorithms for Real-Time Supervisory Controller Positions", *IEEE Trans. on Systems, Man, and Cybernetics*, Mar/Apr. 1987, pp. 230-239.

15. R. O. Duga, P. E. Hart and N. J. Nilsson, "Subjective Bayersion Methods for Rule-Based Inference Systems", in *Proc. Nat. Computer*, Conf.j, 1976, 11 1075-1082.

16. L. Zaden, "The Role of Fuzzy Logic in the Management of Uncertainty in Expert Systems", *Fuzzy Sets*, Syst., Vol. 11, pp. 199-227, 1983.

17. H. Prade, "A Computational Approach to Approximate and Plausible Reasoning with Applications to Expert Systems", *IEEE Trans. Pattern Alan. & Machine Intell.*, Vol. PAS 1-7, pp. 260-283, 1985.

Dr. B. Don Russell and Dr. Karan Watson are researchers with the Texas Engineering Experiment Station and Department of Electrical Engineering at Texas A&M University. They are presently funded in their investigations by the United States National Science Foundation, the Electric Power Research Institute of Palo Alto, California, the National Aeronautics and Space Administration, and several electric utilities.

AN ARCING FAULT DETECTION TECHNIQUE USING LOW FREQUENCY CURRENT COMPONENTS - PERFORMANCE EVALUATION USING RECORDED FIELD DATA

B. Don Russell
Senior Member

Ketan Mehta
Student Member

Ram P. Chinchali
Student Member

Texas A&M University
College Station, Texas

Abstract - For distribution feeders, several low frequencies of the current waveform exhibit modified behavior under fault conditions. Two frequencies, 180 Hz and 210 Hz, were selected for study due to strong magnitude variations associated with arcing faults. A hierarchical algorithm with adaptive characteristics is presented along with the performance results when applied at these low frequencies. The various parameters which affect the sensitivity of the algorithm are discussed. The results of the tests using recorded field data are given. Discussion is included of the effects of using a digital filter front end for the algorithm.

INTRODUCTION

Some distribution primary faults exhibit too low fault currents to be detected by overcurrent protection. There is a very strong desire in the utility industry to detect these faults, because undetected faults present a potential hazard to public safety. There has been much investigation of the nature of these high impedance faults and certain devices and schemes have been developed which can detect some of these faults. A statistical algorithm was developed by Power Technologies, Inc. [1] for the detection of high impedance faults based on changes in sequence current unbalance. The microcomputer-based Feeder Protection and Monitoring System (FPMS) developed at Texas A&M University [2] included an overcurrent relay for overcurrent protection and also identified some low current faults not cleared by the overcurrent protection. Phadke [3] implemented a microprocessor based digital relaying scheme in which changes in the positive, negative, and zero sequence components of the fundamental power frequency are monitored continuously in real-time and the presence of high impedance faults is detected based on these changes. A prototype Ratio Ground Relay was developed at PP&L [4] to provide detection of broken conductor faults. A fault detector was also designed by researchers at Texas A&M University [5] for the detection of those faults which involve ground and which arc. This detector identified the burst noise caused by arcing faults at high frequency, specifically 2 - 10 kHz components. This arcing fault detector operates on an increase in the wideband frequency components of current generated by arc burst noise. Although, only the 2 - 10 kHz components of arcing were investigated, it is well-known that arcing generates components both above and below this frequency range. Hence, it was decided to further investigate the significance of the low frequency components as indicators of the presence of faults on distribution primary lines. The specific frequencies placed under consideration were in the range of 30-360 Hz. It was also desired to compare the sensitivity of the two detectors - the high frequency and the low frequency arcing fault detectors.

Prior to development of the detection system, it was necessary to identify specific frequencies from the range of 30-360 Hz which could be used as good indicators of fault occurrence on the distribution lines. Hence, data from staged faults which had been recorded on analog tape was statistically analysed. Frequencies studied included 30 Hz, 90 Hz, 150 Hz, 180 Hz, 210 Hz. These were studied for their variation from unfaulted to faulted conditions. The various characteristics that were considered included burst lengths, magnitudes and frequency of occurrence of these bursts. It was concluded that among the harmonic and in-between harmonic (extra-basal) frequencies in the low frequency range, the two frequencies that gave a very good indication of faults were 210 Hz and 180 Hz. One of the criterion for a good indicator was the necessary precaution that the component should not raise false alarms of faults during air-switch operation, capacitor bank operation, and other normal switching operations. Another criterion was a significant dynamic change in the magnitude from unfaulted to faulted conditions. The precise algorithm used to detect faults on distribution primary lines based on the above two frequencies and their corresponding characteristics is explained after a brief discussion about the overall system architecture.

DETECTOR ARCHITECTURE

The system architecture of the low frequency arcing fault detector can be described as follows. The input signal, after attenuation through a current transformer, is fed to a low pass filter which attenuates all frequencies above 400 Hz. The output signal from the low-pass filter is then passed into an analog-to-digital converter whose sampling rate is adjusted to satisfy the Nyquist criterion. The sampled waveform is then subjected to a software detection algorithm, the program being resident in a microprocessor based system. The software detection algorithm is crucial and determines the performance of the entire detection system.

87 SM 634-9 A paper recommended and approved by the IEEE Power System Relaying Committee of the IEEE Power Engineering Society for presentation at the IEEE/PES 1987 Summer Meeting, San Francisco, California, July 12 - 17, 1987. Manuscript submitted January 29, 1987; made available for printing April 24, 1987.

ORIGINAL PAGE IS
OF POOR QUALITY

DETECTION SCHEME

A flowchart to indicate the logic incorporated in the detection algorithm is shown in Figure 1. The algorithm essentially determines the energy contained at any time in the low frequency signal. The detection technique utilises the summation of the square of the filtered low frequency data samples over an entire cycle of 60 Hz. The detection algorithm does not consider individual impulses in the filtered signal but attaches importance to their cumulative effect over a predefined interval of time. Characteristics of the software detection algorithm include its adaptability, hierarchical nature, and "expertness".

It is imperative for the algorithm to track variations in the low frequency signal which are not associated with faults. Each feeder may have a different normal noise level which itself may be dependent on and vary with the load. A typical distribution system exhibits a periodic load cycle. Hence the detection algorithm must be adaptive to these changes in the load and at the same time avoid a complicated procedure for calibrating arbitrary pickup levels for fault detection. Hence, the algorithm used to detect arcing faults in a typical distribution system should have strong adaptability to these periodic variations in the load so as to maintain reliability of detection. The feature of adaptability has been incorporated in the detection algorithm by making use of two thresholds - a dynamic threshold and a static threshold. The thresholds are used to adjust to changes in the load on a continuous basis.

A hierarchical nature is bred into the algorithm by making use of three levels in the detection process before signaling a fault. The system starts by recognising a "disturbance" and on the occurrence of a disturbance, the system devotes its attention to trying to verify if the disturbance qualifies as an "event". An event recognition is followed by an attempt to classify the episode into either a "fault" or a normal occurrence. The progression of the detection scheme from one level to another and the updating of all values through this progression is automatic. The progression also implies the use of time as a discriminatory factor. The definitions for these hierarchical levels can be given as follows :

Disturbance : A cycle of data showing a certain percent increase of energy over the average energy per cycle, the average being calculated over some previous period of time, constitutes a disturbance. Thus, if a cycle shows a certain percentage (e.g. 25 percent) increase in energy over the previous average, a disturbance is said to have occurred. If the energy present in the present cycle is reasonably equal to the previous average, then a new average is calculated and disturbance detection is begun again. The purpose of the disturbance detection routine is to identify changes in the low frequency current on the feeder. Such an occurrence could be one of any number of events such as load drop or addition, switching event, bolted fault, or high impedance fault.

Event : Once a disturbance is detected, a preselected series of cycles of data are tested. If a set percentage of these cycles show a certain percentage increase of energy per cycle over the average energy per cycle (the average being calculated over some previous period of time), then an event is said to have occurred. A point of interest here is the fact that the dynamic threshold is updated even after the recognition of a disturbance. Statistical analysis has shown that a 75 percent increase in the low frequency component energy is reasonable for event identification, and this factor was used in the prototype program. Typically, five cycles could be tested where 3 of 5 showing the requisite increase might dictate a necessity to proceed onto the next hierarchical level of fault detection. By varying the number of test cycles, the sensitivity of detection can be varied.

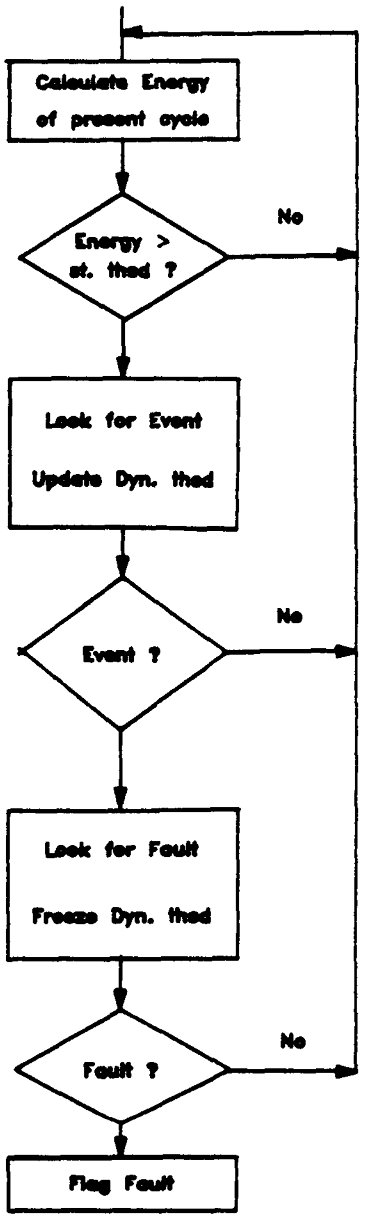


Figure 1. Flowchart for the Detection Algorithm.

Fault : Once an event is recognised, control is transferred to the fault identification routine in the detection algorithm. The dynamic threshold is frozen in the fault identification routine. This action is necessary because percentage (relative) change in the signal magnitude is used to detect faults as opposed to absolute changes from predefined fixed thresholds. One choice involved is the number of cycles after an event, required to show an increase in low frequency activity, before a fault is specified. The compromise is between correct identification of intermittent or very low grade faults and the possibility of identifying a normal event as a fault. Thus, an important parameter involved is the choice of the length of time after an event that is allowed for evaluating the low frequency current to make the trip/notrip decision.

The various parameters involved are adjusted so as to obtain an "optimal " detection scheme which takes a reasonable length of time for making decisions and is sensitive enough to make distinctions between arcing faults and normal operations. The validation of this detection algorithm is explained in the following section.

DATA ANALYSIS

Data analysis was performed for two specific frequencies to validate the efficacy of the algorithm previously described. The frequencies chosen were 180 Hz and 210 Hz. Statistical analysis performed on arcing fault and normal switching data indicates that several harmonic and subharmonic frequencies in the range 0 - 360 Hz can be used to distinguish arcing faults from normal switching events. Several of the 'in-between' harmonic frequencies can be used to make distinctions between faults and normal switching operations and 210 Hz had the added feature of a maximum dynamic change in energy during arcing faults. Similarly, the 180 Hz component indicated a maximum dynamic change for the harmonic frequencies. In this section, we proceed to validate the algorithm using these two specific frequencies for various arcing faults and switching conditions.

The data obtained from the field for staged faults was first low-pass filtered and then it was sampled using a 12-bit analog-to-digital converter. Each record of faulted data was sampled for a duration of 10 seconds and then filtered for the above two specific frequencies using linear phase FIR digital filters. The Parks-McClellan algorithm was utilized to design the two digital filters - one for the ' in-between' harmonics and the other for the harmonic frequencies. Each filter had a length of 128 coefficients. The filtering was done in the time domain using discrete convolution of the digitized data with the filter coefficients. Numerous such filtered data records were subjected to the arcing fault detection algorithm.

The use of a digital filter imposes several limitations on the processing of the data. Due to the fact that the digital filter selected was of order 128, the use of the convolution technique dictates that the first 128 output samples from the filter be ignored. Thus the transient response of the filter imposes one limitation. The filter also results in a considerable amount of 'ringing' in the output. This, in effect, means that a small burst of energy in the input waveform results in a spreadout of energy over a few cycles in the output waveform from the filter. For an arcing burst spanning k cycles of the fundamental frequency, the digital filter output will indicate an increase in energy over the next (n * k + 128) samples, where n is the samples per cycle of the fundamental frequency. This phenomenon is exemplified in the unfiltered time domain waveform of Figure 2a which shows a single cycle burst between sample numbers 3841 and 3969 that gets translated into a burst of several cycles in the 180 Hz filtered output waveform of Figure 3a. The same point is stressed by comparing Figure 2a and the 210 Hz filtered waveform of Figure 4a for the same range of sample numbers. The two limitations have been taken into account and can be somewhat mitigated in the hardware implementations of the system through the use of front end analog filters.

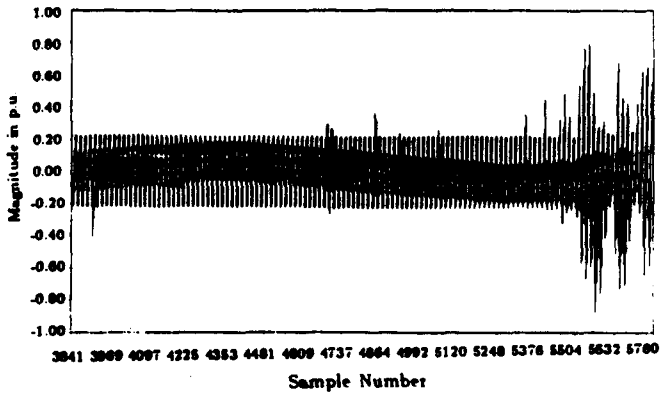


Figure 2a. Variation of Unfiltered Signal over Time.

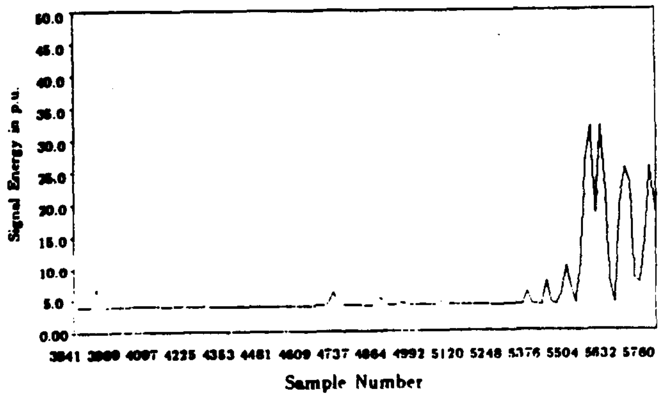


Figure 2b. Variation of Unfiltered Signal Energy over Time.

Both the unfiltered data from staged faults and the filtered data were processed using the detection scheme and the corresponding energy variation of the signal. Figure 2b shows the energy variation of the unfiltered signal over time. As can be seen from the figure, there is a considerable increase in the energy of the signal for long duration arcing bursts occurring between sample numbers 5376 and 5760. Even for single cycle bursts, a detectable increase in energy is evidenced. The detection algorithm adaptively tracks these changes in energy by correspondingly incrementing the dynamic threshold.

Similarly, Figure 3b shows the variation of the signal energy of the 180 Hz component for both, small and long duration bursts. As mentioned earlier, the change in energy in this case is seen to persist over a somewhat exaggerated duration due to the response of the filter. Figure 4b shows the corresponding energy variation of the 210 Hz signal. The output of the algorithm is shown in Table 1. This output corresponds to the processing of the unfiltered data by the algorithm. The increase in energy of the signal between sample numbers 3841 and 3969 in Figure 2b is detected as a disturbance at sample number 3920 as shown in Table 1. However, as the increase in energy did not persist for a sufficient duration, the dynamic threshold was automatically reset and the episode did not progress into an event. Similar such disturbances were detected at sample numbers 4704, 4848, 5344, 5408, 5456, and 5472. These disturbances correspond exactly to increases in signal energy in Figure 2b.

The entire progression of the detection scheme from a disturbance to an event and eventually to the recognition of a fault is indicated in Table 1 starting at sample number 5520. This entire episode originated as the detection of a disturbance at sample number 5520 when a sufficient dynamic change was detected in the signal energy. At this juncture, the control of the algorithm is transferred to the event detection routine where a counter is incremented to track the number of subsequent cycles that show increased signal energy over the dynamic threshold, which is also updated every cycle. This counter reports a count of 3 which is deemed sufficient

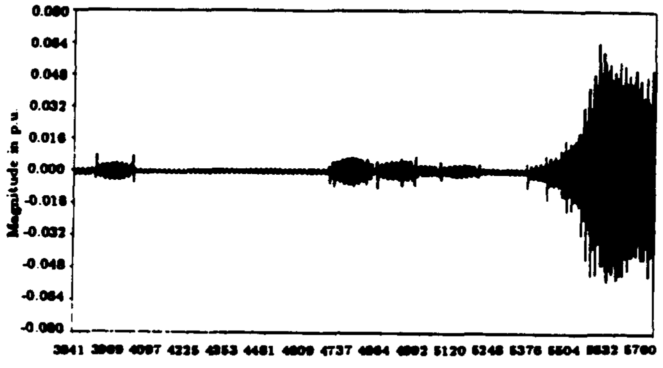


Figure 3a. Variation of 180 Hz Filtered Signal over Time.

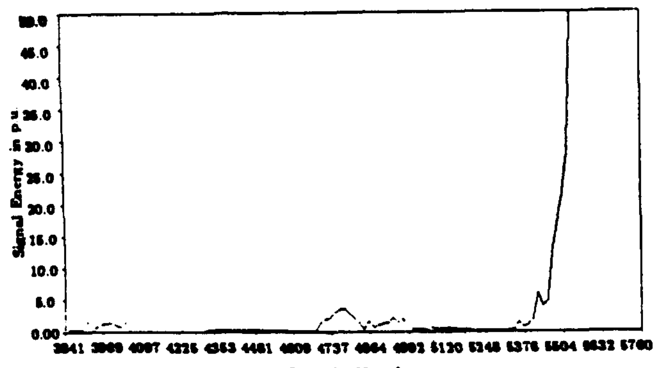


Figure 3b. Variation of 180 Hz Filtered Signal Energy over Time.

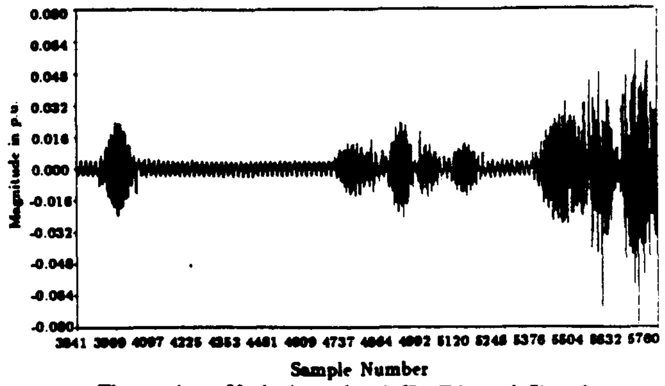


Figure 4a. Variation of 210 Hz Filtered Signal over Time.

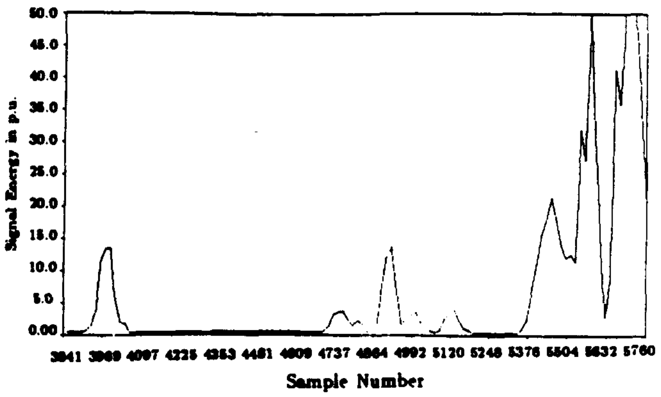


Figure 4b. Variation of 210 Hz Filtered Signal Energy over Time.

to classify the episode as an event. Hence, control is transferred to the fault detection routine. The purpose of the fault detection routine is to identify an event and classify it on a fault/no fault basis. The fault detection routine uses time as the discriminating factor to achieve this objective. In the example at hand, the episode was found to progress into a fault and is reported as such in Table 1. The performance of the detection scheme was tested successfully with the aid of a number of such arcing fault data from different site locations. The remaining problem was to test the algorithm for its performance during normal switching operations where no false alarms are expected. The three normal operations placed under consideration were capacitor bank operations, air switch operations and load tap changer operations.

Table 1. Output from the Detector with the use of Unfiltered Data from an Arcing Test.

```
Disturbance found at   3920
Disturbance found at   4704
Disturbance found at   4848
Disturbance found at   5344
Disturbance found at   5408
Disturbance found at   5456
Disturbance found at   5472
Disturbance found at   5520
Event found with   3 counts at 5520
Fault start at         5520
```

ORIGINAL PAGE IS
OF POOR QUALITY

In Figure 5a is shown the 180 Hz signal energy variation over time when a capacitor bank is disconnected from the feeder. The switching translates into a step decrease of signal energy between sample numbers 4225 and 4481. The 180 Hz signal energy variation of Figure 5b, on the other hand, shows a step increase due to a capacitor bank being switched onto the feeder. The step decrease in the signal energy is seen to last for the entire interval of time (sample numbers 4481 to 6657 from Figure 5a and Figure 5b) for which the capacitor bank was absent. Figure 6a and Figure 6b correspond to the 210 Hz signal energy variation for the same data record. In this case, the switching operation translates into essentially an impulse, rather than a step as in the case of the 180 Hz component. Thus, only a brief disturbance is observed in the signal energy. This characteristic of the 210 Hz component can be utilized to discriminate arcing faults from normal switching events. Table 2 is the output resulting from the detection algorithm due to the processing of the 180 Hz filtered data. As desired, the detection scheme 'smartly' recognizes these transients as disturbances and not as an event or a fault. The output due to the processing of the 210 Hz filtered data is depicted in Table 3.

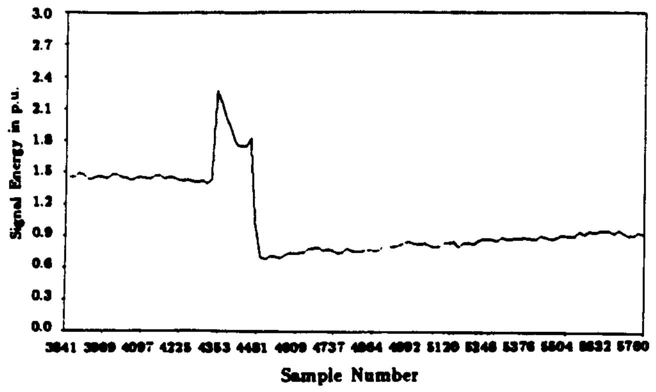


Figure 5a. Variation of 180 Hz Filtered Signal Energy over Time–Capacitor Bank Switched Off.

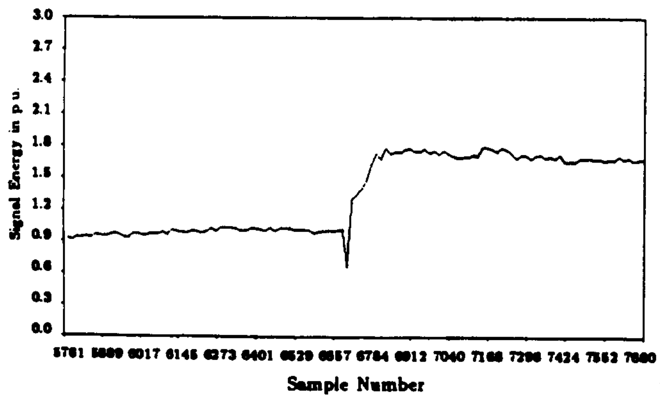


Figure 5b. Variation of 180 Hz Filtered Signal Energy over Time–Capacitor Bank Switched On.

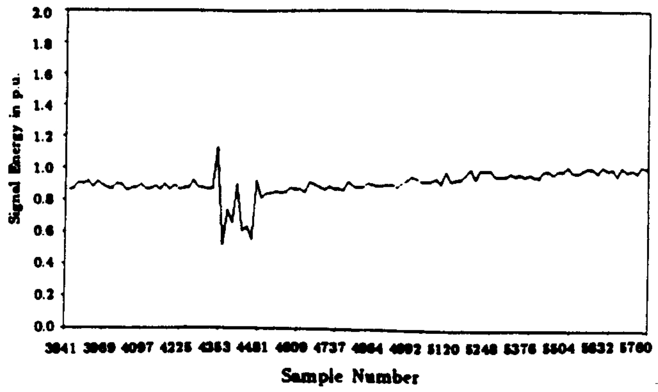


Figure 6a. Variation of 210 Hz Filtered Signal Energy over Time–Capacitor Bank Switched Off.

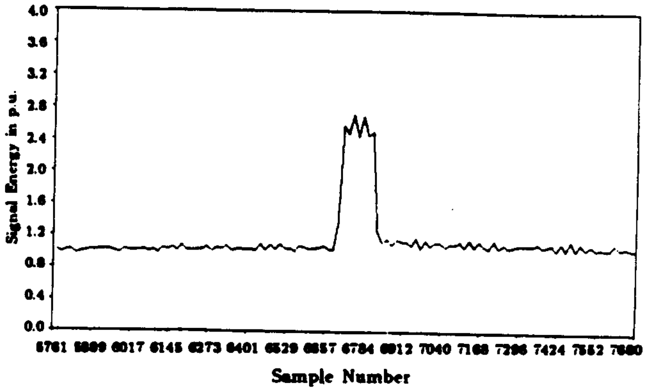


Figure 6b. Variation of 210 Hz Filtered Signal Energy over Time–Capacitor Bank Switched On.

Table 2. Output from the Detector with the use of 180 Hz Filtered Data from an Arcing Test.

```
Disturbance found at   3920
Disturbance found at   3936
Disturbance found at   3952
Disturbance found at   3968
Disturbance found at   3984
Event found with  5 counts at 3984
Event starts at        3904
Disturbance found at   4352
Disturbance found at   4448
Disturbance found at   4528
Disturbance found at   4544
Disturbance found at   4560
Disturbance found at   4576
Disturbance found at   4592
Event found with  5 counts at 4592
Event starts at        4512
Disturbance found at   4928
Disturbance found at   4944
Disturbance found at   4960
Disturbance found at   4976
Disturbance found at   4992
Event found with  5 counts at 4992
Event starts at        4912
Disturbance found at   5328
Disturbance found at   5344
Disturbance found at   5360
Disturbance found at   5376
Disturbance found at   5392
Event found with  5 counts at 5392
Fault starts at        5312
```

Finally, the performance of the algorithm was tested for air switch operations. A part of the staged tests included an air switch being repeatedly switched. The energy variation for the 180 Hz signal and the 210 Hz signal for this operation is shown in figures 7a and 7b respectively. The detection algorithm did not flag any events but successfully recognized the disturbances, as desired.

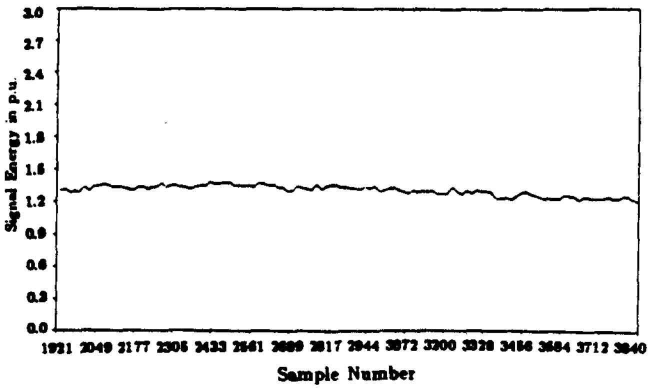


Figure 7a. Variation of 180 Hz Filtered Signal Energy over Time–Air Switch Operation.

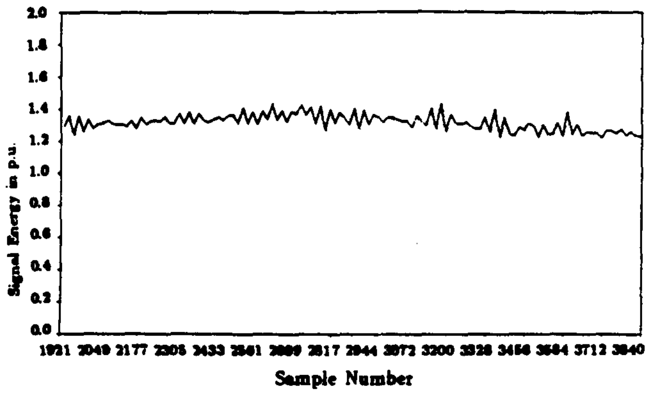


Figure 7b. Variation of 210 Hz Filtered Signal Energy over Time–Air Switch Operation.

SENSITIVITY OF THE DETECTION SCHEME

The sensitivity of the detection scheme described above is determined by various parameters including the transient response of the front end digital filter for the frequency under consideration, the frequency itself, the rate of varying the dynamic threshold to adapt to the ambient noise in the feeder and the preset value of the static threshold. A qualitative analysis was undertaken to determine the sensitivity of the algorithm to the above parameters. To determine the effects of the frequency in use for detection and the digital filter response, both unfiltered data and data filtered at two specific frequencies was processed. Table 4 depicts the output from the algorithm when 180 Hz was used to detect arcing faults. The data record is the same as the one used to obtain output of Table 1, except that it was prefiltered for a 180 Hz signal content. Due to the 'ringing' effect of the filter, the single cycle burst at sample number 3920 which was merely a disturbance in Table 1 is now classified to be an event in Table 4. In order to compensate for this over-sensitivity, the time duration in the event detection routine has to be increased. This modification is imperative due to the use of a digital filter but may become redundant with the use of an analog filter. If digital filters are to be used, this phenomenon must be carefully considered, particularly in light of the processing time required to implement the filters.

The sensitivity of the algorithm to the frequency in use can be explained by comparing Tables 2 and 3. In a certain fixed period of time, there exists a greater number of cycles of 210 Hz than of 180 Hz. Because of this fact, a single 180 Hz cycle burst will spread over a larger number of cycles of 210 Hz. Since time is used as the discriminatory factor in the event detection routine, the parameters need to be fine tuned to obtain the same sensitivity for both frequencies. This explains the anomaly between the detection of an event in Table 3 and the detection of a disturbance in Table 2. It also exemplifies the complexity of setting 'sensitivity' levels given the wide variations in fault characteristics.

Table 3. Output from the Detector with the use of 180 Hz Filtered Data from a Capacitor Bank Operation.

```
Disturbance found at   4352
Disturbance found at   4480
Disturbance found at   4592
Disturbance found at   4688
Disturbance found at   4784
Disturbance found at   4880
Disturbance found at   4976
Disturbance found at   5072
Disturbance found at   5168
Disturbance found at   5264
Disturbance found at   5360
Disturbance found at   5456
Disturbance found at   5552
Disturbance found at   5648
Disturbance found at   5744
Disturbance found at   5888
Disturbance found at   6000
Disturbance found at   6128
Disturbance found at   6272
Disturbance found at   6448
Disturbance found at   6704
Disturbance found at   6720
Disturbance found at   6736
Disturbance found at   6752
Disturbance found at   6768
Event found with   5 counts at 6768
Event starts at        6688
```

Table 4. Output from the Detector with the use of 210 Hz Filtered Data from a Capacitor Bank Operation.

```
Disturbance found at   4352
Disturbance found at   6720
Disturbance found at   6816
Disturbance found at   6912
Disturbance found at   7008
Disturbance found at   7104
Disturbance found at   7200
```

CONCLUSION

An adaptive, hierarchical and 'expert' algorithm has been presented for the detection of high impedance arcing faults on distribution feeders. The detection algorithm was successfully tested with faulted and normal switching data obtained at different test sites. A qualitative analysis was performed using two specific frequencies to identify the various parameters that affect the sensitivity of the detection scheme. The algorithm can be implemented inexpensively in a microprocessor based architecture and can be integrated with other detection schemes for more secure fault identification.

ACKNOWLEDGEMENT

The Electric Power Institute at Texas A&M University wishes to acknowledge the financial support of numerous electric utilities, Dow Chemical, and 3M Corporation without which this research would not be possible.

REFERENCES

1. "Detection of High Impedance Faults," EPRI Report EL-2413, Prepared by Power Technologies, Inc., June, 1982.
2. B. M. Aucoin, J. Zeigler, and B. D. Russell, "Feeder Protection and Monitoring System, Part I: Design, Implementation and Testing," IEEE Transactions on Power Apparatus and Systems, Vol. PAS-104, April, 1985, pp. 873-879.
3. A. G. Phadke and He Hankun, "Detection of Broken Distribution Conductors," Proceedings of IEEE Southeast Conference, Raleigh, N. Carolina, Paper No. CH2161-8/85/0000-0074.
4. H. Calhoun, M. T. Bishop, and C. H. Eichler, "Development and Testing of an Electro-mechanical Relay to Detect Fallen Distribution Conductors," IEEE Transactions on Power Apparatus and Systems, Vol. PAS-101, June, 1982, pp. 1643-1650.
5. "Detection of Arcing Faults on Distribution Feeder, " EPRI Report EL-2757, Prepared by Texas A&M University, December, 1982.

BIOGRAPHY

B. Don Russell, (SM) received B.S. and M.E. degrees in Electrical Engineering at Texas A&M University. He holds a Ph.D. from the University of Oklahoma in power systems engineering.
Dr. Russell is Associate Professor of Electrical Engineering and directs the activities of the Power Systems Automation Laboratory of the Electric Power Institute, Texas A&M University. His research centers on the use of advanced technologies to solve problems in power system control, protection, and monitoring. He is the recipient of several awards for advanced technology applications. Dr. Russell chairs several working groups and subcommittees of PES.
Dr. Russell is a member of the Substation Committee and Power Engineering Education Committees of PES. He is a registered professional engineer and a director of the Texas Society of Professional Engineers.

Ketan Mehta, (S'86) was born in September, 1964 and received his B.E. (Hons) degree from the University of Bombay, India in the year 1985. Mr. Mehta is currently working towards a Master of Science degree in Electrical Engineering at Texas A&M University.

His current research interests include the application of artificial intelligence methodologies to various disciplines such as VLSI design automation, power system protection and parallel processing architectures. Mr. Mehta also possesses immense experience in the development of microprocessor-based systems for equipment control and monitoring.

Ram Chinchali, (S'84) received B.E. (Hons) degree from the University of Madras, India in 1981 and a M.E. in Electrical Engineering from Texas A&M University in 1984. He is currently working toward his Ph.D. degree.
Mr. Chinchali's research interests are microprocessor applications in power system control and protection. During the period between 1981 and 1983, he was a Relay Engineer at English Electric Co., India where he developed protection schemes for 220 KV and 400 KV substations.

BEHAVIOUR OF LOW FREQUENCY SPECTRA DURING ARCING FAULT AND SWITCHING EVENTS

B. Don Russell, Senior Member
Ram P. Chinchali, Student Member
C. J. Kim, Student Member

Texas A&M University
College Station, Texas

ABSTRACT - Distribution feeder faults modulate primary current and generate noise through arcing phenomena. The variation and behaviour of selected low frequencies during fault conditions are herein presented. These are contrasted to normal events such as feeder switching and capacitor bank operations. Recorded field data has been analysed and is statistically presented in the paper. Specific behaviour characteristics such as arc duration, arc repetition rate, and magnitude of low frequency spectra are presented. Comparisions are made for different soil types and conditions.

INTRODUCTION

High impedance faults can be described as those distribution feeder faults that do not draw sufficient fault current to be detected by conventional protective devices such as phase overcurrent and/or ground relays. Such faults may be caused by a downed conductor on a poorly conducting surface. Often, arcing is associated with these faults which poses a potential hazard to public safety. It is therefore important that these faults be detected quickly and the faulted feeder isolated. There has been considerable interest in solving this problem resulting in several major research efforts.

Phadke [1] has suggested a microprocessor based digital relaying scheme in which changes in the positive, negative and zero sequence components of the fundamental power frequency are monitored continuously in real time. The presence of a high impedance fault is detected based on changes in the ratio of the symmetrical components. Balser [2] suggested a technique which monitors the imbalance in the fundamental, third and fifth harmonic feeder currents and performs a statistical evaluation of the present imbalance relative to the past imbalance. Using hypothesis testing, the presence of a fault is detected if the chi-square test statistic exceeds a pre-determined threshold value. Graham [3] suggests monitoring the distribution feeder input impedance at high frequencies in the range 50 KHz to 100 KHz. Russell [4,5] suggested monitoring the energy of high frequency components in the range 2 KHz to 10 KHz to detect arcing faults. There are other significant schemes including the ratio ground relay resulting from research at Westinghouse and PP&L. [6,7]

This paper presents the behaviour of several low frequency spectra during arcing fault and normal switching conditions. It has been observed that the arcing phenomena associated with high impedance faults causes certain low frequency spectra to change in magnitude and phase from pre-fault conditions. The frequency spectra selected for investigation were :

1.	30 Hz	7.	60 Hz
2.	90 Hz	8.	120 Hz
3.	150 Hz	9.	180 Hz
4.	210 Hz	10.	240 Hz
5.	270 Hz	11.	300 Hz
6.	330 Hz	12.	360 Hz

Of these spectra, the first six comprise the 'in-between' harmonic frequencies and the last six, the harmonics of the power frequency. The different events that were investigated to characterise the behaviour of the above spectra included :

1. Arcing faults on different soil conditions.
2. Arcing faults with capacitor bank switched off.
3. Arcing faults with capacitor bank switched on.
4. Arcing faults with air switch operations.
5. Air switch operations.
6. Capacitor bank operations.
7. Load tap changer operations.

The first four events are fault conditions on distribution feeders and the last three are normal switching events. The behaviour of the magnitude of the low frequency spectra for the events listed above is the primary subject of this paper.

DATA ACQUISITION AND PROCESSING

A frequency domain analysis was performed for each event investigated using certain digital signal processing techniques. Analog recordings of waveforms were first converted to digital domain by sampling them at a suitable rate. The sampling rate chosen must take into consideration the maximum frequency of interest and also satisfy the Nyquist sampling rate in order to avoid aliasing effects. The maximum frequency of interest being 360 Hz, the analog signal was first bandlimited to 360 Hz by an analog bandpass filter of passband range 0~360 Hz. The bandlimited signal was then sampled at a rate of 960 Hz using a 12 bit A/D converter.

87 SM 633-1 A paper recommended and approved by the IEEE Power System Relaying Committee of the IEEE Power Engineering Society for presentation at the IEEE/PES 1987 Summer Meeting, San Francisco, California, July 12 - 17, 1987. Manuscript submitted January 29, 1987; made available for printing April 28, 1987.

In order to extract the specific frequency of interest, the sampled data was further processed by filtering with two separate multi-bandpass linear phase Finite Impulse Response (FIR) digital filters - one for filtering the 'in-between' harmonics and the other for filtering the harmonic frequencies. The Parks - McClellan design of linear phase FIR digital filters was adopted to design the two digital filters. A linear phase filter is necessary to preserve the phase characteristics of the analog signals. The frequency response of these filters is shown in Figure 1a for the 'in-between' harmonics and Figure 1b for the harmonic frequencies. A passband width of 10 Hz about the center frequency was used for each passband in the digital filter design.

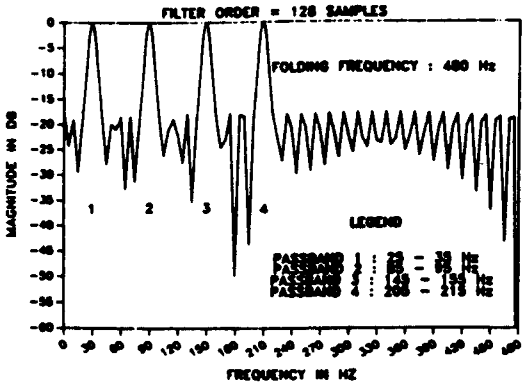


Figure 1a. Frequency response of in-between harmonic digital filter.

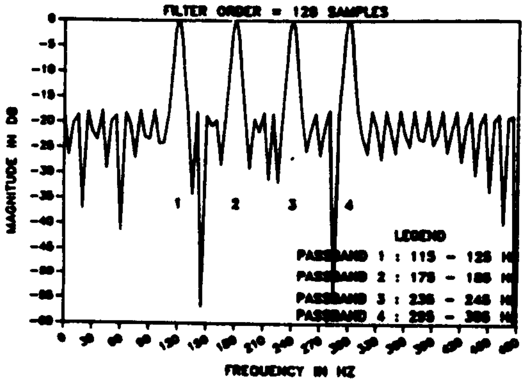


Figure 1b. Frequency response of harmonic digital filter.

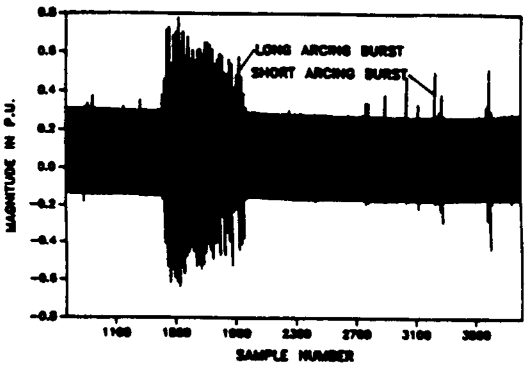


Figure 2. Arcing fault waveform.

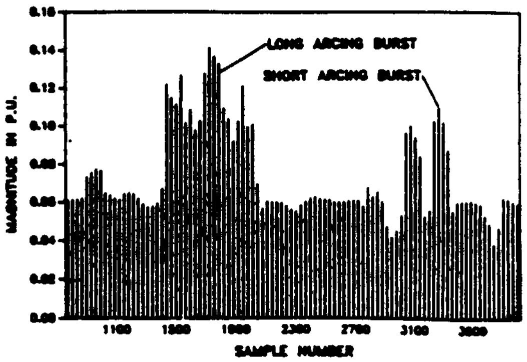


Figure 3a. Variation of 90 Hz spectrum during arcing fault.

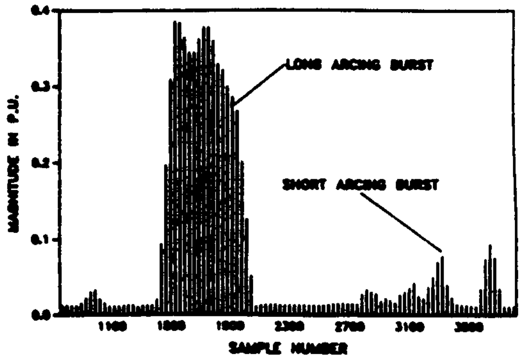


Figure 3b. Variation of 180 Hz spectrum during arcing fault.

Next, the frequency spectra of the filtered data was obtained by performing a Fast Fourier Transform of the filter output. The time domain signal (sample amplitude vs. sample number) was reconstructed from the sampled data and plotted as shown in Figure 2. The frequency spectra of interest were also plotted as a function of time in order to determine their magnitude variation. Examples are shown in Figure 3a for 90 Hz component and Figure 3b for the 180 Hz component. The start and end of an arcing 'burst' can be determined from the frequency plots by comparing them with the corresponding time domain waveform. This procedure was repeated in order to obtain the variation of each frequency spectrum for all of the seven events that were investigated.

DATA ANALYSIS

In order to analyse the data, bar plots were created for each event indicating the change in magnitude of the spectra from pre-event to an event condition. These were contrasted to the maximum values attained by the spectra during pre-event and event conditions. The magnitudes shown reflect the average of the values attained by the spectra during several identical tests conducted for each event.

Figure 4 shows the relative change in average magnitude of several 'in-between' harmonic and harmonic frequencies for arcing tests performed during switching operations. The plot indicates that all the frequencies show an increase of

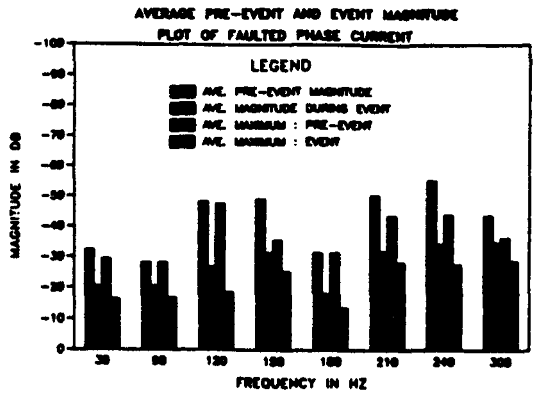


Figure 4. Change in spectral magnitude during switching conditions.

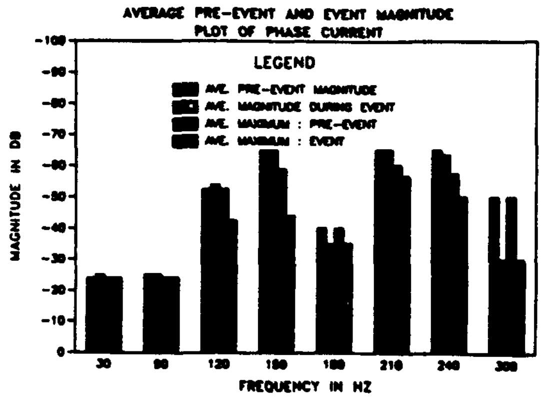


Figure 5. Change in spectral magnitude during capacitor bank operation.

at least 10 db in magnitude under arcing fault conditions. The 150 Hz and 210 Hz components indicate the maximum change for the 'in-between' harmonics (~ 20 db). All the harmonic frequencies show an increase of at least 15 db under fault conditions. The 120 Hz and 240 Hz components indicate the maximum change for harmonic frequencies (~ 25 db).

Figure 5 shows the behaviour of the low frequency spectra for a capacitor bank operation. It is observed that the 'in-between' harmonic frequencies remain fairly constant during the switching operation whereas the harmonic frequencies show a greater change in magnitude. This is especially true in the case of 180 Hz and the 300 Hz component. Figure 6a shows the variation of the third harmonic during a capacitor bank operation. Figure 6b shows the variation of the fifth harmonic for the same operation. It is observed from these figures that both the 180 Hz and the 300 Hz spectra show a step increase in magnitude which persists the entire duration the capacitor bank is switched on. On the other hand, the 'in-between' harmonics do not indicate such phenomena. Their magnitude change is more of an impulse nature lasting for a short duration of time due to the transients of the switching operation itself. In the case of other switching operations such as an air switch or a load tap changer, such anomolies were not observed. Figure 7 shows the behaviour of the low frequency spectra for a load tap changer operation and Figure 8 for an air switch operation.

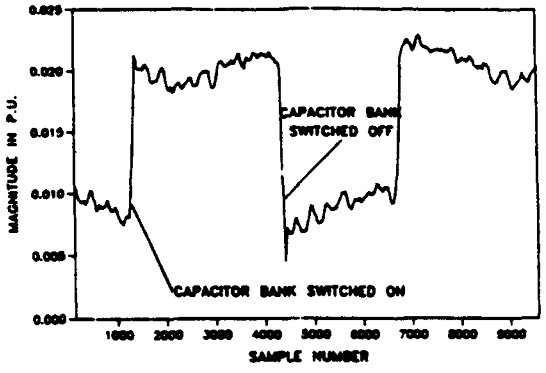


Figure 6a. Variation of 180 Hz spectrum for capacitor bank operation.

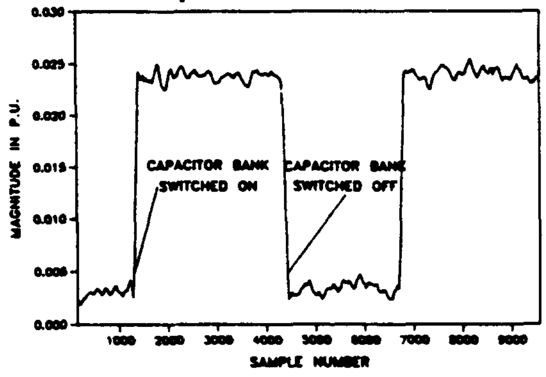


Figure 6b. Variation of 300 Hz spectrum for capacitor bank operation.

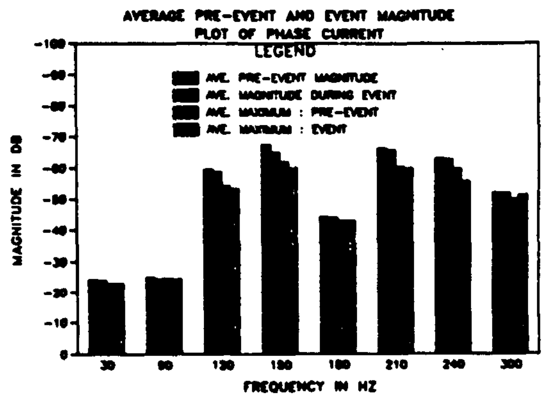


Figure 7. Behavior of low frequency spectra for load tap changer operation.

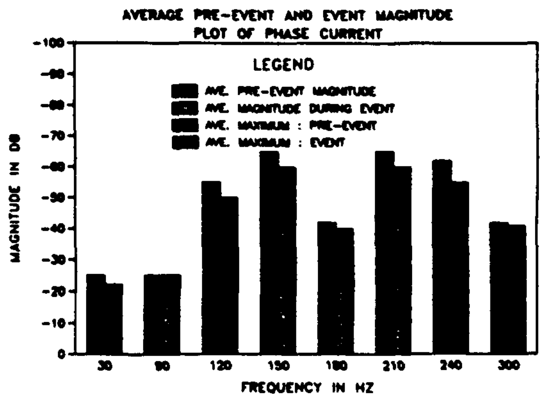


Figure 8. Behavior of low frequency spectra for air switch operation.

STATISTICAL ANALYSIS

A statistical analysis was performed to investigate the dependency of the low frequency spectra magnitude upon arcing burst duration and soil conditions. Arcing fault data was obtained from three separate test locations representing different soil conditions. In test site 1 the arcing tests were conducted on wet soil, in site 2 they were conducted on dry soil and on sandy soil in site 3. Each site data record was then subdivided into at least three different clusters, each cluster comprising arcing bursts of approximately the same burst duration. The different arcing burst durations typically considered comprised 'short' duration bursts (1 ~ 3 cycles), 'medium' duration bursts (5 ~ 20 cycles) and 'long' duration bursts (> 20 cycles). At least 50 different arcing events were considered to comprise a single cluster making a total of at least 150 events investigated at each site location.

Magnitude Evaluation

A frequency domain analysis was performed for the individual events in a cluster to determine the relative change in magnitude of the various spectra from pre-fault to fault conditions. Next, the mean and standard deviation of the magnitude change was determined for each frequency spectrum by averaging over all the 50 odd events in the cluster and determining the variation about the mean. This procedure was repeated for each of the clusters at the three test site locations and the statistics indicated by bar plots. Figure 9a shows the average relative increase in magnitude of the spectra due to short duration arcing bursts on wet soil conditions. The standard deviation indicates the dispersion of the magnitude about the mean for short arcing bursts. It is seen that the 'in-between' harmonic frequencies indicate a large dynamic change in magnitude (20 ~ 50 times greater) under arcing fault conditions. However, this change is very random as indicated by the large standard deviation. This means that even though the 'in-between' harmonics show a large percentage increase in magnitude, the dispersion about the mean is also large thereby indicating that the precise amount of increase is unpredictable. The harmonic frequencies however show a reliable change in average magnitude as indicated by their smaller value of standard deviation. This suggests that the relative change in magnitude is consistent for the different events comprising the cluster.

Figure 9b shows the relative change in magnitude of spectra for medium duration bursts on wet soil conditions. In this case, it is seen that the change in magnitude of the 'in-between' harmonics is more reliable as indicated by their low value of standard deviation. Figures 10a & b show similar plots for arcing faults on dry soil conditions. Again, the 'in-between' harmonics show randomness for short arcing bursts as compared to medium duration bursts. Figures 11a &b show the relative changes in spectra magnitude when arcing tests were conducted on sandy soil.

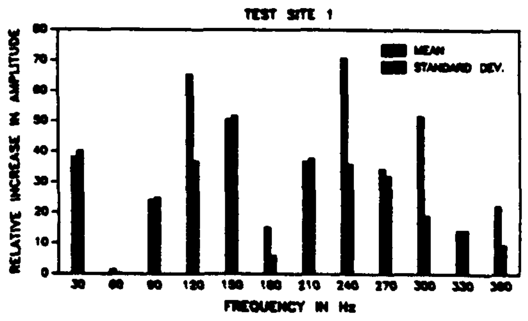


Figure 9a. Statistics of short duration bursts on wet soil.

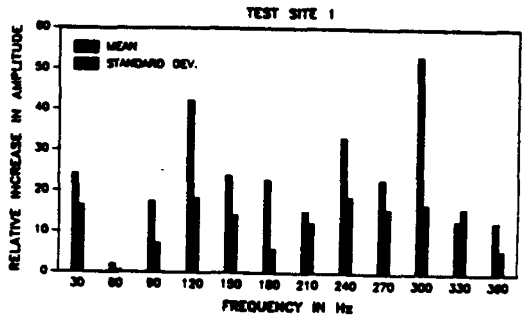


Figure 9b. Statistics of medium duration bursts on wet soil.

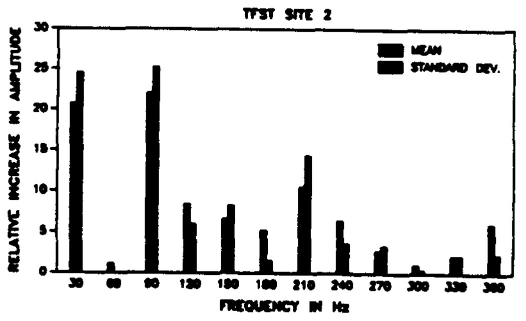


Figure 10a. Statistics of short duration bursts on dry soil.

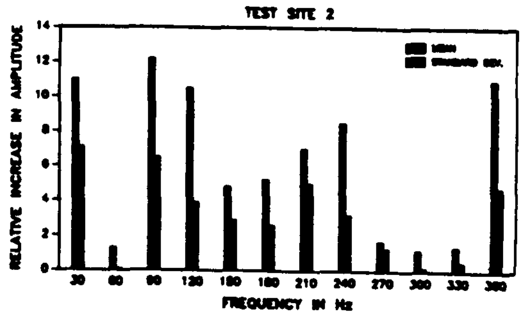


Figure 10b. Statistics of medium duration bursts on dry soil.

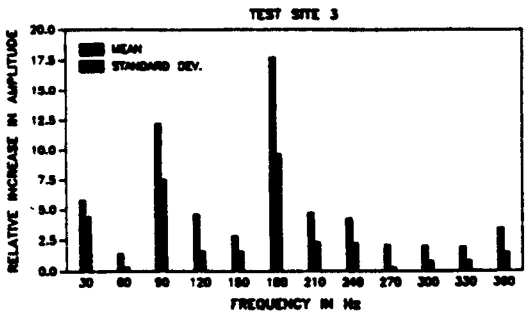


Figure 11a. Statistics of medium duration bursts on sandy soil.

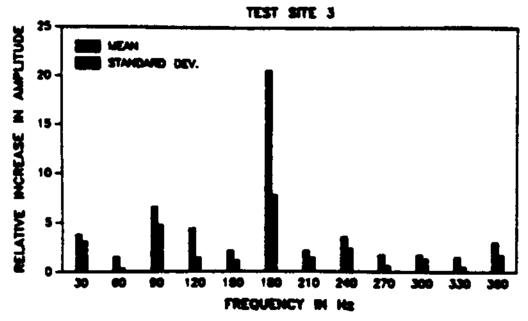


Figure 11b. Statistics of long duration bursts on sandy soil.

Summarizing, it can be said that the 'in-between' harmonic frequencies are random noise spectra which show a large relative change in magnitude under arcing fault conditions. The change in magnitude is more predictable for medium and long duration arcing bursts as compared to short bursts. The harmonic frequencies show a more consistent change in magnitude without respect to burst duration.

Results of Correlation by soil type

The dependency of arcing burst duration on soil conditions was also investigated. Figure 12a shows the distribution of arcing burst duration on wet soil conditions. It is seen that a large percentage of the events comprised short arcing bursts typically 2 or 3 cycles in duration. Figure 12b shows the distribution of arcing burst duration on dry soil conditions. On

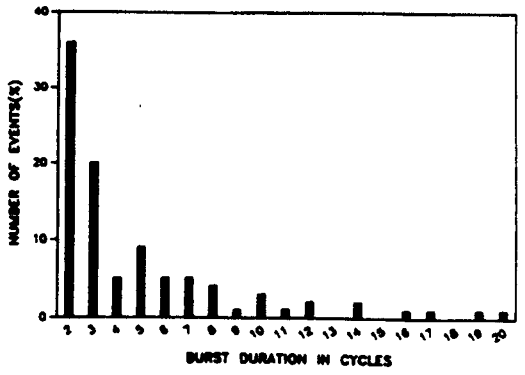


Figure 12a. Distribution of burst duration on wet soil.

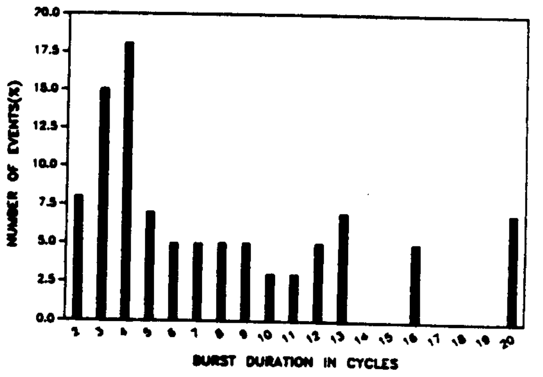


Figure 12b. Distribution of burst duration on dry soil.

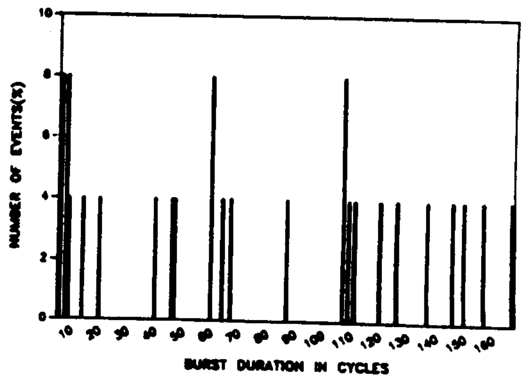


Figure 12c. Distribution of burst duration on sandy soil.

Table 1. Statistics of low frequency spectra.

TEST SITE	Ft.Worth, Texas			Kluge, Texas			Embudo, New Mexico		
SOIL TYPE	wet soil			dry soil			sandy soil		
BURST LENGTH	short			medium			long		
OFF-DURATION	short			medium			long		
MEAN and STANDARD DEVIATION of magnitude st.dev inside ( )									
	2-4 cycles	5-10 cycles	11- cycles	2-6 cycles	7-13 cycles	15- cycles	6-47 cycles	60-120cycles	130- cycles
60 Hz	1.4(0.2)	1.7(0.3)	2.0(0.5)	1.1(0.0)	1.3(0.1)	1.3(0.1)	1.4(0.3)	1.5(0.3)	2.0(0.3)
120 Hz	65.3(36.7)	47.7(29.4)	42.1(18.2)	8.4(6.0)	12.4(12.2)	10.5(3.9)	4.6(1.6)	4.4(1.4)	6.3(1.2)
180 Hz	15.3(5.9)	21.4(7.6)	22.6(5.8)	5.2(1.5)	6.3(1.7)	5.2(2.6)	17.7(9.6)	20.6(7.7)	32.3(7.4)
240 Hz	70.8(35.9)	43.6(25.7)	33.2(18.6)	6.4(3.6)	8.3(3.9)	8.5(3.2)	4.2(2.2)	3.5(2.3)	6.0(1.9)
300 Hz	51.7(19.1)	58.9(20.2)	53.2(16.8)	1.0(1.9)	1.9(2.2)	1.2(0.2)	2.0(0.8)	1.7(1.3)	3.0(1.4)
360 Hz	22.1(9.4)	17.5(8.4)	12.6(5.7)	6.0(2.3)	9.8(5.2)	11.0(4.8)	3.4(1.5)	3.0(1.7)	4.3(1.9)
30 Hz	38.7(40.1)	25.3(29.2)	24.0(16.4)	20.7(24.5)	13.4(11.2)	11.0(7.1)	5.8(4.4)	3.8(3.1)	5.0(4.0)
90 Hz	24.2(24.9)	15.5(18.0)	17.3(7.1)	22.0(25:2)	15.0(12.2)	12.2(6.5)	12.2(7.5)	6.6(4.8)	7.7(6.9)
150 Hz	50.5(51.6)	26.0(27.1)	23.6(14.0)	6.6(8.2)	3.7(4.3)	4.8(2.9)	2.8(1.6)	2.1(1.1)	3.3(1.7)
210 Hz	36.7(37.7)	20.0(23.7)	15.0(12.2)	10.5(14.3)	8.7(5.1)	7.0(5.0)	4.7(2.3)	2.1(1.4)	2.8(1.3)
270 Hz	34.4(32.1)	19.2(16.8)	22.8(15.6)	2.7(3.3)	2.3(2.2)	1.7(1.3)	2.1(0.3)	1.7(0.6)	1.7(1.0)
330 Hz	14.2(8.8)	8.8(9.5)	12.9(15.9)	2.1(2.1)	2.0(1.3)	1.4(0.5)	1.9(0.8)	1.5(0.5)	1.6(0.7)

dry soil, the arcing usually lasted for 4 to 20 cycles and most of the events fell in this category. Figure 12c shows the distribution on sandy soil condition. Here it is observed that the arcing persists longer, usually greater than 20 cycles in duration. These plots indicate that on wet soil conditions, arcing is of very short duration. On dry soil, the arcing persists longer and these type of bursts can be classified as medium duration bursts. On sandy soil, the bursts are of very long duration and can be categorised as long duration bursts. The statistics of the relative magnitude changes on the three soil conditions are indicated in Table 1.

The period duration between successive arcing bursts was also investigated on the three different soil types and their distribution plotted as a function of the soil type. Figure 13a shows the distribution of the interval between successive arcing bursts on wet soil. On comparison with Figure 12a, it is observed that the interval between bursts has a distribution similar to the distribution of the burst duration itself. This indicates that on wet soil, the arcing bursts are mostly of short duration with short intervals between successive bursts. Figure 13b shows the distribution of the 'off-interval' between successive arcing bursts on dry soil. Again, on comparision with Figure 12b, it is observed that the 'off-intervals' have a distribution similar to the burst duration, i.e. on dry soil conditions, the arcing is of medium duration (4 ~ 20 cycles) separated by 'off-intervals' of similar duration. Figure 13c shows the distribution of the 'off-interval' between successive bursts on sandy soil. On comparision with Figure 12c, it is seen that the 'off-intervals' show a distribution similar to the arcing duration on sandy soil.

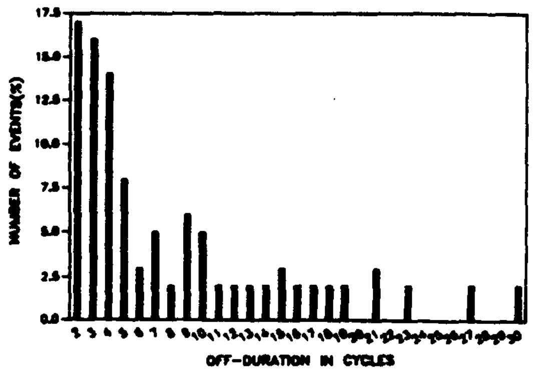


Figure 13a. Distribution of interval between successive bursts on wet soil.

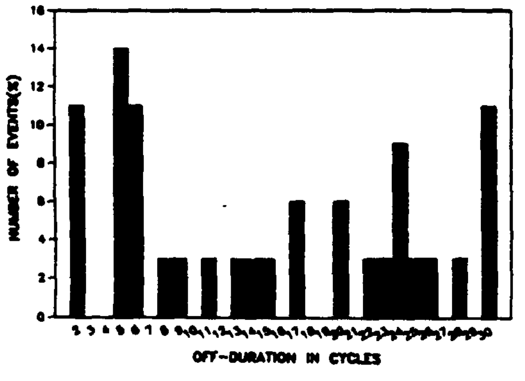


Figure 13b. Distribution of interval between successive bursts on dry soil.

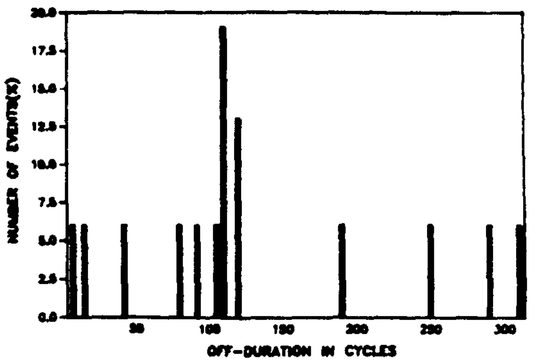


Figure 13c. Distribution of interval between successive bursts on sandy soil.

Finally, the dependency of the magnitude of the 'in-between' harmonics on soil conditions was investigated. The relative magnitude change of several 'in-between' spectra at each site was plotted as shown in Figure 14. As seen from the figure, the magnitude of 'in-between' spectra depends on soil type. The change in magnitude is higher on wet soil as compared to dry or sandy soil. This is possibly due to the initial higher conductivity of wet soil. The magnitude dependency of harmonic frequencies on soil type were not investigated because these frequencies were shown to depend and vary as a function of other factors including system impedance and loading.

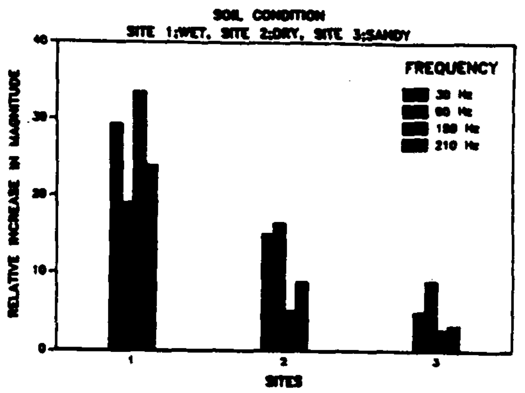


Figure 14. Dependency of spectral magnitude on soil type.

RESULTS QUALIFICATION

The data presented has proved of great value to TAMU researchers investigating high impedance faults. The nature of these faults is more clearly understood from this data analysis. Since several other research teams are studying the problem using magnitude, phase and time domain characteristics at low frequencies, it was felt that the data should be made available. The data comes from the TAMU database, probably the most comprehensive in existence. However, we are constantly concerned that more data and data analysis for different fault scenarios and feeder conditions will result in modified conclusions or results. Those using this data should recognize it is statistical and only absolutely valid for the specific faults that were studied. In spite of this, we believe the results are generally true for many faults and do give an insight into fault behaviour.

CONCLUSION

The behaviour of several low frequency spectra was investigated for arcing faults and normal switching events. It was observed that the 'in-between' harmonic frequencies can be used to discriminate the presence of arcing faults on distribution feeders. More important, these frequencies can be used to distinguish arcing faults from capacitor bank switching operations. The harmonic frequencies indicate reliable increase in magnitude during arcing fault conditions. However, they are not immune to switching and capacitor bank operations.

Statistical analysis was performed to study the behaviour of the low frequency spectra on different soil conditions. It was found that the magnitude of 'in-between' harmonics depend on burst duration and soil condition. The change in magnitude was found to be higher for short duration arcing bursts and on wet soil conditions. Finally, the distribution of arcing burst duration and the interval between successive arcing bursts was determined on different soil conditions. It was observed that the arcing bursts were mostly of short duration seperated by short intervals of inactivity on wet soil, of medium duration separated by similar intervals of inactivity on dry soil and of long duration separated by long intervals of inactivity on sandy soil.

ACKNOWLEDGEMENT

The Electric Power Institute at Texas A&M wishes to acknowledge the financial support of numerous electric utilities, Dow Chemical, and 3M Corporation without which this research would not be possible.

REFERENCES

1. A.G.Phadke, He Hankun, "Detection of Broken Distribution Conductors", Proceedings of IEEE Southeast Conference, Raleigh, N. Carolina, Paper No. CH2161-8/85/0000-0074.
2. S.J.Balser, K.A.Clements, D.J.Lawrence, " A Microprocessor-based Technique for Detection of High Impedance Faults ", Paper No. 86 WM 155-6. IEEE/PES WPM,1986.
3. H.L.Graham, A.J.Carlson, T.A.Granberg, "Broken Conductor and High Impedance Fault Detection by High Frequency Impedance Monitoring", Paper No. A-80-06406. IEEE WPM,1980.
4. B.M.Aucoin, B.D.Russell, "Distribution High Impedance Fault Detection Utilising High Frequency Current Components", IEEE Transmission and Distribution Conference, Minneapolis, Paper No. 81-TD 603-0.
5. B.M.Aucoin, B.D.Russell, "Detection of Distribution High- Impedance Faults Using Burst Noise Signals Near 60 Hz", IEEE Transactions on Power Delivery, 86T&D 546-6.
6. H.Calhoun, M.T.Bishop, C.H.Eichler, "Development and Testing of an Electro-Mechanical Relay to Detect Fallen Distribution Conductors", IEEE Transactions on Power Apparatus and Systems, vol.PAS-101, No.6 June 1982.
7. R.E.Lee, M.T.Bishop, "Performance Testing of the Ratio Ground Relay on a Four-Wire Distribution Feeder", IEEE Transactions on Power Apparatus and Systems, vol.PAS-102, No.9 Sep. 1983.

BIOGRAPHY

B. Don Russell, (SM) received B.S. and M.E. degrees in Electrical Engineering at Texas A&M University. He holds a Ph.D. from the University of Oklahoma in power systems engineering.
Dr. Russell is Associate Professor of Electrical Engineering and directs the activities of the Power Systems Automation Laboratory of the Electric Power Institute, Texas A&M University. His research centers on the use of advanced technologies to solve problems in power system control, protection, and monitoring. He is the recipient of several awards for advanced technology applications. Dr. Russell chairs several working groups and subcommittees of PES.
Dr. Russell is a member of the Substation Committee and Power Engineering Education Committees of PES. He is a registered professional engineer and a director of the Texas Society of Professional Engineers.

Ram Chinchali, (S'84) received B.E. (Hons) degree from the University of Madras, India in 1981 and a M.E. in Electrical Engineering from Texas A&M University in 1984. He is currently working toward his Ph.D. degree.
Mr. Chinchali's research interests are microprocessor applications in power system control and protection. During the period between 1981 and 1983, he was a Relay Engineer at English Electric Co., India where he developed protection schemes for 220 KV and 400 KV substations.

C. J. Kim , (S'86) received his B.S.E.E. and M.S.E.E. degrees from Seoul National University, Korea in 1980, and 1982 respectively. During 1983-1985, he worked at the Gold Star Instrument Research Center in computerised process control systems.
He is presently a research assistant in the Department of Electrical Engineering at Texas A&M University.

A DIGITAL PROTECTION SYSTEM INCORPORATING REAL TIME EXPERT SYSTEMS METHODOLOGY

by

B. DON RUSSELL
Professor

KAREN WATSON
Assistant Professor

Texas A & M University
U. S. A.

ABSTRACT

This paper describes a digital system architecture for implementing protection, monitoring, and diagnostic algorithms on distribution feeders. The functional requirements of this system are described including the requirements for implementation of real-time expert system programming. The hardware and database design is described in detail and software considerations are presented. The advantages of this architecture are enumerated.

Keywords: Protective Relaying, Expert Systems, Digital Architecture, High Impedance Faults, Adaptive Relaying

1. INTRODUCTION

Existing practices in distribution system protection and diagnostics call for the use of dedicated relaying devices, typically using preset parametric thresholds, to determine the health and state of the power system. It has long been known that certain power system faults and abnormal conditions cannot be detected by these protection equipments and methodologies [1,2]. Recent research has shown that knowledge based systems are capable of significantly improving the detection and diagnosis of certain distribution faults and abnormal conditions [3,4].

Conventional automation, control, and protection equipments for distribution systems are passive/responsive. These devices typically respond to changes in the power system as measured by fixed thresholds and preset limits. The systems are designed to prevent catastrophic failure or incorrect operation and work well for many protection, data acquisition, and supervisory control functions. Commercial products have recently become available which use digital technology and offer conventional protection and data acquisition coupled with the advantages of improved communications. In spite of these recent hardware improvements and the use of digital technology, many advanced protection and diagnostic algorithms cannot be implemented on present hardware, for example, advanced detection methods for low current, high impedance faults. [5]. Analytical routines capable of diagnosing incipient fault conditions and equipment failures have processing and data requirements that cannot be provided by conventional hardware architectures. [6].

Additionally, various "adaptive" relaying concepts have been proposed for distribution protection which incorporate knowledge based system programming and the use of nonelectrical parameter inputs. These concepts require a unique architectural environment for implementation.

The purpose of this paper is to describe an architecture suitable for implementing the simplest as well as the most complex protection algorithms for distribution feeders. Conventional protection methods would be maintained, but significantly enhanced using adaptive relaying concepts. The system must also be capable of supporting the data requirements of various diagnostic routines whose purpose is to analyze and discriminate operating conditions on the distribution feeder which are abnormal, but not catastrophic. For example, a downed conductor which draws little current can sometimes be detected, but only after significant data analysis using advanced algorithms. This may mean the analysis of data at other than fundamental and harmonic frequencies. The architecture must be capable of supporting these input data requirements. [7].

In summary, the architecture must support conventional detection algorithms, incipient fault diagnostic algorithms, adaptive relaying concepts, and high impedance/arcing fault detection algorithms. It also must be capable of supporting knowledge based programming, must efficiently store and recall historical data files, and must be capable of dynamic program modification based on changing input data scenarios. It also must be inexpensive!

Given these relatively harsh architectural requirements, it was decided that a "real-time" expert system methodology would be used in an appropriate parallel processing hardware architecture. Most "expert systems" are not intended for closed-loop operation in real time. However, it has been shown that modified expert operating systems can be successfully used in real-time fault detection and diagnosis without the conventional expert system requirements of human interaction and guidance. [8].

The following is a description of the architecture and related software considerations that are presently under research at Texas A&M University.

Presented at the CIGRE Bournemouth Symposium, June, 1989.

2. FUNCTIONAL OVERVIEW AND ARCHITECTURAL DESCRIPTION

The architecture being presented describes a system which monitors one three phase distribution feeder. However, one of the primary goals has been to allow for flexibility in the system. Therefore, the system has been designed so it can easily be adapted to handle more than one feeder. This kind of flexibility is found not only at the input but also is found throughout the system. Figure 1 shows the primary functions involved in the protection system. These functions include the Signal Conditioning and Conversion Unit (SCC Unit) where the voltages and currents from each phase of the feeder and other inputs are sensed, filtered, amplified or otherwise conditioned as appropriate. These signals are then converted from analog to 12-bit digital signals before being packaged in blocks, then serialized and converted to optical media. The optical signals are then passed across optical fibers which serve primarily to isolate the computing equipment downstream. This also allows the input units to serve more than one feeder, if desired. This unit prepares the sensor data for a number of parametric checks. There are open slots in the system for more analog treatment of the signals if desired.

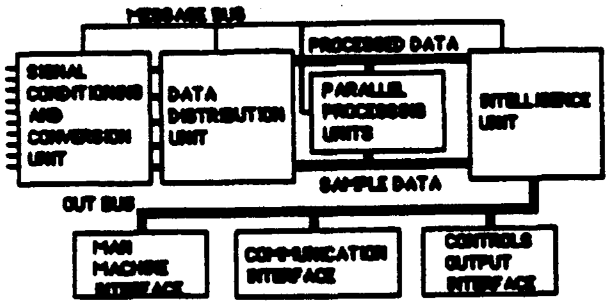


Figure 1. System Automation

The next unit, the Data Distribution Unit, converts the optical signals back into electrical pulses, deserializes and reblocks the data. This data is then distributed to the short term memory of all the Processing Units across a 16-bit data bus referred to as the Sample Data Bus. This data contains all the information, sampled data, amplifier gains and time tags necessary for conversion to engineering units. A block of memory to provide medium term memory can easily be inserted here to store all of the raw data for a short time frame (approximately 800 k bytes of memory per second of raw data is required). Without this memory, the Processing Units have all the data in their short term memory and must determine what needs to be retained. There are two types of processing units and multiple combinations of the two types are possible. No hard upper bound has been encountered for the number of processing units to be utilized, however, the message bus which allows communication between all the units must be carefully studied if more than 10 units are conversing heavily. The Processing Units do further parametric derivations and algorithmic checks. Here signals such as the high frequencies, odd harmonics or even harmonics, are monitored for changes characteristic of high impedance and incipient faults. In addition, traditional overcurrent detection is also implemented in the Processing Units.

Following the Processing Units is the Intelligence Unit. This unit can access the raw data, but usually works with the processed data coming out of the process units on the 16-bit Processed Data Bus. This unit handles the diagnostic decisions and the long term memory maintenance. The unit is necessary in order to combine and weigh as many detection algorithms as appropriate for the monitored feeder. The unit over time will learn to recognize more effectively what is normal and what is faulty for a given feeder. The Intelligence Unit controls the serial Message Bus which interconnects all the units of the system. Finally, the Control Output Units, Man-Machine Interface and Communication Interface all couple into the Intelligence Unit by the Out Bus. Each of the units mentioned will be described in more detail in a section to follow, but first the data flow structure will be presented.

2.1 Data Flow Structure

Standard devices such as transformers for voltage and current provide signals to the Signal Conditioning Unit. In this unit, analog filters are used. Low pass filters for antialiasing are used on all voltages and currents and these signals are passed to their individual amplifiers. In addition, the current signals are passed through a notch filter which eliminates the fundamental frequency of the power line and these signals are then passed to their amplifiers. Also, the notched signal is passed to a high pass filter and the high frequency current information coming out of these filters are passed to their own amplifiers.

The amplifiers all have variable gain; we are currently working with four discrete settings for each amplifier. The four settings for antialaised signals will be different from the gains for the notched signals, which are different from the high frequency gains. Each amplifier's gain setting is individually controlled by a data acquiring processor found in the SCC Unit. These processors also control the sampling time of the analog to digital converters which follow the amplifiers. The A/D converters sample and hold all voltage and currents from the amplifiers. These samples are multiplexed through the A/D converters with 12-bit resolution. The same processor mentioned above gathers these digital signals and adds 4-bits to each. Two bits are an identification tag for the particular signal type (antialaised, notched, high frequency) and two bits indicate the amplifier setting being used during the sampling. As the samples are collected they are converted to serial signals and transformed for optical transmission.

At the other end of the optical fiber, the data is converted back to electrical signals. A Data Receiving Processor packages the data in a block containing the digital data on all the voltages and the variety of filtered currents as well as their gains, a real-time tag, and a quality of data word. The quality of data word indicates if the data made it through the optical conversions and transmission clean. Now the Data Distributor Processor controls the distribution of this data to all the processing elements short term memory. Each processing unit has a two port memory unit which can only retain about 19 sets of all the sampled data values, thus we refer to it as short term memory.

The Processing Units will do signal processing and other numerical manipulations on the data. The number of processing units required is controlled by the responsiveness desired and the number of algorithmic tests being made. Examples of signal processing functions which may be performed are:

- finding the energy found in a discrete frequency (e.g. a harmonic)
- finding the energy found in a set of frequencies (e.g. all high frequency)
- finding the Fourier transform of a given signal

The parameters computed by the processing units are regathered by the Parameter Distributor Processor which controls the Processed Data Bus. This processor directs the parameters to other processing units and the Intelligence Units. The Numerical Processing units will normally be programmed with algorithms that watch for changes of significance in either the raw data or the parametric data. These numerical processors have the ability to be tuned for various levels of numerical processing (e.g. floating point coprocessors can be easily added).

After the multiple signals and parameters have been calculated and watched for significant changes or trends, the Processing Units pass message flags as well as significant data to the Intelligence Unit. This unit gathers the flags from all the many algorithms being checked and the corresponding data. In the Intelligence Unit these flags are checked to see if a conclusive diagnostic of the feeder can be made. If there is no conclusive diagnostic, the system checks its long term memory which has stored the representative data of the various significant states encountered by the feeder. The current state is tested against these previously encountered states. If there is a close match the system assumes that this is the state of the feeder and then attempts to prove it. The work of proving an assumed state may involve messages being sent from the Intelligence Unit to the rest of the units. Examples of this include a change of gain on the amplifiers, a change of filtering on a signal processor, a change of threshold on a parameter check, or change in a time window for watching a trend. If, on the other hand, the current state seems distinct from any previously encountered state, it may be recorded as a new, but mysterious state. If the machine never resolves the state of the system, a message will be prepared for the man-machine interface. Thus human expertise will be called in to help classify the unusual state encountered.

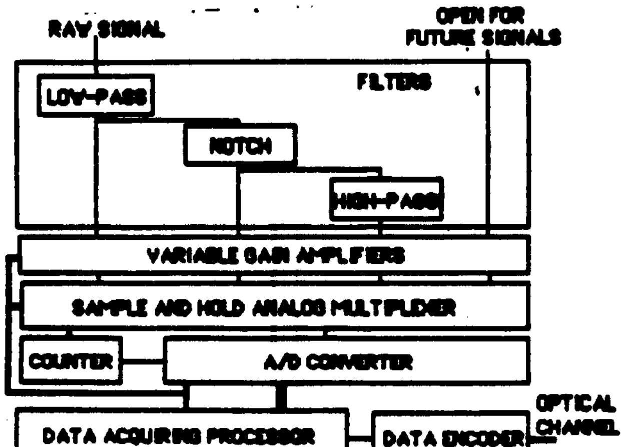


Figure 2. One channel of Signal Conditioning and Conversion Unit.

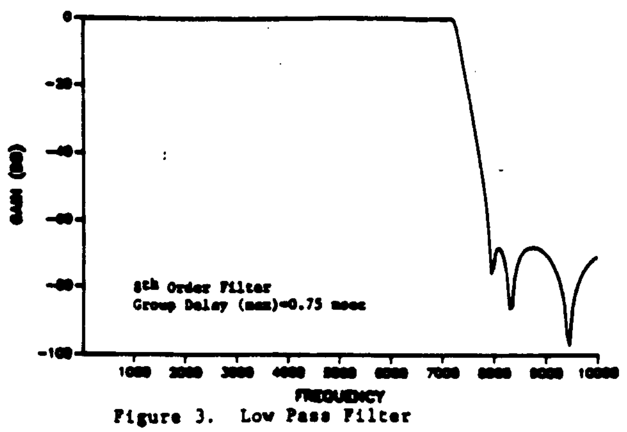


Figure 3. Low Pass Filter

3. THE SIGNAL CONDITIONING AND CONVERSION UNIT (SCC UNIT)

The unit described here represents one channel of the SCC Unit. For isolation purposes a channel will handle either voltages or the signal of one phase current. We have 5 channels in the SCC Unit (Fig. 2).

For proper detection of a fault condition, the input "raw" signal must be preprocessed. This is accomplished by the following filtering scheme:

- Low-Pass filtering at 5400 Hz to remove high frequency content. This signal is used directly to test for overcurrent faults.
- Notch filtering at 60 Hz to remove the fundamental. This signal is used to test harmonic indication of high impedance faults.
- High-Pass filtering at 2000 Hz (having an effective Band-Pass between 2000 Hz and 5400 Hz) to remove the low frequency harmonics. This is used to complement the above tests for high impedance faults.

Figures 3, 4, and 5 show the characteristics of each filter.

To implement a digitally controlled variable gain amplifier, a Multiplying Digital to Analog Converter is used as resistance in both the feedback path and the input to the configuration shown in Figure 6. The gain will be determined by the ratio of resistance once the digital inputs have been set by the controlling microprocessor.

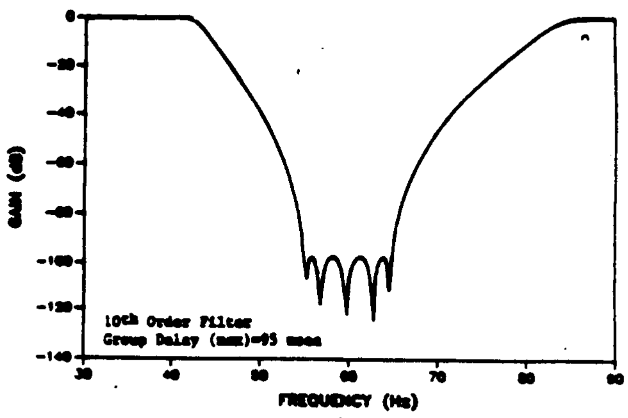


Figure 4. Notch Filter

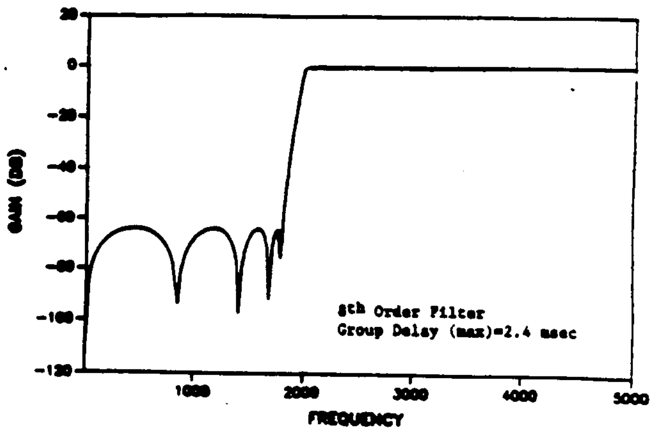
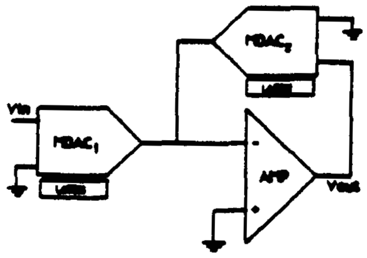


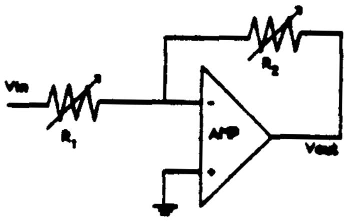
Figure 5. High Pass Filter

ORIGINAL PAGE IS
OF POOR QUALITY

ORIGINAL PAGE IS
OF POOR QUALITY



a) Circuit



b) Equivalent Circuit

Figure 6. Variable Gain Amplifier

The sample and hold multiplexer accepts 4 analog inputs and multiplexes them onto 1 analog output. A new sample of all signals is taken concurrently every 66 μ seconds (~15 kHz). A two bit counter then provides the multiplexer the address of the one out of four sampled inputs which should be passed to the A/D converter. When the output of the A/D converter is latched a signal is provided, from the A/D, which allows the address of the multiplexer to be incremented. This is repeated until all four inputs have each passed to the A/D converters.

The A/D Converter continuously convert up to 1 million samples per second if an 8 mHz clock is used. A CMOS buffer is used with this chip to help reduce the cross talk between the digital and analog parts of the circuit.

The CPU, referred to as a Data Acquiring Processor (DAP), for this unit is an 8 bit microcontroller with a serial communication controller capable of transmitting and receiving data up to 2.4 M bits per second. Therefore, the DAP will transmit the gain of the 4 signals sampled by this channel and then the data from the A/D converter. A 2 bit identification code and parity bits have been appended to the 12 bit data which was generated by the A/D converter. The DAP controls the sequencing and latching of data from the sample and hold multiplexer through the A/D converter and into its communication controller. For serial transmission the DAP encodes the 8 bits of gain setting and 4 words of data into a CRC block of data in order to validate transmission quality at the receiving end. In addition, the DAP adjusts the gain of any signal which has exceeded the current conversion levels, (i.e. the gain is lowered if need be). However, the DAP waits on a message from the Intelligence Unit before it will resensitize the gains. This is to reduce gain jitter caused by noisy data. The DAP also controls the calibration of the sample and hold multiplexer and the A/D converter during start-up or testing times.

Finally, the serial data from the DAP passes through the coded data transceiver. This device receives and transmits serial data in Manchester encoding. It receives serial data from the DAP, encodes it and passes it to an optical transmitter. Any messages from the Intelligence unit pass on an independent optic fiber to an optical receiver and then into the data transceiver. Each channel of the SCC Unit is connected to 4 optic fibers. One fiber transmits sampled data and another fiber the clock signal for the transmission rate. The third fiber carries messages from the Intelligence Unit and the final fiber carries the message clock signal.

4. THE DATA DISTRIBUTION UNIT (DD UNIT)

The DD Unit has many components discussed in the SCC Unit. The optical transmitters and receivers, the transceiver for Manchester decoding/encoding, and the CPU, referred to as the Data Receiving Processor (DRP), operate in a similar, but reversed flow, to how they were operated in the SCC Unit. There are 5 channels whose data is received and checked by a DRP (Fig. 7). Then the data is stored in dual port memory. In addition to the 8 bits of gain information and four 16-bit words of sample data, parity and ID tags, 8 bits of information to indicate the transmission status (dirty or clean) for the incoming optical data has been placed in the dual port memory.

From the other port of the memory, a CPU, referred to as the Data Distributor Processor (DDP), maintains the system's real-time clock and puts a time tag on the recently received data from the 5 optic channels. This tag and the twenty-five 16-bit words from all the channels are transmitted to the Sample Data Bus. This is completed before the next set of data has been received (within 66 μ sec.). The DDP uses a Direct Memory Management Unit to facilitate the transfer of the data from the receiver memory to the short term memory of the Processing Units. Another CPU, referred to as the Parameter Distributor Processor (PDP), works in a very similar manner to distribute the parametric information from the processing units onto the Processed Data Bus.

5. THE PROCESSING UNITS (P Units)

The processing units utilize either a standard microprocessor for the Numerical Units or signal processor chip for the Signal Processing Units (Fig. 8). The Short Term Memory is a dual port memory written to the Sample Data Bus and read by the processor in the P Units. There is 1 k-byte of memory on each P Unit which contains no more than 19 blocks of sample data before it begins to overwrite on older data. The P Units have a variable amount of programming memory which could be RAM, PROM, or ROM memory. There is another 1 k-byte of dual port memory, referred to as the Parameter Memory, which can be written to by the processor in the P Unit. The PDP cpu found in the DD unit, distributes the information written to this parameter memory, via the Process Data Bus, to the appropriate P Units in a similar fashion to the DD units control of the Sample Data Bus. These parameters are usually computed on a per cycle basis so the data on this bus, although there may be many parameters, are generated much slower than the sampled data.

The Processing Units also have an interface and buffer for messages from the Intelligence Unit. The actual message passing scheme is handled much like an "ethernet" so that any P Unit, the DD Unit, or the Intelligence Unit may receive or generate a message. The SCC unit simply receives messages from this bus.

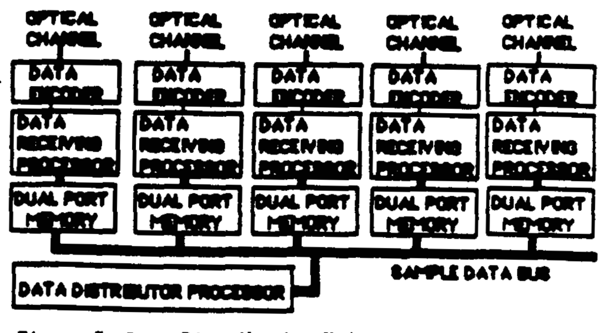


Figure 7. Data Distribution Unit

ORIGINAL PAGE IS
OF POOR QUALITY

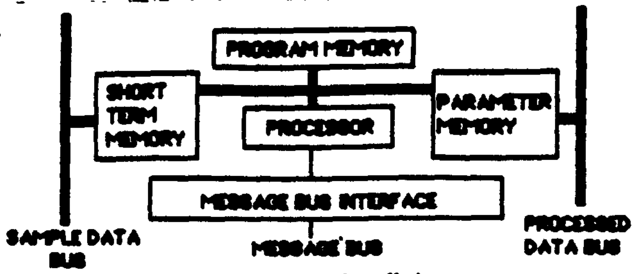


Figure 8. Processing Unit

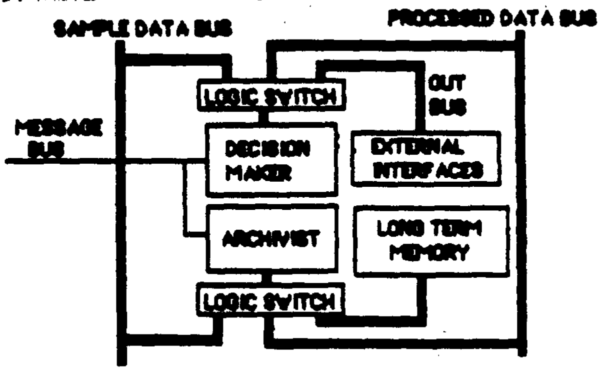


Figure 9. Intelligence Unit

6. INTELLIGENCE UNIT (I UNIT)

This unit involves a Decision Maker CPU and at this point a standard processor seems acceptable for the job (Fig. 9). The Decision Maker reads the flags, parameters, and data of the Message Bus, Processed Bus, and Sample Bus. The accumulated information is passed through a Knowledge Base Program to see if a conclusive diagnostic can be made. Even with a non-conclusive diagnostic, the Decision Maker makes a best guess of the state of the feeder. Based on this hypothesis, the long term memory is checked and messages for more sensitivity to the possible state are sent to all the other units in the system. The long term memory is maintained and updated by the Archivist. This is a CPU dedicated to pattern recognition and data base control. It monitors the Sample, Process and Message Buses and watches for new and unusual states. When such a state is encountered it is recorded with the final conclusion for diagnostics reached by the Decision Maker. Information on the new state will then pass to the man-machine interface whenever possible to seek confirmation. The Archivist will maintain ambient data (e.g. feeder servicing in progress, time of year and day, and weather conditions) along with the information about the feeder electrical state.

Now that the system units have been described the type and location of the system software will be presented.

7. THE SYSTEM SOFTWARE:

Initially, let us state that, if a simple algorithm was found to diagnose and characterize all feeder faults, then the many different units described could be collapsed into fewer units. However, we presently believe that several algorithms with numerous input parameters need to be monitored to achieve quality feeder protection.

In the SCC Unit, the software in the DAP is for data flow and conversion, gain setting, system calibration and testing, and message interpretation. In the DD Unit the DRP software handles data conversion and flow, quality checks, and message interpretation. The DD Unit DDP software gathers and distributes the sample data, maintains the real-time clock, initiates self testing for the system via the message bus, and interprets messages received.
In the Processing Unit different algorithms are stored. Examples include:

- overcurrent fault detection
- low frequency indicators for high impedance faults
- high frequency burst patterns for arcing faults
- pattern changes for incipient faults

These algorithms are stored in numerical and signal processing units as suitable. Normally, for fault tolerance, each algorithm is stored in more than one processing unit, even though it may only run on one at a time. Each Processing Unit will run its default set of algorithms unless a message from the Intelligence Unit indicates it should make a change. Based on the algorithm chosen, the appropriate sample data and parameter data will be used and, if the need to retain data is indicated by the algorithm or by a message from the Intelligence Unit the data is sent to the Archivist. In the Processing Units software includes algorithmic programs, communication controls, message generators and interpreters, and self-test modules.

In the Decision Maker of the Intelligence Unit there is a hierarchical Knowledge Base Program. The top level checks the incoming information and decides if immediate action is required (e.g. a flag indicating overcurrent). If action is required, a process is begun to initiate the action, both controls and communications, and to save pertinent information. If immediate action is not required, a hypothesis of the state of the feeder based on current information and recent trends is made. This hypothesis is checked against historical trends and diagnostic modifications are made, if necessary. The present information and historical trends are also used to develop a confidence level and uncertainty value for the proposed hypothesis. As confidence grows, the hypothesis becomes much more specific (e.g. not only do we suspect an occurence, but "it appears to be an arcing fault on phase B"). The Decision Maker software must gather data and process it through a knowledge base, initiate actions on the interfaces, generate and interpret messages, run and interpret system self-test results, and accept and direct new programs or information from the man-machine interface. This last function means that through the Decision Maker the software in other units may be modified.

The Archivist in the Intelligence Unit actively watches incoming information in order to classify it with a known, previously encountered state. If the information cannot be matched, a new state is recorded. The Archivist will respond to inquiries from the Decision Maker to find Historical Data to support or deny the hypothesis. This software represents an Intelligent Data Base system. The intelligence is in the ability to add updates from the data seen as well as respond to requests for data base information.

CONCLUSION

The sophistication of fault detection and diagnosis algorithms proposed for use on distribution feeders precludes the use of conventional protection system architectures. The data requirements are significant including the need for heavily processed, nonfundamental frequency data taken from fault spectra. This data is used in detection and diagnosis algorithms which impliment signal processing routines with heavy processsing requirements.

The proposed architecture uses a highly flexible front-end which can meet stringent specifications for data in both the time and frequency domain. The data is distributed to the various intelligence processing units as required. This data acquisition and distribution is adaptive to changing data requirements in the processing units.

The intelligence processing section is also highly flexible and capable of implimenting not only conventional protection routines, but newly proposed detection and diagnostic methods which require historical data bases for decision making.

The intent of this architecture is to provide a versatile hardware implimentation that can meet all the data requirements and processing requirements of programming routines using classical programming and expert system programming methods.

It is expected that this architecture will facilitate the integration of advanced protection systems with other substation functions and communications systems. It is also anticipated that such a system would make the substation hardware independant of future changes and requirements of relaying and diagnostic algorithms.

REFERENCES

[1] Aucoin, B. M., Russell, B. D. "Detection of Distribution High Impedance Faults Using Burst Noise Near 60 Hz" (IEEE Transaction on Power Delivery, vol. 2, no. 2, pp. 342-348, April 1987).

[2] Narendorf, M., Russell, B. D., Aucoin, M. "Microcomputer Based Feeder Protection and Monitoring System-Utility Experience" (IEEE Transactions on Power Delivery, vol. 2, no. 4, pp. 1046-1052, October 1987).

[3] Russell, B. D., Watson, K. "Power Substation Automation Using a Knowledge Based System-Justification and Preliminary Field Experiments" (IEEE Transactions on Power Delivery, vol. 2, no. 4, pp. 1090-1097, October 1987).

[4] Russell, B. D., Watson, K. "Knowledge Base Expert Systems for Improved Power System Protection and Diagnostics" (Proceedings, Symposium on Expert Systems Application to Power Systems, Stockholm-Helsinkl, Norway-Finland, August 1988).

[5] Russell, B. D., Chinchall, R. P. "A Digital Signal Processing Algorithm for Detecting Arcing Faults on Power Distribution Feeder" (Presented at IEEE/PES 1988 Winter Meeting, New York, Paper No. 88 WM 123-2, 1988).

[6] Russell, B. D, Chinchall, Kim, C. J. "Behavior of Low Frequency Spectra During Arcing Fault and Switching Events" (IEEE Transactions on Power Delivery, vol. 3, no. 4, pp. 1485-1492, October 1988).

[7] Russell, B. D., Mehta, K., Chinchall, R. P. "An Arcing Fault Detection Technique Using Low Frequency Current Components-Performance Evaluation Using Recorded Field Data" (IEEE Transactions on Power Delivery, vol. 3, no. 4, pp. 1493-1500, October 1988).

[8] Watson, K., Russell, B. D, Hackler, I. "Expert System Structures for Fault Detection in Spaceborne Power Systems" (Proceedings, Intersociety Engineering Conference on Energy Conversion, Denver, August 1988).

A DIGITAL PROTECTION SYSTEM INCORPORATING KNOWLEDGE BASED LEARNING

Karan Watson, B. Don Russell, Kurt McCall

Texas A&m University

ABSTRACT

This paper describes a digital system architecture used to diagnoses the operating state and health of electric distribution lines and to generate actions for line protection. The architecture is described functionally and to a limited extent at the hardware level. This architecture incorporates multiple analysis and fault detection techniques utilizing a variety of parameters. In addition, a knowledge based decision maker, a long term memory retention and recall scheme, and a learning environment are described. Preliminary laboratory implementations of the system elements have been completed. Enhanced protection for electric distribution feeders is provided by this system; advantages of the system are enumerated

1. INTRODUCTION

Designers of distribution protection systems generally agree that technological advances in the electronic computer industry should be incorporated carefully into new fault detection and protection systems. New systems must retain the level of security and reliability acheived by current relaying devices. Existing devices typically use preset current thresholds to detect faults, sometime resulting in excessive fault currents in the lines before tripping or a lack of sensitivity to faults. A level of sophistication has been reached in placing these devices, with various preset operations, in a protective network which not only protects generation and transmission equipment, but also attempts to disrupt the distribution system in a minimal way. However, a new protection system will improve performance while providing more extensive diagnostic capability for abnormal conditions.

New protection systems under investigation attempt to bring a higher degree of diagnostic intelligence toward the load in the electric system. This effort must be continually balanced between the performance to be gained and the cost of the system. The desired protection system allows for adaptability in the setting of parametric thresholds used to detect abnormal fault current situations. This is coupled with the ability to store, and possibly transmit, event situations to decision processors and operating personnel up the monitoring hierarchy, to enable better reactions to catastrophic faults in the system.

Research has shown that monitoring the parameters used for overcurrent protection, even when allowing adaptable thresholds, is not sufficient for detecting many faults on the distribution lines. [1] Many faults, such as high impedance arcing faults, downed lines, and incipient faults, pose a danger to the public, without necessarily drawing a level of electric current which can be classified as overcurrent. The development of methods to detect these non-catastrophic faults have been based on observing a number of distribution system parameters in addition to the fundamental currents and their harmonics. [2] However, no single set of parameters and algorithms has been documented to give complete detection of unhealthy situations in the distribution systems. Ancillary factors such as line construction, load types, weather, time, and even soil types affect the diagnostic ability of the proposed methods of detection. Therefore, the current knowledge in the development of a diagnostic system for power distribution lines indicates a combination of parameters and algorithms must be used to adequately protect the power system. Recent research indicates that knowledge based system programming techniques will enhance the capability of a protection system to combine various detection algorithms and allow for flexibility and adaptation. [3]

To be presented at the Intersociety Engineering Conference on Energy Conversion, Washington, D.C., August, 1989.

The development of a protection system which goes beyond catastrophic fault detection is desired for both terrestrial and long-term space-borne systems. In both environments, low-current faults may not cause immediate damage to equipment, but they pose a significant danger to personnel. However, unlike catastrophic faults where a clear reaction to detection is almost always prescribed, low current faults do not always create such clear reaction alternatives. Some situations should allow a low current fault to persist, as long as warnings have been issued, instead of disrupting parts of the power system which may be supporting life sustaining functions. Likewise, a choice of leaving a live wire on the ground may have to be weighed against cutting power to traffic controllers or hospitals. The protection system must be tuned for the environment where it operates.

The protection system described in this paper would be positioned as close to the loads as electrically and physically managable. We have envisioned this location to be at the distribution substation level with possible distributed data collection points on the feeder. The system provides a suitable architect for detection and reaction in a real-time frame. Rerouting, load adjustment, and even fault location are not necessary features at this level. However the provision of information to higher processing levels as they make these types of decisions is desired.

All of the desired protection system attributes were considered in the development of a digital system architecture for the power distribution environment for terrestrial, three phase 60 hertz systems. Nonetheless, this architecture seems flexible enough to be suitable for space-borne systems, even those using 20 kilohertz power.

This paper presents the architecture under development. Emphasis on the intelligence, both the knowledge based environment and learning techniques under investigation, will follow the description of the protection and diagnostic system hardware. While confidence remains high in the performance and flexibility of the hardware, discussion and input from many experts in the design of protection systems is still required in order to maximize diagnostic capabilities and reactions. Therefore, the knowledge based techniques and learning environment represent the current methods being investigated by the authors, and as always, the hope is to stimulate constructive inputs for future developments.

2. ARCHITECTURAL DESCRIPTION

A detailed description of the flow of data through the hardware of the protection system architecture can be found elsewhere. [4] Here the concern is to outline the architecture in order to understand the limitations which are presently imposed on the knowledge based and learning environments of the system. A functional block diagram of the architecture is found in Figure 1. The primary functions in the system include the Signal Conditioning and Conversion element ,SCC, the Data Distribution element, DD, the Processing elements, PE, the Intelligence element, and the Interface element.

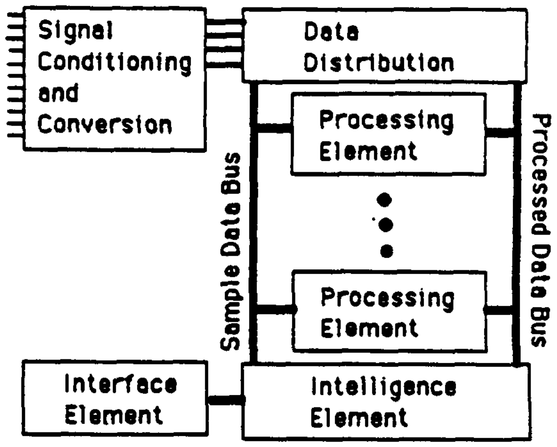


Figure 1. System Architecture

The SCC element (Fig. 2) contains a data acquisition board for each phase current of the line being monitored and an acquisition board for line voltages and other signals to be monitored. The choice of board separation is made with electrical isolation and system fault tolerance more in mind than board population. On these boards, signals are input after being transformed and conditioned to maintain safe levels for the digital equipment on the boards. The first conditioning steps on the boards involve analog filtering. These filters provide anti-alaising for the soon to be sampled signals. In addition, the currents are filtered further to provide various frequency ranges of information. For example, the anti-alaised signal is passed to a notch filter which eliminates the primary frequency to provide a signal for monitoring current harmonics. This signal then passes through a high pass filter to provide a signal with only high frequency information. All of these signals are amplified before sampling. Each signal has an independently controlled variable gain amplifier. Present versions of these amplifiers allow four discrete levels of amplification to be set for each signal. The processor on the acquisition board monitors the level of sampled data, and with consideration given to messages received from the Intelligence element, determines the appropriate gain setting for each signal. The processor also directs the sampling and digitization of the signal values. After conversion to 12 bit digital data, the data are serialized and passed through optic conditioners to transform from electric media to optic media. The data is output from the SCC element on optic fibers. These fibers are mainly used to enhance the electrical isolation of the elements to follow. However they also allow various SCC elements to be somewhat remotely located from the rest of the diagnostic system.

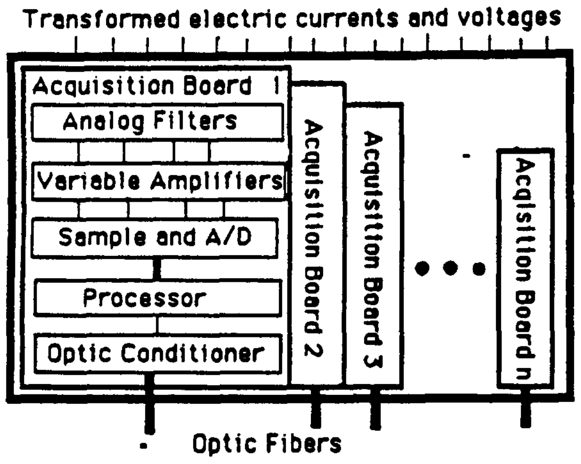


Figure 2. SCC Element

The DD element (Fig. 3) has a small data receiver board for each SCC optic fiber input. This board separation is made strictly for fault tolerance and replacement considerations. These boards function as converters from the the optical media back to electric media. Then a processor on board checks for transmission errors in the incoming data. Presently the data is simply flagged as 'dirty' when a transmission error is detected. Dirty data is not processed in the fault detection algorithms. All data and flags (clean/dirty, signal number, amplifier gain) are formatted into 16 bit words and stored in dual port memory found on the data distribution board. This board maintains the real-time system clock, provides the communication link between the Intelligence element and the SCC elements, and pushes data in a preset order onto the Sample Data Bus. In addition, this system watches for short term trends in dirty data to help warn the Intelligence element about drops in system integrity due to lost data. The processor on the data distribution board is the sole controller of the Sample Data Bus. Every processing element and the Intelligence element all receive a copy of the sample data being transmitted on this bus.

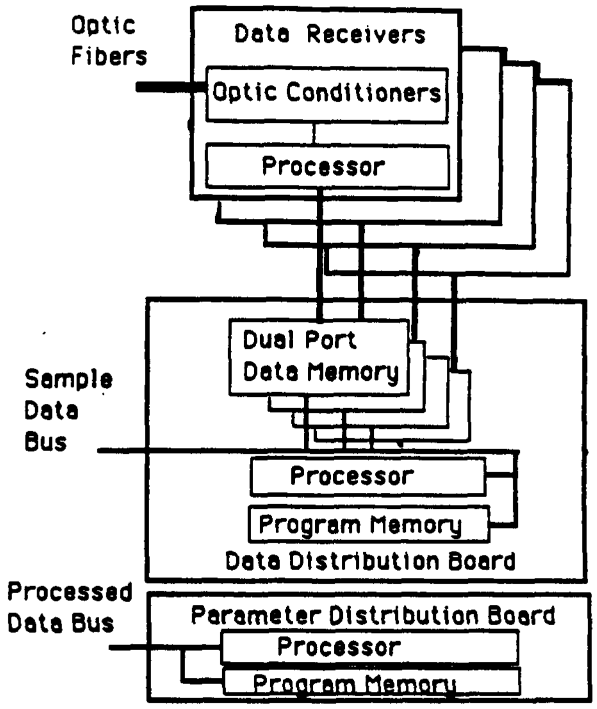


Figure 3. DD Element

The final board in the DD element is the parameter distribution board. The processor on this board is the arbitrator for the Processed Data Bus. This bus allows Processing elements and the Intelligence element to share processed and partially processed information. The Intelligence element sends system messages to all the other elements, excluding the Interface element, via this bus. Information is provided to the bus with an interrupt signal. The parameter distribution board sorts through these interrupts in order to optimize the flow of data through the bus.

The Processing Elements can have a variety of forms. Presently two forms exist, one has a standard 16 bit processor, with or without a floating point coprocessor, and the other has a 32 bit digital signal processor. (Fig. 4) Both types of PEs have a 1 KByte dual port RAM to interface to the Sample Data Bus. After 15 sets of sample data have been written in this RAM by the data distribution board, the oldest sets of data begin to be overwritten by the newest sets of data. The processor on the PE must access the data before it is overwritten or there is no easy way to regain the data. A variably sized scratch pad RAM is provided on the board for storage of partial results and data which must be retained for longer periods of time. Another 1 KByte dual port RAM is utilized to interface to the Processed Data Bus. In this RAM each parameter has a preset address and the responsible PE for that parameter updates the RAM and then generates an interrupt for the parameter distribution board to distribute the new data. Each PE assumes the parameter values in this dual port RAM are the most currently available values. The PEs also have program memory space. The PEs function as algorithmic processors to extract relevant system parameters from sampled data and to utilize these parameters in fault detection algorithms. The outputs of the PEs are parameter values, such as high frequency noise levels or third harmonic energy levels, as well as detection and confidence levels of various suspicious situations on the power line. The detection indicators and their confidence factor are used by the Intelligence element to reach a final diagnosis of the state of the power lines.

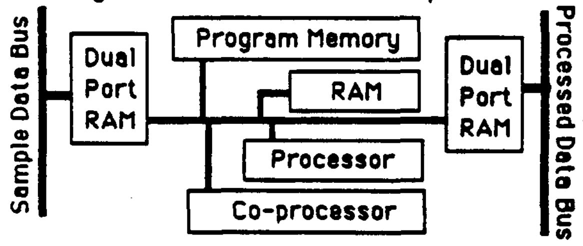


Figure 4. PE

The Intelligence element is a multi-board environment which is the focus of the following section. First the Interface element is described. There are three functions the Interface element supports for the diagnostic system. The Man-Machine interface allows limited amounts of information to be read from or input to the system by on-site operators. The Communication interface provides the path for information exchange to other processing environments or personnel in the overall power protection network. This is the path utilized for initial programming and program updates. Finally, the Output Control interface generates the required signals to cause power system actuators to respond to diagnosed situations on the power lines.

3. INTELLIGENCE ELEMENT

This element is responsible for three primary functions in the diagnostic system. (Fig. 5) The first of these functions is the system decision maker, DM. The DM determines the ultimate diagnosis regarding the health of the power system and the degree of certainty gained for this diagnosis. The second intelligence function is found in a unit called the Archivist. This unit maintains the long-term memory for the diagnostic system, and works to relate this memory to present events. The third function is the Learner unit. As indicated by these units the goal of the Intelligence element is to capture the knowledge and thinking which human experts, in our case operators and/or engineers, utilize to diagnose and react to the health of the power lines. These functions should be automated since there are not enough human experts available to monitor all power lines in a tireless manner, and even if a line were being continually monitored by a human, he may not be able to respond fast enough for true system protection. In capturing human expertise, we still have the goal to keep final implementations on board level micrcomputers.

3.1 <u>The Decision Maker, DM</u>

Several techniques for the detection of faults on power lines have been proposed. To date no single technique seems adequate for detecting all faults. Therefore, we utilize several of these techniques before the DM makes the final diagnosis for the system. Examples of detection techniques incorporated are:

- Amplitude Technique: a signals amplitude is compared to threshold values over different time periods. (Generally overcurrent detection method)
- Energy Technique: the energy found in a certain frequency range, or a discrete frequency, is compared to self-adaptible thresholds.
- Phase Relationship Technique: monitors abnormal shifts in the phase relationship

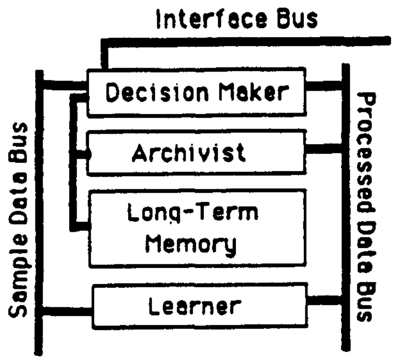


Figure 5. Intelligence Element

between combinations of primary and harmonic frequencies.
- Harmonic Sequence Component Technique: monitors abnormal levels in harmonic frequencies.
- Amplitude Ratio Technique: monitors amplitude changes of certain frequencies with respect ot other frequencies.
-Randomness Technique: Monitors nonharmonic frequencies for large random noise patterns.

Each technique can be applied to a number of different parameters in the system. The number of technique and parameter combinations activated concurrently influences the number of PEs reqired in the implementation. The DM must combine the information from the multi-techniques into a single diagnosis.

To formulate the ultimate system diagnosis the DM accepts fault detection indicators and confidence levels from the various techniques running. In addition the Archivist will supply the DM with information from similar experiences in the past. The primary output of the DM is a description of the diagnosis of the power lines with a belief level. The confidence factors are somewhat probabilistic in nature, although realistically they can sometimes be described as heuristic odds. Presently DM implementations explore the use of Dempster-Shafer calculus, or fuzzy set theorems , or a combination of the two in order to resolve multiple beliefs in detected faults into a single belief level. [5] Implementations are rated based on their success at detecting faults and their tendency to give false alarms. Potential users have indicated that in some situations no false alarms are tolerable even if some faults go undetected, while in other situations limited numbers of false alarms are acceptable. [6]

The diagnosis and confidence level generated by the DM is processed by a subunit in the DM which determines, with the utility operating procedures in mind, what the appropriate actions are for the diagnosis and protection system. *Some possible actions include:* adjusting the SCC amplifier levels, modifing parameter computations, changing the techniques which are active, transmitting a warning, alarm or report to other processors or operators, or operating a breaker on the power line. Thus the DM will form a diagnosis and plan appropriate actions for the situation at hand.

3.2 The Archivist

One important tool available to human experts is the ability to draw upon past experiences to form a conclusion and to plan a course of action. On line computing systems typically represent past experiences only in the form of algorithms or parameter values. Our system retains past experiences in a more accessible and adaptable format. The Archivist monitors parameters and beliefs available in the system. This data is compressed to a level which retains the essence of experiences relevant in fault detection. Compression also enhances the speed in which the information can be accessed. As well as classifying, or clustering, instantaneous experiences for storage, the Archivist classifies sequences of states.

After the classification process the Archivist informs the DM of the most similar previous experiences to the present situation. *The measure of similarity is also indicated.*

Finally the Archivist and DM determine the way the present situation should be remembered. This involves determining if this is another case of an previosly seen experience and should be used to augment the description of that previous experience. In contrast it may be decided that this is a new experience with too little similarity with previous experiences to affect their description. In this case the new experience is retained until it resolves into an old experience or becomes a new class of its own.

3.3 The Learner

The Learner attempts to look at the information available from the power signals and the diagnostic system with fewer preconceived notions about what will be good fault indicators and what will not. In this sense the Learner may request various parameter calculations that the DM has not activated. The DM's action subunit decides if this request is to be honored or not. The Learner monitors system data in order to derive its own diagnostic classification rules. This process can proceed in an unsupervised manner. However, we have allowed for a level of confirmation to be input to the Archivist and Learner through the Man-Machine interface unit. We expect our Learner to be most helpful in modelling non-fault situations to help reduce false alarms. This is due to the vast amount of non-fault data, with respect to fault data, usually encountered on an operating power line. In the long run, learning what is normal in a given environment, while using predefined detection techniques, should provide a quality combination for a diagnostic system.

The Learner represents the state of the system in terms of a multi-dimensional vector space. A similarity metric is used to classify new input vectors. Metrics range from complex correlation functions to simple Euclidian distances. The methods for clustering are numerous. We have given attention to methods such as the K-mean pass method [7] which utilizes a square mean error criterion for clustering. Although somewhat successful at classifying certain situations, it is an extremely slow and computationally expensive method. Another approach, Competitive Learning [8], is receiving the most attention at the writting of this paper. This approach is much faster and simpler to implement, but as predicted does not always result in clusters that correlate to actual events. Tuning of the implementation of this method should provide a very reasonable and helpful Learner for the diagnostic system.

4. CONCLUSION

A diagnostic system to determine the health of electric power distribution lines has been designed to enhance the level of fault detection and protection beyond existing overcurrent protection capabilities. The architecture presented allows flexibility for tuning and future developments in detection for not only overcurrent faults but also, high impedance arcing faults and even incipient faults. Diagnosis relies on inputs from multiple detection techniques followed by intelligent resolution of conflicting or uncertain information. A long term memory controller is important for the intelligent resolution to a final diagnosis. A learning environment is also involved, although not necessarily set up to modify decision making procedures in a closed loop fashion. The system promises enhanced detection and protection capabilities which are tunable to the economic and operational constraints imposed on the implementation.

5. REFERENCES

[1] Narendorf, M., Russell, B.D., Aucoin, M. "Microcomputer Based Feeder Protection and Monitoring System-Utility Experience" (IEEE Transactions on Power Delivery, vol. 2, no. 4, pp.1046-1052, October 1987).

[2] Russell, B.D., Chinchali, Kim, C.J. "Behavior of Low Frequency Spectra During Arcing Fault and Switching Events" (IEEE Transactions on Power Delivery, vol. 3, no. 4, pp. 1485-1492, October 1988).

[3] Watson, K., Russell, B.D., Hackler, I. "Expert System Structures for Fault Detection in Spaceborne Power Systems" (Proceedings, Intersociety Engineering Conference on Energy Conversion, Denver, August 1988).

[4] Russell, B.D., Watson, K. "A Digital Protection System Incorporating Real-Time Expert System Methodology" (Proceedings, CIGRE, Paris, June 1989).

[5] Tanimoto, S.L. The Elements of Artificial Intelligence Computer Science Press Inc., 1987, pp. 239-277.

[6] Russell, B.D., Aucoin, B.M. "Summary Report EPRI High Impedance Fault Detection Workshop" (New Orleans, November 1988.

[7] McQueen, J.B. "Some Methods of Classification and Analysis of Multivariate Observations" (Proceedings, Fifth Berkeley Symposium on Mathematical Statistics and Probability, 1967).

[8] Grossberg, S. "Competitive Learning: From Interactive Activation to Adapative Resonance" Cognitive Science, 1987, VII, pp. 22-63.

A PARALLELIZED RULE-BASED SYSTEM SHELL FOR MONITORING APPLICATIONS

Dissertation and Research Report

Grady Cook II

Department of Electrical Engineering

Texas A&M University

Submitted to

Dr. Karan Watson
Dr. B. Don Russell

TABLE OF CONTENTS

II, III, IV, V

TABLE OF CONTENTS (Continued)

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Artificial intelligence is currently a very active research area. Industry and academia alike are energetically searching for ways to employ artificial intelligence technology in useful applications. The branch of artificial intelligence research which has had the most commercial success thus far is the area of expert systems, also known as knowledge-based systems. Expert systems are computer programs which attempt to follow the human reasoning process in solving problems in a particular domain.

In the past, expert systems were developed to work in an off-line, advisory capacity. Typically, a user would use a computer terminal to input a problem and some pertinent information to the expert system, and at a later time the expert system would return an answer to the problem. An extension to this type of use of expert systems is presently being pursued. This extension is the incorporation of on-line, real-time control and monitoring capabilities into an expert system. There are major obstacles in the way, the most obvious being the inherent slowness of expert systems.

A class of expert systems which is receiving much attention in the research world today is rule-based systems (RBS). Most of today's RBS shells are designed with the assumption that the database of facts changes slowly. This is the assumption made by the designers of the most popular matching algorithm used in RBSs known as the Rete match algorithm [1]. This algorithm is usually very effective because this assumption is true in almost every case, but not in the case of monitoring problems. In monitoring problems many sensors are watched which

Journal model is *IEEE Transactions on Computers.*

causes the database of facts to change rapidly. This is the area of RBS research being addressed by this work.

There exists an obvious need for the development and implemention of a RBS shell which uses a matching technique that will work more effectively than the standard Rete matching technique for monitoring-type applications. In order to help satisfy this need for increased speed, parallelism should be explored. In creating a new matching algorithm to replace the Rete match algorithm, straightforward adaptability of the algorithm to a true multi-processor machine is highly desirable.

A. Objectives

The objective of this work is the creation of a RBS shell which runs faster than existing RBS shells for problems which have the property of a rapidly changing fact base, such as is found in monitoring type applications. The development and implementation of this new RBS shell requires the design of a new matching algorithm. This matching algorithm will run in parallel on a true multiprocessor computer for the purpose of obtaining faster response times. The ultimate goal is the creation of a technique for allowing the use of expert systems in monitoring applications.

The Rete match algorithm often used in RBS shells is considered by many to be the best matching algorithm available today. It attempts to cut down on searching requirements by storing partial match results from previous searches. This work shows that when the Rete match algorithm is used in a RBS developed for monitoring, the process of storing the partial matches causes the RBS to be very inefficient. This is due to the significant amount of time necessary to store partial results, which in the monitoring problem are rarely used. A replacement algorithm for the Rete match algorithm is developed and implemented on a true multi-processor machine as a means of achieving faster response times.

By exploring techniques for increasing the speed of RBSs used in monitoring problems, the probability of more intelligent monitoring devices being constructed in the near future increases. In a time when many companies are seeking to add intelligence to existing conventional technologies, this research appears to have the potential of helping achieve that very goal in the limited subarea of high-speed, intelligent process monitoring.

B. Dissertation Organization

The following chapter examines expert systems in general and how they are starting to be used in real-time applications. Examples of existing and proposed real-time expert systems are discussed. An analysis of the differences between standard real-time systems and conventional expert systems is also given. Chapter III presents CLIPS, a rule-based system shell developed by NASA, which is used as the basis for the implementation of the modified matching algorithm developed to run on a parallel machine. The Rete match algorithm as implemented in CLIPS is also discussed.

In Chapter IV information concerning parallel programming and parallel processing is reviewed. Techniques and methods which are available to parallel programmers are examined. Data about the Sequent Balance 8000 multi-processing computer is presented since this is the target machine for the parallelized modified match algorithm. The details of the new match algorithm are presented in Chapter V. How this algorithm differs from the Rete match algorithm and how it lends itself well to a parallel machine are discussed. An analysis of results obtained from test inputs is also given in this chapter.

Conclusions about the final results of this work are presented in Chapter VI. The influence which the Sequent Balance 8000 architecture had on the final results is discussed, as well as the possible advantages of other target machine architectures. Finally, the overall usefulness of the new parallelized match algorithm and what further work can be done on it are addressed.

CHAPTER II

REAL-TIME EXPERT SYSTEMS

A. Introduction

One of the most useful outgrowths of artificial intelligence research has been the development of expert system technology. Expert systems are presently being built and used in a wide variety of applications. Most of these applications are off-line processes in which the expert system receives a problem and some pertinent information and, at a later time, returns a solution. Expert systems have been very useful in these interactive, advisory types of applications, and will continue to be.

A potentially beneficial extension of the use of expert systems beyond the standard off-line applications level appears to be developing. This extension is the integration of expert system technology with on-line, real-time control and diagnostic systems. Conventional expert systems and real-time systems are very different in their basic structure. However, each technique has properties which could enhance the value of the other if an integration could be performed. Expert systems could benefit from the higher execution speed of real-time systems, and conversely, real-time systems could profit from the higher level of problem-solving ability inherent in expert systems.

An analogy of these two technologies can be drawn from the business world. An expert system may be thought of as a very knowledgeable consultant, and similarly, a real-time system (especially a real-time controller) may be thought of as a fast-acting manager. A consultant is typically well-versed in the technology of the company, but may not possess great organizational skills. Likewise, a manager is normally a good motivator and organizer, but may lack some skills in the finer details of company work. Both people are interested in helping the company make money, even though individually they are very different. If it were possible for the consultant and the manager to merge their positive traits into one, the resulting employee would be extremely valuable to the company. So it is with expert systems and real-time systems.

Many practical problems need to be overcome in designing a useful real-time system using expert system technology. The degree of difficulty varies depending on the particular application. For example, a system which requires control information in relatively long time intervals can more easily incorporate a slow-running expert system than can a system which requires control information in short time intervals.

In this chapter, expert systems and real-time systems initially will be described separately. Then a comparison of the two techniques will be made, followed by a discussion on what is required to achieve their integration. Next, previous work on real-time systems will be examined, along with some actual examples. Potential applications and areas of needed research will then be addressed before summarizing. By performing this study, the direction in which expert system applications are going will be better understood. In the not-too-distant future, these systems will constitute a major part of our technology.

B. Description of Expert Systems

An expert system is a computer program which embodies artificial intelligence techniques in order to simulate the reponses of a human expert in a given field of expertise. It is generally agreed that expert systems are presently the most useful application of artificial intelligence. Expert systems are being developed at a rapid rate as new beneficial uses are discovered.

There are several well-known expert systems in operation today[2]. One of the most famous is MYCIN, which was developed at Stanford to aid medical doctors in diagnosing bacterial infections and to make antibiotic therapy recommendations. Another well-known diagnostic expert system is PROSPECTOR. This expert system is used by geologists in locating mineral deposits, and is credited with actually finding a previously unknown deposit of molybdenum.

In these two examples, as in most conventional expert systems, the expert system is simply a program running on a computer which gives advice to the user interactively via a computer terminal. Information is fed to the expert system by the user, and after a period of time, the system returns a written response on the screen. This type of expert system obviously has many practical uses. However, a new idea being developed is to have expert systems automatically receive information from sensors or other sources, and automatically transmit control signals to whatever they are monitoring and controlling. This new idea will be discussed in more depth in section E.

Expert systems differ from standard procedural programs in that expert systems do not have a simple sequential control flow throughout the program. Symbolic processing, as opposed to numerical processing, is emphasized in expert systems. This is why LISP (LISt Processing language) is often used as the underlying programming language. One drawback in using symbolic processing is its inherent slower execution speed and greater memory requirements. In many applications, however, this is more than offset by the higher level of decision-making sophistication which is made possible. The general operation of an expert system will now be discussed.

There are three main parts to a conventional knowledge-based expert system - the knowledge base (rules), the data base (facts), and the inference engine (rule interpreter). The knowledge base is the set of rules which provides the reasoning for taking specific actions. Generally, these rules are written in a modular IF-{set of antecedents}-THEN-{set of consequents} format which makes it easy for rules to be added, modified, or deleted. An antecedent is a pattern that may or may not exist in the data base. When all the antecedents in a rule are present in the data base, the associated consequents may then be fired (executed). These consequents are usually assertions of new facts to be placed in the data base, which may then cause a different rule to have all of its antecedents satisfied. A consequent may also initiate some type of control action, such as sending a message to the screen. The modularity of the knowledge base makes development of the system more organized, and therefore simpler.

As already alluded to, the data base of facts, also known as the working memory, is the set of data or patterns that exist in the expert system at a specific time. Generally, the knowledge base of rules is static once the expert system is running, but the data base is very dynamic with facts being continually inserted, modified, and deleted by the consequents of rules which are firing.

Since the rules in the knowledge base do not fire sequentially, some form of control must be implemented to manage the ordering of events. This function is performed by the heart of the expert system, the inference engine. It is the responsibility of the inference engine to determine which rule fires when, since it is possible that several rules have all of their antecedents satisfied concurrently. The inference engine is also responsible for making all the pattern matchings necessary to determine which rules are ready to fire, as well as making sure a rule fires only once each time its antecedents are satisfied.

There are two main strategies that an inference engine uses to control program flow. The first has already been indirectly discussed. It is the *data-driven* control strategy, also known as *forward-chaining* and *antecedent-reasoning*. In this method, the inference engine checks rules to see which have all of their antecedents present in the current data base. Rules that have all their antecedents matched are fired, and rules that do not are left alone. This process continues until no more rules can fire, or until some desired goal is reached and execution is halted. Figure 1 shows the control flow found in forward chaining.

The second main control strategy used by inference engines is called the *goal-driven* control strategy. The terms *backward-chaining* and *consequent-reasoning* are also used for this method. In this method, a goal (problem) is initially posted (asserted) in the data base. The inference engine then takes that goal and searches through the rules to see which consequents match the goal. If such a rule is found, the antecedents of that rule are searched for in the data base. If all the antecedents can be satisfied, the problem is solved. However, if some antecedents are not satisfied, they are either posted as subgoals so that the process may continue recursively, or the program prompts the user via the terminal to see if the missing antecedents are true or not. At this point, the expert system can easily explain to the user why it is needing the information - it just shows the goal that is solved if the antecedents in question are true. Figure 2 shows the control flow found in backward chaining.

Expert systems have been a topic of research for two decades, but only recently have actual systems been developed. Now that the wheel is turning,

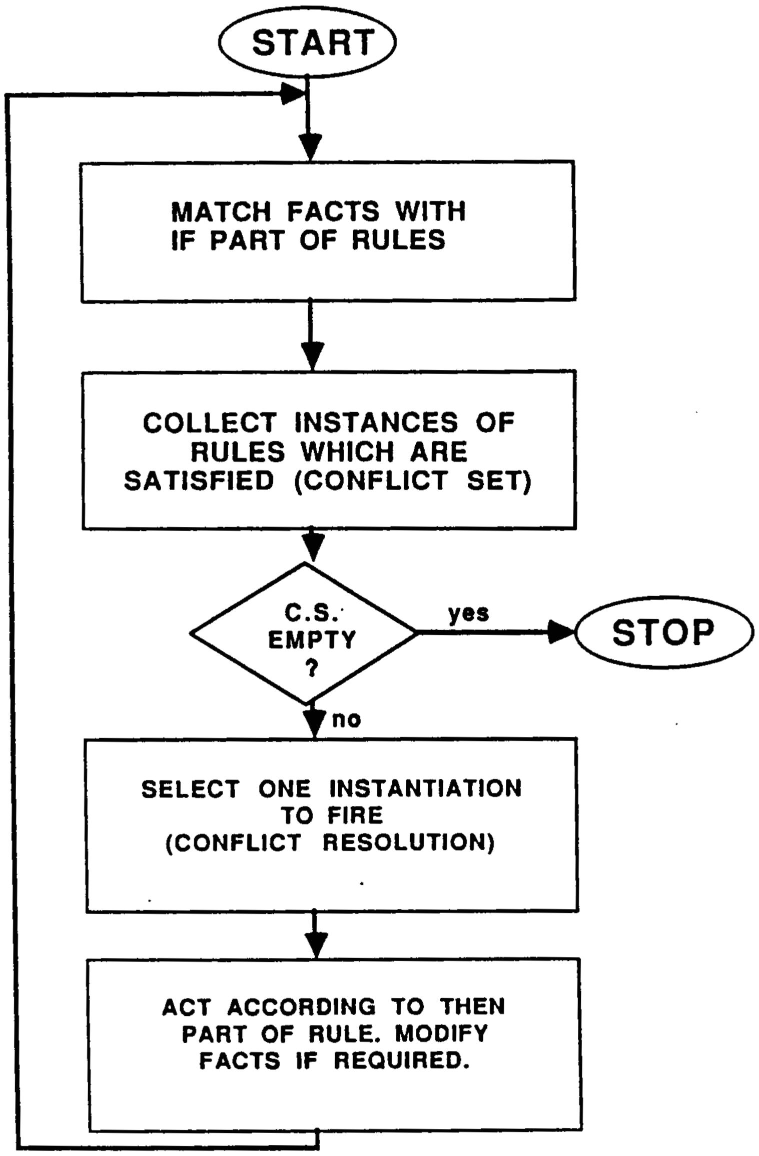

START
MATCH FACTS WITH IF PART OF RULES
COLLECT INSTANCES OF RULES WHICH ARE SATISFIED (CONFLICT SET)
C.S. EMPTY ?
yes
STOP
no
SELECT ONE INSTANTIATION TO FIRE (CONFLICT RESOLUTION)
ACT ACCORDING TO THEN PART OF RULE. MODIFY FACTS IF REQUIRED.

Fig.1 Forward Chaining

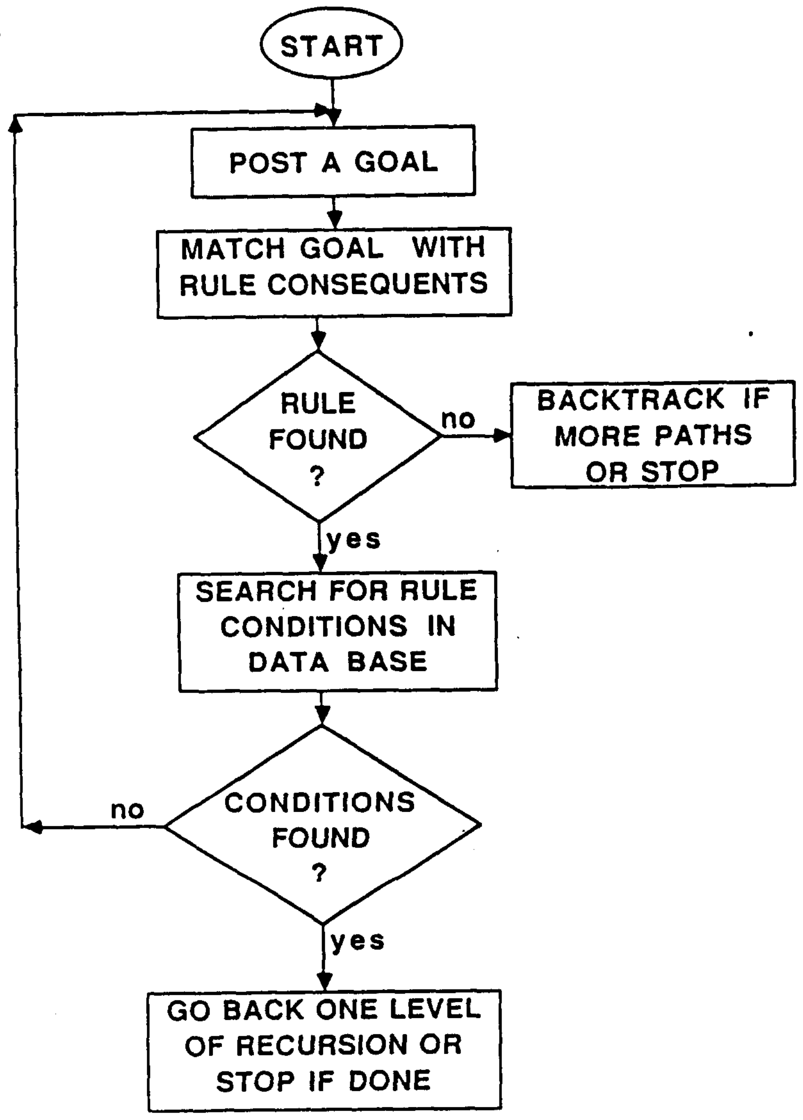

START
POST A GOAL
MATCH GOAL WITH RULE CONSEQUENTS
RULE FOUND ?
no
BACKTRACK IF MORE PATHS OR STOP
yes
SEARCH FOR RULE CONDITIONS IN DATA BASE
CONDITIONS FOUND ?
no
yes
GO BACK ONE LEVEL OF RECURSION OR STOP IF DONE

Fig.2 Backward Chaining

it is gaining speed rapidly. Although previous expert system work was oriented towards off-line, interactive advice, that limitation may soon change to include real-time activity.

C. Description of On-Line, Real-Time Systems

Real-time systems are systems which are guaranteed to respond within a fixed period of time. The value of the fixed period of time can vary greatly, depending on the specific system in question. For example, the term "real-time" in a water reservoir controller refers to something considerably slower that what is referred to by "real-time" in a fighter aircraft navigation controller. In general, real-time refers to some type of strict time requirement that has been placed on a particular system.

"On-line" is a term which implies automatic and autonomous control of a process. In the sense of a system controller, this means the controller is continuously in operation receiving status data and transmitting control signals. Usually, when a process is referred to as "real-time", it is implied that the process is "on-line" as well, but not always. This is discussed in more detail later.

As mentioned earlier, real-time systems are characterized by their high-speed execution. How fast a real-time systems needs to run depends on the particular application being considered. Real-time systems are generally used in control and diagnostic applications, with the most common of the two being control.

A typical real-time controller consists of a small microprocessor-based system which has a limited amount of memory. The emphasis of the controller is on high speed and not on sophistication of operation. A limited number of operations are required of the system, and these are performed by the processor via numerical manipulation, which the processor is well suited to do.

The typical microprocessor-based real-time controller is partitioned into highly modular tasks which are coordinated by a well-defined algorithm. Often, in an effort to increase operating speeds, a table-driven scheme is set up. The program code in a processor is usually able to manipulate data in tables more quickly than to repeat calculations. The code is also completely compiled, which gives it a faster execution speed than an equivalent interpretive system.

A real-time controller is on-line because of its closed-loop configuration. A real-time diagnostic system, however, does not exert control on the process it is monitoring, and thus is not on-line, even though it senses in real-time. This illustrates another feature of real-time systems – the manner in which they handle input/output requirements. In the typical real-time controller, the input is a set of sensors, and the output is a set of effectors. This is a distinguishing mark of most real-time systems.

There are many applications in which on-line, real-time systems are being used. Some of these include process control, robotics, fault detection and diagnosis, and signal processing. In such systems, when the complexity increases, the controller becomes very hard to construct. This is due to the inherent inability of real-time systems to handle sophisticated decision-making requirements. Obviously, this is an area of real-time systems technology which needs to be researched and developed.

D. Comparison of Expert System and Real-Time System Characteristics

The characteristics of expert systems and real-time systems are quite different. One of the most noticeable differences is execution speed. A real-time system typically responds in less than a tenth of a second, whereas an expert system may take ten seconds or more to respond. The difference in execution speeds may be understood by examining the way real-time systems and expert systems normally process data. Real-time systems are usually implemented on standard microprocessors which are designed to do fast numeric calculations. Expert systems, however, are usually run on computers with LISP processors which process data symbolically. Symbolic manipulation is inherently a slower process than number crunching.

Another characteristic which influences execution speed is the program code form. Program code for real-time systems is normally compiled to run on a microprocessor. Expert systems, on the other hand, because of their LISP basis, generally have program code which must be interpretted. Interpretted code execution is slower than compiled code execution.

Real-time systems and expert systems are typically used for unrelated purposes. Real-time systems are normally found in controllers and diagnostics systems, whereas expert systems have thus far been most useful as interactive advisors. Expert systems have also been used in diagnostics work, but only in the sense of receiving symptoms entered at a terminal and returning a diagnosis of the problem. The differences in application areas of real-time systems and expert systems can be attributed to their individual strengths and weaknesses. For example, expert systems are capable of a high level of sophistication in decision making, which makes them suitable for advisory work. Real-time systems generally are not capable of making decisions based on complex information. However, they have the ability to have closed-loop control over a process. Expert systems typically are not able to directly control any process. Obviously, as far as system capabilities are concerned, expert systems and real-time systems are far apart.

The physical structure of real-time systems and expert systems is also quite different. As mentioned earlier, real-time systems are normally implemented on standard microprocessors, while expert systems usually use a LISP processor. In addition to this, there is a great difference in the memory requirements of the two. Real-time systems do not need nearly as much memory as do expert systems. As a rough estimate, a typical real-time system will use 64 Kbytes of memory, while an expert system will use 4 Mbytes. The amount of physical space taken up by the two types of systems also differs considerably. A real-time system may require only a single board, whereas a typical expert system requires a complete computer system. Because of the expert system's extra computer resources, more highly developed system development interfaces are possible. Real-time systems typically have only rudimentary system development interfaces, if any at all. In this sense, expert systems are easier to develop than real-time systems.

One last physical structure which may be contrasted between real-time systems and expert systems is their respective input/output devices. Real-time systems, especially real-time controllers, generally use sensors as inputs and effectors as outputs. Conventional expert systems, however, use a computer terminal for both.

A summary comparison of real-time systems and expert systems is given in Table I.

E. Integration of Expert Systems with Real-Time Systems

Expert systems are known for their slow response time. This is due to the basic control flow of the expert system. The system must generate the conflict set

TABLE I.
Characteristics of Real-Time Systems and Expert Systems

	Real-Time System	Expert System
Typical Speed	$< 10^{-1} sec.$	$> 10^{1} sec.$
Processor Type	microprocessor	Lisp processor
Processing Method	numeric	symbolic
Program Code Form	compiled	interpretted
Typical Use	control	advising
Decision Capability	low	high
Control Type	closed-loop	no control
Memory Required	64 Kbytes	4000 Kbytes
Physical System Size	single board	complete computer
Development Interface	rudimentary	highly developed
Input/Output Devices	sensors/effectors	terminals

from the entire set of rules and then resolve the conflict set before any action is taken. Therefore, the integration of an expert system with a real-time system at first appears to be an almost impossible task. Indeed, for this operation to take place, some constraints must be placed on the resulting system. However, even with these constraints, very useful on-line, real-time expert systems are possible even with today's technology. This will be demonstrated very soon.

There is at least one trivial solution to the problem of integrating an expert system with a real-time system. This trivial solution is to begin with a real-time system where the timing constraints are very lax, as in the case of a water reservoir controller[3]. Almost any expert system technique could be used to control this process. While this solution is trivial, it is valid since it satisfies the definition of real-time by responding within a fixed period of time.

Another solution, which is not trivial, is to have faster computers running the software. This is a solution being addressed by the designers of many new computing systems, including the LISP and Prolog machines. Since expert systems are often written in LISP, a speed up in the execution of LISP would result in a speed up of the execution of many expert systems. The best way to speed up LISP is to have specialized hardware which is designed with that goal in mind. LISP processors and LISP machines using these processors have been designed and built with significant speed increases[4]. Continued development of faster and faster LISP processors will aid in the on-going processs of integrating expert systems with real-time systems.

In special cases, the response time of an expert system may be sped up by translating the original computer code into a language which usually runs faster[5]. Since many expert systems are written in LISP, it is conceivable that translating from this slow running language to a faster one, such as C, could result in a more compact and faster code. Another minor technique which may help in a few special cases is to cut down the number of rules in a knowledge base to the bare minimum. This could result in a faster running system.

For most applications, the use of parallelism appears to be the best solution to the problem of integrating real-time systems with expert systems. In this technique, a trade-off is made between space and time; the response time is reduced, but more space is needed. In space-restricted applications, such as on the space station, this may not be the best method to use. However, in general, this appears to be the most promising solution.

An architecture for a parallel inference machine has been developed by Tanaka of Japan[6]. This machine, known as PIE, is organized into two distinct levels. The top level has 64 identical subsystems, each of which contains 16 inference units. This makes the total number of inference units to be 1024. Within each inference unit is a memory module, a unifying processor, definition memory, a fetch buffer, and an activity controller. Inferences are conducted in parallel within the inference units under the global control of a central system manager. Simulations using this architecture have shown cases of problem solution speeds being increased by a factor of as much as 170 over the speed of a single conventional processor.

The concept of parallelism can be used in different ways. One potentially useful configuration is to have a special LISP microprocessor present to process knowledge-based inferences quickly. Other conventional processors would be present to do standard calculations in parallel with the LISP processor, as well as each other. If an efficient and effective interfacing of the processors could be developed, this parallel scheme would be very useful[7].

In some problems, parallelism will not be able to make the total computation time short enough to be classified as real-time. Another technique known as *metaplanning* may be able to help in such situations[8]. Metaplanning is a strategy in which a separate monitor (metaplanner) overlooks the activities of the lower-level process controllers. The metaplanner itself must run in real-time, but the lower-level process controllers may be non-real-time algorithms. The metaplanner plans the overall operations, assigns tasks, keeps track of time, and, in general, manages all activity. When problems arise, it replans the operations in order to keep the system within the real-time constraints it has been given.

The main techniques for integrating expert systems with real-time systems, then, are faster specialized hardware, metaplanning, and parallelism. Of these, a combination of faster specialized hardware and parallelism appears to be the best general solution.

F. Examples of Real-Time Expert Systems

By examining some of the real-time expert systems being developed today, more information may be extracted on the characteristics of this type of system. The next several paragraphs will deal with the design concepts used by different groups in constructing their real-time expert systems.

Lockheed Aircraft has designed a real-time expert system which controls the chemical processing and coating of aircraft parts[9]. This is one of the few on-line, real-time expert system designs which has actually been implemented. Lockheed's experience led them to conclude that the real-time control process should be separated from the expert system by a special interface. In this setup, the real-time controller is composed of a network of microprocessors which are responsible for controlling the mechanical hardware. The expert system part, which is implemented on a separate minicomputer, communicates with the network of microprocessors and is responsible for scheduling the overall activities. With this arrangement, the expert system itself is not operating in real-time, but it is contributing to the overall operation of the system. To handle its problem of not being a real-time process, the expert system is designed to output a contingency plan when it does not have time to make a complex analysis of the system's current status.

An on-line, real-time expert system has been developed by IBM to control the operations of a computer system[10]. A modified version of OPS5 was used in constructing the expert system. Part of the strategy used in building the system was to use a virtual memory computer to allow the total system to be modularized. Inside the single virtual memory computer, three interconnected virtual machines were built. These three machines are the expert system, the display controller, and the communications control facility (CCF). The CCF interacts directly with the computer to be monitored. It translates messages from the monitored computer into code which can be understood by the expert system virtual machine, so that the expert system does not have to bother with communications requirements. This allows for faster inferencing. The display control is connected to a terminal for external operator control, when desired. This overall strategy is similar to that proposed by Lockheed in that the real-time controller and expert system are separated. IBM has built and used this system successfully. Since the development effort was great, and the resulting system was very machine dependent, expansion of this idea to other systems will probably be slow.

The concept of adaptive trip levels is discussed by Henkind in his paper on the use of real-time expert systems in medicine[11]. In a standard intensive care unit (ICU) monitoring system, several sensors which measure various physiologic parameters are attached to the patient. When one of the sensors measures a reading which is outside of its preset range, it activates an alarm. In some cases, the preset range could be inappropriate for a particular patient. For example, the left ventricular filling pressure for a normal patient is 8 to 12 mm, but for a patient who has had ventricular hypertrophy, the appropriate range is 14 to 18 mm. The expert system which is monitoring this condition should have a rule which checks for a ventricular hypertrophy condition, and adjusts the trip level accordingly. This is an example of one type of adaptive trip level mechanism. The more general type makes adjustments according to on-going measurements.

For proper operation, the trip level of a sensor may need to be adjusted according to physiological changes, as well at to changes in time. For this reason, Henkind has proposed an expert system scheme which has blocks of rules which are applicable depending on the time and physiological state. This type of system is not on-line in the strict sense since it only sets off an alarm, but it effectively can be classified as on-line in the sense that the activity is continuous. It is real-time since data is being processed within a given time-frame.

More intelligence can be added to this system by having it compare readings from different sensors. Some serious conditions can be detected by looking at the values of multiple sensors, even when all the sensors are individually within their safety limits. This multivariate interaction has not been considered before in ICU monitoring systems, and is obviously an area where a real-time expert system would be valuable. To handle the multivariate interaction problem and the adaptive trip level problem, Henkind has proposed to divide his rule base into two parts, the permanent rule base and the working rule base. The working rule base has time-dependent rules, and the permanent rule base has non-changing rules. While the actual implementation of this system is not complete, it appears that when it is, it will be a useful *real-time* expert system.

A system has been built in England named Escort (Expert System for Complex Operations in Real-Time)[12]. This system has been configured to assist process operators in oil production platform control rooms. One of its main objectives is to keep operators from being so overloaded with system warnings that their performance is lowered. Escort monitors all activities and when problems arise, it displays the problems to operators in order of importance. Only the six most important problems are visible to operators at one time, which makes it easier for operators to start taking corrective action.

Escort is implemented with LISP and Loops (an expert system building tool) on a Xerox 1108 LISP workstation. It responds in real-time, which in this system is defined to be within one second. It is on-line in the sense it is continually in operation sensing the status of the plant. However, it does not exert any active control. By monitoring the plant status, Escort is able to warn operators of problems, and to even make suggestions. In the future, when Escort has stabalized and proven its reasoning ability, it may be given power to automatically control corrective processes.

Escort has broken its duties down into modules which are then regulated by a scheduler. The first module is the filter. The filter is designed to recognize situations which may require operator attention. The next module, the initial prioritizer, takes the data from the filter and attempts to arrange the plant problems in order of importance. This prioritization process may be rather crude at this point since the underlying problem has not yet been determined. The diagnostic module then analyzes the output of the initial prioritizer in an attempt to determine the main problem. Instrument failures, such as open or closed circuits and stuck valves, as well as operator errors, are considered in the diagnostic phase. After this is done, final ordering is performed in the final prioritization module. This data is then displayed for the operator. All of the activity is controlled by the scheduler.

While Escort is implemented on a single fast processor, a distributed system with multiple processors could potentially speed up the system even faster. Escort's built-in modularity could make the transition relatively easy. Overall, Escort is a good example of a useful real-time diagnostic expert system.

All of the above examples have two characteristics in common: the timing requirements are not very strict, and a large computational system is available on which to build and run the system. These factors cause the resulting system to look more like a conventional expert system than a conventional real-time system. The addition of input sensors, however, is a major development over standard expert systems. Apparently, very little work has been done to incorporate expert system technology into space-restricted applications. The only known work in this area will be looked at next.

SRI International has developed a prototype real-time knowledge-based control system which runs on a microcomputer[13]. The name given to this system is Hexscon, which stands for Hybrid Expert System Controller. The goal of the project was to produce a framework on which controllers could be built which were small and fast, as well as sophisticated. Hexscon itself does not have any domain-specific knowledge, but rather contains an inference engine which knows how to run a knowledge base. The information that goes into the knowledge base is provided by experts in a particular field. Therefore, Hexscon can be used in many different types of real-time control problems.

Figure 3 shows how Hexscon is organized. As the figure indicates, knowledge engineers communicate with experts in order to get the information needed for the system's knowledge base. This information is first loaded into a larger machine. In order for the system to be microprocessor-based, the knowledge base is then compiled into space efficient code to be loaded into the microcomputer. Inside the microcomputer system the compiled knowledge base resides with the Hexscon inference engine and conventional logic.

The conventional logic is the main guarantee that real-time response will be achieved. If it determines that there is not enough time to do more detailed thinking about how to respond, it responds in the same way a conventional real-time controller would. If time is available, a second level of reasoning is performed by the knowledge-based portion of the controller. Up to four levels of reasoning are possible if time permits. The three top levels of reasoning are all carried out by the knowledge-based portion.

The designers of Hexscon decided to use Pascal as their implementation language since it compiles into a fast, compact code. Ideally, they would like to have a LISP machine on which to develop the original knowledge base, and then have this data converted to Pascal or directly into machine code. To do this, only a subset of LISP could be used. Future versions of the Hexscon system will probably incorporate this idea.

Preliminary test results of systems using Hexscon appear good. An Intel 8086-based system with 256 Kbytes of memory was able to respond in a time

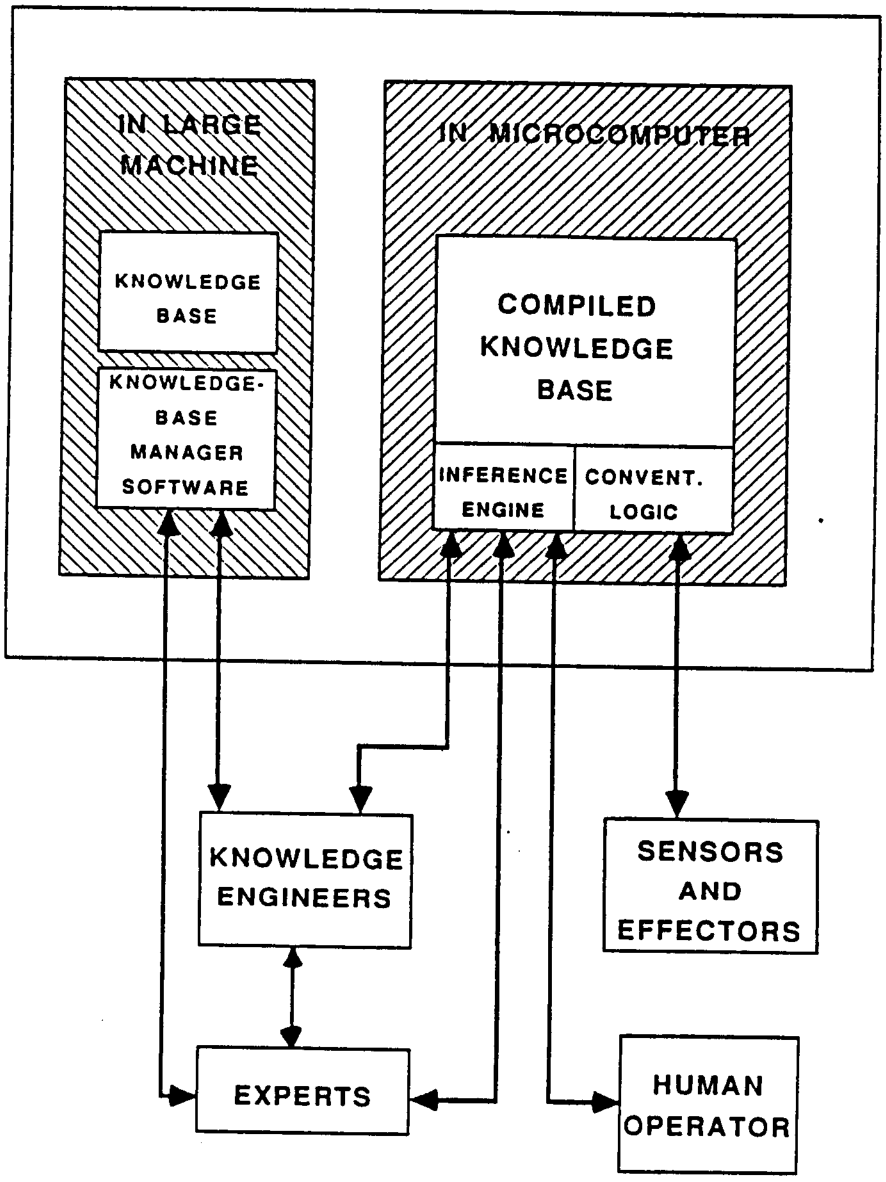


Fig.3 Hexscon Architecture

range of 0.25 to 0.50 seconds. In this set up, a moderately sophisticated rule set was used which contained about 5000 rules, of which about 250 were actually used. This example run shows that fairly fast response times are achievable with small, microprocessor-based systems.

A fairly new system on the market which claims to be useful in real-time expert system applications is PICON (Process Intelligent CONtrol)[14]. PICON runs on an LMI Lambda/PLUS enhanced LISP machine. This machine is essentially a dual processor computer in which one processor is a specialized LISP chip and the other is a standard 68010 microprocessor. These two processors are tightly coupled by a direct bus. Operations which require very fast response times are handled by the 68010, and operations which need more high level reasoning are handled by the LISP processor.

PICON can be arranged in a heirarchical order in two ways. First, rules may be arranged in a heirarchy for efficiency. Secondly, separate PICON rule sets may be arranged in a heirarchy so that the results of one rule set may be input to a higher rule set. The developers of PICON feel it will be a highly used system in the near future. Its main apparent drawback is its strong ties to a particular machine.

G. Potential Applications

It is fairly easy to think of applications in which real-time expert systems would be valuable. Any system which has a need for fast response and intelligent decision making is a potential candidate.

One field of research which is an obvious area where real-time expert systems could be used is robotics. A robot system with any capabilities will have a large number of sensor inputs and effector outputs which must be processed rapidly. This requires real-time processing ability. To handle the large amount of sensor data, expert system qualities also will be needed.

Process control of complex systems is another obvious potential application area for real-time expert systems. Complicated signal processing problems also could benefit from this technology.

Other uses which have been proposed are as an aircraft pilot assistant[15], a mass spectrometer controller[16], a computer system debugger[17], and a nuclear power plant controller[18].

One especially interesting potential application being looked into at Texas A&M is a real-time expert system for monitoring and protecting power systems. Ideally, this system could make diagnostic checks of the state of power systems and take corrective actions when necessary. This type of system could be used in earth bound and space bound systems. Since space bound power systems will not be very accessible, some built-in intelligence on self-diagnostics and correction could make its operation more reliable and manageable.

H. Areas of Needed Research

As mentioned earlier, most existing real-time systems using expert system technology lean more towards being conventional expert systems than conventional real-time controllers. Usually a large computer system is available on which the system can be developed. In many cases this is very convenient, but there exist many other applications which could use real-time expert system technology if the total physical space requirements were reduced down closer to the size of a typical microprocessor-based real-time system. This is precisely the case in the Space

Station power system where space is at a premium. It would be absurd to ship up a complete, large-scale computer system to do the controlling and diagnostics operations.

Some research needs to be performed to determine how to build powerful, but small, real-time expert system controllers. A study should be made into which artificial intelligence techniques would provide the best speed results in this type of system. This would involve checking timing characteristics of techniques such as forward and backward chaining, and depth-first and breadth-first searching.

I. Summary

Expert systems and real-time systems in themselves are very different. However, by combining the positive traits of both of these technologies, an extremely powerful new technology emerges. Countless potential applications for this new technology already exist, and once the technology has been refined, many more will be found. At present, the real-time expert systems which do exist are not as powerful as future systems hope to be. As parallelism techniques are developed, many more useful real-time expert systems should appear. Real-time expert systems will be a major part of our technology in the near future.

The following chapter takes a look at an existing RBS shell which uses an efficient matching algorithm. A discussion is made of the problems which arise when using current RBS shells to implement real-time expert systems. Much work needs to be done in this area.

CHAPTER III

CLIPS AND THE RETE MATCH ALGORITHM

CLIPS is a rule-based system shell which was developed by NASA as a tool to allow for the easy creation of expert systems. The matching technique employed by CLIPS is known as the Rete match algorithm, and was developed by Charles Forgy at Carnegie-Mellon University. This chapter discusses some details about CLIPS and how the Rete match algorithm it uses operates. CLIPS and the Rete match algorithm are discussed here because they are progenitors of the new match algorithm and its implementation which are presented in Chapter V.

A. CLIPS

CLIPS stands for "C Language Intelligent Production System." It is written in the C programming language, which is a fairly recent trend in expert system shell writing. In the past, LISP was used almost exclusively in writing expert system shells. CLIPS has several basic features which make it attractive to many users. These features are enumerated in the following subsection. Next, the structure of programs input to CLIPS is looked at.

Basic Features: As mentioned, CLIPS is written entirely in C. The developers choice of doing this brought several advantages. One of the main advantages of the C language is its high portability. Programs written in C usually can be moved from one machine to another with little or no modification of the source code. Another advantage of programs written in C is their run time performance. C is closer to assembly language than most programming languages and tends to produce fast, compact code. These advantages of C are inherited by CLIPS.

Another basic feature of CLIPS is its reasoning technique, which is forward-chaining. Forward-chaining is designed to efficiently take a set of premises or conditions and arrive at a conclusion by going through a series of inferences. The matching technique which is used in performing the forward-chaining process is the Rete match algorithm, which is described in detail in section B.

CLIPS is very versatile. It can be totally embedded within another C program and called as a subroutine, or it can be run independently. This freedom allows for enhanced usefulness. Users are even capable of defining their own functions to be called within CLIPS. This demonstrates CLIPS notable extensibility which makes adding new features very easy.

From a development point of view, CLIPS is well designed. Interactive development is possible for the relatively easy construction of rule sets to be run under CLIPS. Debugging aids have been included to help in quickly tracking down problems in coding.

For system developers, CLIPS has the added features of available source code and some documentation. The main source of documentation is the reference manual [19]. Obviously, the source code contains complete information about CLIPS, but in a cryptic form.

Input Code Structure: In general, rule-based systems are made up of three components: rules, facts, and an inference engine. CLIPS is no exception. The inference engine is simply the run time portion of the rule-based system shell. CLIPS provides this for the user. The user, however, must supply the rules and facts in order to create an expert system. Different rule-based system shells have varying capabilities and restrictions on how the rules and facts may be described. The characteristics of CLIPS rules sets will now be examined.

Rule declarations have two main divisions, a left-hand side (LHS) and a right-had side (RHS). The LHS corresponds to the conditions of the rule, and the RHS corresponds to the actions to be taken if the rule is fired. In CLIPS the LHS contains some information in addition to the list of conditions. Each rule is required to have a unique name and a comment about what the rule does. Even though the comment may not say anything, the creation of the slot is still required, which strongly encourages documentatioin of rule sets. These are included in the LHS.

The most important part of a rule is the set of conditions in the LHS. CLIPS is very flexible in what it allows in the set of conditions. Essentially every combination of Boolean logic is legal in setting up the individual conditions of a rule. For example, for a rule to be satisfied, it may be required that both of the first two conditions be true, and only one of the following three, and not the last condition. Even more flexibility is achieved by the unlimited number of patterns possible within each individual condition.

In CLIPS a pattern looks like a LISP list, which is a pair of parentheses enclosing one or more elements. Each element in CLIPS is a character string which represents something, usually a constant pattern, a single variable pattern, or a multiple variable pattern. A constant pattern element in a condition must match the exact same constant pattern in a fact for it to be possible for the condition to be satisfied. The single variable pattern in a condition is a little more flexible, and will match any single constant pattern in a fact, as long as the elements are in the same location in their lists. Multiple variable patterns are even more flexible than single variable patterns since they will match with zero or more constant patterns in a fact. Obviously, with this high level of flexibility also comes a high level of complexity. Many rule-based system shells do not allow for this much leeway in rule writing.

CLIPS conditions are even more flexible than what has just been described. Each element in a condition may also have logical operators attached to them which limit the scope of possible matches. Essentially every combination of "not," "and," and "or" are possible. Also, predicate functions may be attached to individual elements which can perform almost any additional restrictive checks desired. Therefore, conditions may be extremely general or extremely specific, depending on what the intended goal is.

The salience of a rule is also stored in the LHS. Whenever a rule has all of its conditions satisfied by the current fact base, it will fire if its salience value is higher than the salience values of all other rules which have their conditions satisfied. This is a means of giving more relative importance to certain rules, which is what the human mind often does in making decisions.

The RHS of CLIPS rules is where the actions to be taken are described. There are two main actions which typically take place when a rule fires. The first is the assertion of a new fact into the data base of facts, and the second is the retraction or deletion of an old fact from the fact base. Some rule-based system shells allow for the direct modification of an existing fact, but CLIPS does not. However, CLIPS is able to accomplish the same results by simply performing a retraction and an assertion together.

In addition to assertions and retractions, the RHS may also perform some type of input-output function, such as write a message to the screen. Indeed, almost any funtion call may be made, since CLIPS allows for the insertion of user defined functions. Very powerful code can be generated because of this easy extensibility.

The RHS also is capable of performing conditional execution of actions by providing an if-then-else statement. This certainly in not a necessity for writing a rule base, but at times it makes the code more understandable. Looping is also possible with the while statement provide.

As mentioned earlier, the user of CLIPS must provide a set of initial facts in addition to the set of rules. Facts are very straightforward in their declaration. Each fact is a list of elements, just as in a rule condition, except fact elements are not allowed to be variables - they must be constant character strings. As a program runs, these facts are constantly retracted and asserted to signify the present state of the system.

When a CLIPS program is to be run, it is first loaded into the system with a load command. Next, it is reset with the reset command, and finally it is executed by typing the run command. Depending on the contents of the rules and facts, the program will prompt the user for inputs, or else run to completion on its own. Other interactive commands are available to the user, such as a command which prints the facts presently in the fact list, and another command which shows the contents of the agenda. These are helpful in debugging rule sets.

B. The Rete Match Algorithm

Rule-based systems cycle through three steps when they are running. These are select, act, and match. The selection step is known as conflict resolution and involves deciding which instantiation from the set of possible candidates (the conflict set) should be selected to be fired next. The action step deals with executing the RHS of the instantiation that was selected. Typically the actions cause a modification of the fact base. The final step is the matching process, which is concerned with matching facts with rule conditions to determine which rules should go into the conflict set. It is this final matching step which can consume more than 90% of the total execution time, and is therefore the area that is considered when speedup is desired. The matching algorithm used by CLIPS and other rule-based systems, such as OPS5, is the Rete match algorithm. Its operation will be examined next.

The Main Components: A rule-based system using the Rete match algorithm has four main components: the agenda, the fact list, the opnet (or pattern net or condition net), and the join net (or web or rule net). These four components work together in providing the structure needed to execute the rule set.

The agenda is essentially an ordered conflict set. It holds all the instantiations which are satisfied by the rule set but which have not yet fired. The instantiations are ordered according to their salience values so that the top number of the agenda will be the next one to fire.

The fact list is a list of facts stored in the machine which represents the state of the system. Facts are constantly added and deleted from the fact list as execution of the expert system continues. The initial state of the fact list is determined directly by the programmer, but the final state is determined by the flow of the program.

The word "rete" means network, so it is logical that the Rete match algorithm uses two networks. The first network used is the opnet, which is utilized during run time to match facts from the fact list with condition patterns from the set of rules. The opnet is a compiled indexing structure which helps cut down on the need for an exhaustive search between facts and conditions. Facts are fed into the top of the opnet and compared, element by element, with the nodes of the opnet. Mismatches of elements cause a redirection of the matching search, so that unfruitful branches are not taken which would waste time. If a fact is able to match all the way to a terminal point in the opnet structure, a pointer is provided for the fact to know where to attach to the join net, which is discussed next. A single fact may match several different condition patterns since patterns are allowed to contain variables that will match zero or more elements. The actual flow of things should be more apparent when an example rule set is explained in the next section.

The join net mentioned above is used by the Rete match algorithm to perform intercondition consistency checking of variable bindings. This is in contrast to the opnet, which does intracondition consistency checking. In the join net, all the conditions of a rule are checked to see if they have consistent variable bindings. Each condition may be satisfied by a fact in the fact list, but if a common variable name is bound to different elements, the rule is not yet an instantiation.

The join net contains a set of nodes, each with a right-hand side (RHS) and a left-hand side (LHS). The RHS of a node is used to hold information about facts which satisfy the particular condition represented by the node. This is the location pointed to by the opnet when a fact is successfully fed through the opnet. The LHS of a node is used to hold the set of variable bindings which is consistent up to that point. Variable binding information stored on the LHS is compared with the binding information on the RHS to determine if any combinations for a consistent set exist. All consistent sets of variable bindings are sent up to the LHS of the next higher node. The comparison process continues as long as new legal combinations are found. Whenever a set of variable bindings is to be passed up past the last node of the rule, that indicates an instantiation has been found. The set of variable bindings is then used to create a new member of the agenda. To determine the location of the new member in the agenda, the associated salience value is used.

When a fact is to be retracted, all variable binding sets in the join net which use the fact must be located. Facts in the fact list keep pointers to where they are being used in the join net to make this operation more efficient. The fact to be retracted is first traced to the RHS of a node in the join net. Next, a comparison is made between the fact and the RHS of the node to see of it has been pushed up to the next join. If it has not, no more action needs to take place, but if it has, the same type of process must be done up the join list as far as possible. All variable binding sets using the fact to be retracted need to be excised. If an instantiation uses the fact, it needs to be taken out of the agenda since it can no longer legally fire. The example given in the next section should explain the details of the Rete match algorithm more clearly.

C. An Example of CLIPS Execution

Figure 4 shows the source code of a small CLIPS program which represents a problem in which blocks on a table are moved around, one at a time, to achieve some specified goal. In this case, the goal is to end up with block B on top of block D. Figure 5 shows graphically the initial set up of the system and the fact list which describes this state.

This rule base has three rules which are used to manipulate the fact list until the goal is satisfied. The first rule, move-block-from-table-to-goal, is activated when a single move of a block from the table to the specified goal state can be

```
;;;-----------------------------------------------------
;;;   Sample Blocks World Problem
;;;-----------------------------------------------------

(deffacts initial-facts
   (on A B)
   (on B E)
   (on C D)
   (on nil A)
   (on nil C)
   (ontable E)
   (ontable D)
   (goal B on-top-of D))

(defrule move-block-from-table-to-goal ""
  (declare (salience 100))
  ?clause-1 <- (goal ?block-1 on-top-of ?block-2)
  ?clause-2 <- (ontable ?block-1)
  (on nil ?block-1)
  ?clause-4 <- (on nil ?block-2)
  =>
  (assert (on ?block-1 ?block-2))
  (retract ?clause-1 ?clause-2 ?clause-4)
  (printout ?block-1 " moved on top of " ?block-2 "." crlf))

(defrule move-block-from-block-to-goal ""
  (declare (salience 100))
  ?clause-1 <- (goal ?block-1 on-top-of ?block-2)
  ?clause-2 <- (on ?block-1 ?block-3)
  (on nil ?block-1)
  ?clause-4 <- (on nil ?block-2)
  =>
  (assert (on ?block-1 ?block-2))
  (assert (on nil ?block-3))
  (retract ?clause-1 ?clause-2 ?clause-4)
  (printout ?block-1 " moved on top of " ?block-2 "." crlf))

(defrule move-block-to-table ""
  (declare (salience -100))
  (goal ?block-x on-top-of ?block-y)
  (on nil ?block-1)
  ?clause-3 <- (on ?block-1 ?block-2)
  =>
  (assert (ontable ?block-1))
  (assert (on nil ?block-2))
  (retract ?clause-3)
  (printout ?block-1 " moved on top of table." crlf))
```

Fig.4 Example CLIPS Code

FACT LIST

1- (ON A B)

2- (ON B E)

3- (ON C D)

4- (ON NIL A)

5- (ON NIL C)

6- (ONTABLE E)

7- (ONTABLE D)

8- (GOAL B ON-TOP-OF D)

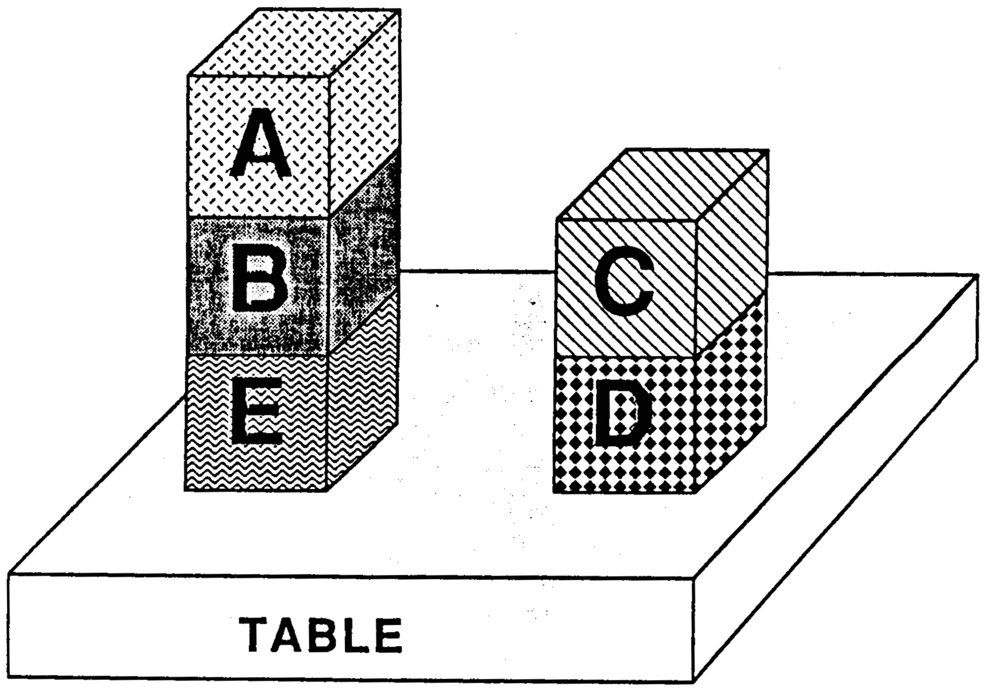


Fig.5 Depiction of CLIPS Example

legally made. This rule has four conditions which must be satisfied in order for the associated actions to fire.

The second rule, move-block-from-block-to-goal, is similar to the first rule, except it is activated when a block, which is presently on another block, can be moved to achieve the desired goal. This rule also has four conditions. The salience values of the first two rules are high because their actions indicate the goal has been reached. Further action will not be necessary after either one of them has fired.

Most of the real work is done by the third rule, move-block-to-table. This rule has a low salience since it should be performed only when nothing else can be done. It is used to move a block off of a stack of blocks onto the table.

Figure 6 shows the opnet generated by CLIPS to be used in the Rete match algorithm. It is used to efficiently match facts with rule conditions. In order to create the opnet, CLIPS takes all the conditions from all the rules and checks them to find similarities between them. For example, the condition (on nil ?block-2) from the first rule starts out the same as the condition (on ?block-1 ?block-2) of the second rule. The opnet uses this observation to create a network which matches the element "on" at the same location for both conditions, but will then branch off in matching the rest of the elements. This can be seen toward the bottom of figure 6. This indexing scheme helps cut down on the search space when matching is being performed.

When a fact matches all the way through the opnet, it is then pointed over to the join net. For example, the first fact, (on a b), goes through the opnet and gets directed to the join net by the 45-degree angle pointer at the bottom right corner of figure 6. This pointer points to two places in the join net as seen in figure 7.

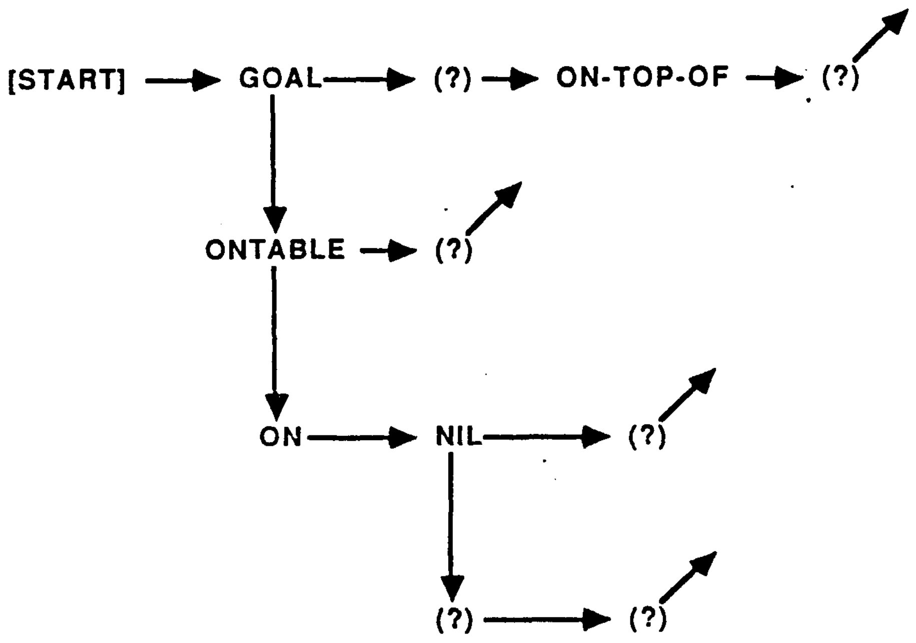


Fig.6 Example Opnet

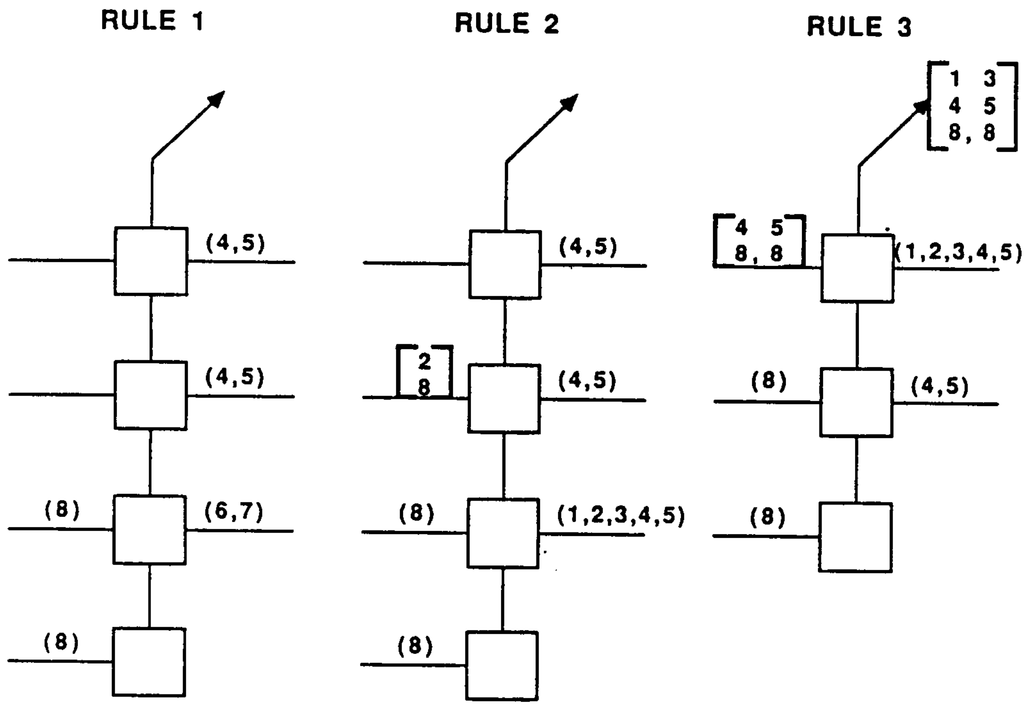


Fig.7 Example Join Net

The opnet pointers always point to the RHS of the nodes in the join net since they represent intracondition consistency. For example, the first fact is stored in the second condition of rule 2 and the third condition of rule 3. By looking at the source code in figure 4, this makes sense. The nodes (square boxes) listed in the join net are in reverse order compared to the source code listing of the conditions. By taking all the initial facts and going through the opnet, their correct locations in the join net may be determined. The numbers on the RHS of the nodes in figure 7 show the number of the facts which match the associated condition patterns.

Looking at figure 7, it is seen that the bottom nodes of each rule are different in that they do not have a RHS. This is because these nodes represent the first condition of their respective rules and do not have any previous conditions to compare variable bindings with. The numbers on the LHS of these nodes represent facts which match these conditions. The LHS of the nodes directly above the bottom nodes contain the same facts as the bottom nodes, except when the bottom node uses negative logic. Negative logic is used to mean that there must not be any fact which matches the condition for the condition to be true.

In general, however, the LHS is used to hold the lists of consistent variable bindings of all the nodes below the present node. For example, in rule 3 of figure 7, the LHS of the top node shows two list, (4 8) and (5 8). This means that the facts corresponding to these numbers make the first two conditions true. The next step is to see if either one of these lists is consistent at the top node with any one of facts 1 through 5. In this case, there are two such cases which are consistent: (1 4 8) and (3 5 8). Since these two lists of facts satisfy the conditions of rule 3, they are sent to be put on the agenda for firing at a later time.

When a rule fires, it typically modifies the fact list. In the Rete match algorithm, when a new fact is asserted, it is added to the fact list and then immediately goes through the opnet to find its place in the join net. When the proper location in the join net is found, it attaches to the RHS and straightway compares itself with the lists of variable bindings on the LHS. If a new list of consistent variable bindings is found, it is shipped up to the LHS of the next higher node. At this point, the new list compares itself with the facts on the RHS of the node. If new consistent variable bindings are found, they are similarly shipped up to the LHS of the next higher node. This continues as far as possible. In some cases, the agenda may even be reached with new members.

Retraction in the Rete match algorithm is similar to assertion, except the chaining process described looks for places the fact was used and deletes those variable binding lists. Members of the agenda may also end up being deleted. The process of going all the way through the networks to delete lists which use excised facts would normally be very expensive, but since on the average not many facts are retracted in a production cycle, the total cost is usually acceptable. In cases where several facts are to be retracted each cycle, the cost may become excessive.

On each cycle, the list of facts typically changes, which causes a change in the agenda. The top member of the agenda is fired after all modifications to the fact list performed by the last rule firing are finished. This cycle continues until no more members are left on the agenda, which should indicate the desired goal has been reached.

D. Summary

The Rete match algorithm is a normally efficient technique for matching a large set of constant objects with a large set of patterns[20]. There are some restrictions on the effective use of this match algorithm, however. First, the patterns used must be compilable, which means they must be in a form which makes it possible for an opnet and join net to be constructed. Secondly, the objects (or facts) must be constant. They cannot contain variables. Finally, the set of objects should change relatively slowly. This is due to the way the algorithm stores partial matches between cycles. In monitoring problems, the fact list changes rapidly, so another matching algorithm should be found for dealing with that type of application.

CHAPTER IV

PARALLEL PROCESSING

Parallel processing is a research area which is currently receiving much interest. Specialized computers are being created which can be utilized in many types of problems to achieve significant speed increases over conventional uniprocessor computers. This chapter will discuss some of the techniques used in programming these specialized multiprocessor machines. Also, the different architectures used in designing these computers will be looked at. A closer look at the shared memory architecture will be taken by examining a particular machine with this type of architecture, the Sequent Balance 8000.

A. Parallel Programming Techniques

Efficiently programming a multiple processor machine is much more complex than programming a single processor machine. Many more options are available, as well as potential hazards. Care must be taken in coordinating the activities of the individual processors, or else unpredictable results may be obtained. This section looks at some techniques which may be employed in effectively programming parallel applications. General concepts and basic methods are discussed.

General Concepts: Parallel programs may be catagorized according to several characteristics. Perhaps the most important characteristic is communication, which is the passing of information between processes. If an excessive amount of time is needed for communication, the efficiency of the implementation will drop drastically since little time will be left for computing[21].

Three factors affect communication in parallel programs. First, how much information is being passed? Large messages take a long time to transmit.

Secondly, how often are the messages sent? Frequent communication requires a lot of overhead. Finally, what is the topology of the interconnection network which is being used to transmit the messages? If the physical layout of the processors and interconnection network does not match the logical layout of the parallel programs, messages may need to be routed through several processors before reaching their intended destinations. This could lead to a significant drop in performance.

Parallel programs may also be catagorized by the granularity of the units of computation which are performed in parallel. Granularity refers to the relative size of the code segments which are executed before synchronization is required. A course grained implementation contains code segments which are long, and a fine grained implementation contains code segments which are short. Obviously, granularity is inversely proportional to the frequency of communication. Fine grained implementations pay a penalty for requiring a high frequency of communication.

Another important characteristic of parallel programs is the amount of inherently serial code which must be run. If an implementation must run 10 percent of its code sequentially, then no matter how many processors are available, or how efficiently they may be used, the speedup possible is no more than a factor of ten. This is because even if zero time were needed for the parallel portion of the algorithm, 10 percent of the time would still run sequentially. Some algorithms are simply not adaptable for efficient implementation on a parallel computer.

As mentioned, parallel programs have some inherent complexities not found in serial programs. One complexity is data consistency. In some parallel programs, different processors may have access to a single shared variable. Extra care must be taken when reading and writing this type of variable from memory. Locking mechanisms are typically used so that two processors will not clash in storing and accessing a shared memory segment. If two processors could write the same variable at the same time, chaos would result.

Process synchronization is another consideration which must be made. Often one running process will need another running process to be at a certain point in its execution for correct operation to occur. In these situations, events and barriers are set up. Events are used to tell a process to wait until another process says to continue. Barriers are used to synchronize a set of processes to a certain point in their code. Processes wait at the barrier point until a specified number of other processes arrive at that point. Regular execution then proceeds at that time. Using these techniques helps keep the overall operation from becoming uncontrolled.

There are two main kinds of parallel programming: multiprogramming and multitasking. Multiprogramming is a technique in which unrelated programs are executed concurrently. This type of operation is typically handled by the operating system and is not of much concern to application programmers. However, multitasking is of extreme importance to programmers who could benefit from having several processors running simultaneously on a single application. The two main branches of multitasking are called function partitioning and data partitioning. These two techniques will be discussed in the following two subsections. The discussions offered have a slight tendency to be oriented towards the shared memory architecture since this is the type of architecture which was used in implementing the parallelized match algorithm[22].

Function Partitioning: Function partitioning, also known as heterogeneous multitasking, is a parallel programming method employed when the algorithm to be implemented is so structured as to lend itself to having different tasks executing in parallel. In this arrangement, the complete data set is available to all of the processes being executed, but each process does something different with the data. Each process is unique and can be thought of as a specialized program function which has a specific job to do.

Obviously, trying to coordinate and organize a group of unique processes to work together in such a way as to create an efficient implementation of an algorithm is a very arduous task. Indeed, function partitioning is typically utilized only when no possible way of using data partitioning can be found. Data partitioning is described in the next subsection.

There are, however, cases where function partitioning fits quite naturally. For example, a process control program typically contains a host of independent functions, all of which look at a set of sensor parameters. Function partitioning could potentially be used to assign different process control functions to different processors so that all of the functions could be performed in parallel. Another example which seems to fit the function partitioning mode is program compilation. In this case, various passes through the source code could be performed in parallel. Each pass would perform a different function in compiling the code. Alternatively, separate modules of the source code could be compiled concurrently and linked together when all are finished. However, this would actually be a better example of data partitioning than function partitioning.

There are two basic techniques for doing function partitioning: the fork-join technique and the pipeline technique. The fork-join technique is the more straightforward of the two and requires that each of the main functions be independent of each other. The results of a function are not needed by any other function which is executing in parallel. In this technique, processes which correspond to the different functions are initially created. Next, each task is assigned to a process and begins running. When a process finishes executing, it waits for the rest to finish before continuing on into the serial code.

The pipeline technique is quite different. As its name implies, processors are logically arranged in a sequential fashion so that data may be sent through the pipeline. Each processor performs a unique function and when finished, passes the results on to the next processor in line. Once the pipeline has been filled, all the processors are working simultaneously on their own special jobs, much like the way it is done in a factory assembly line.

Data Partioning: The second main multitasking programming method is data partitioning, also called homogeneous multitasking. This method is essentially just the opposite of function partitioning, since in this case the processes are not unique, but rather they are all identical. Also, the way the data is handled is different. In function partitioning, each process uses all of the data, but in data partitioning, each process is assigned only a portion of the total data set. In general, data partitioning is more efficient and easier to use than function partitioning. However, it is not always possible to fit the algorithm to be implemented into the data partitioning method.

The classic example of where data partitioning is useful is matrix multiplication. In matrix multiplication, rows of the first matrix are multiplied by columns of the second matrix. The exact same operation is repeated n times, where n is determined by the size of the matrices. By assigning different processors the exact same multiplication code to execute, but on different rows and columns, the time needed to perform the matrix multiplication can be drastically reduced.

Just as there are two basic ways of using function partitioning, there are also two basic ways of using data partitioning. These two way correspond to the manner in which execution of a parallel program is scheduled. The two general scheduling techniques are known as static scheduling and dynamic scheduling.

Static scheduling is used when it is known that the computation time needed for each process is approximately equal. In this arrangement, data segments are assigned to processors before execution begins. Since each process will take about the same amount of time, processors will not have to wait too long for other processors to finish their part of the work. This method has the advantage of not requiring any communication between the processes. Communication is a major cause of parallel program inefficiency. The matrix multiplication example mentioned earlier would fit well into the static scheduling technique since it is known that each row-column multiplication takes approximately the same amount of time.

If the length of time for each process in a set of processes to be executed varies significantly, then static scheduling becomes inefficient; processors spend too much time waiting for others to finish. In this situation, dynamic scheduling may be more suitable. In dynamic scheduling, the set of processes which need to be executed are put in a task queue and fed out one at a time to available processors. When a processor finishes a process, it goes back to the task queue and is assigned a new job to do. This continues until the total job is done. Before execution begins, it is not known precisely which processor will be working on which subtask, so pre-execution assignment cannot be made. Jobs are assigned dynamically.

Dynamic scheduling has the positive characteristic of keeping all of the processors busy, which tends to increase the overall efficiency. This is known as load balancing. However, a price is paid to achieve this constant activity, and that is communication costs. Dynamic scheduling requires communication between the processes in order to keep the overall execution ordered. Static scheduling creates no communication overhead. The application to be implemented dictates which trade-off to make. The parallelized match algorithm discussed in Chapter V makes use of dynamic scheduling.

B. Parallel Machine Architectures

Currently there are two main architectures used in designing computers: the distributed memory message passing (DMMP) architecture and the shared memory architecture[23-26]. These two basic architectures are differentiated by the manner in which they interconnect the processors present. The following two subsections discuss these two parallel computer architectures.

Distributed Memory Message Passing Architecture: The DMMP architecture, also known as the multicomputer architecture, is characterized by the presence of direct communications links between processors. Every processor is not directly linked to every other processor, typically, but there do exist sufficient links to allow for messages to be routed between all processors. There is an obvious trade-off between the cost of including extra communications links and the time needed to send a message from one processor to another. The more links present, the lower the transfer time of a message.

Another characteristic of the DMMP architecture is the way system memory is organized. In this architecture, each processor contains its own local memory bank and no global shared memory location exists. For some applications, this structure works well, but in many cases programming this type of computer can be extremely awkward. Applications which are well suited for programming on this architecture include many simulation problems, such as heat diffusion and solid stress simulations. Different processors may be assigned to represent different physical locations of the object being simulated. The message passing capabilities naturally handles the boundary condition requirements.

Figure 8 shows an example of a DMMP architecture. This figure represents a three dimensional hypercube network which contains eight processors. The system memory is not shown, but physically resides in individual segments with each processor. Hypercubes are currently very popular in the parallel computing world. They consist of 2^n processors, each of which is connected to n other processors. The longest distance between two arbitrary processors is n links. In the example hypercube in figure 8, each of the eight processors can directly communicate with three other processors, and the maximum distance which has to be traveled between any two processors is three.

Shared Memory Architecture: The shared memory architecture, also known as the multiprocessor architecture, is characterized by the lack of direct communication links between the processors. Communication is achieved by using a single shared memory to which all processors are connected. The complexity of this architecture comes in designing the interconnection network used to connect the processors and the shared memory.

One extreme possibility of an interconnection network for the shared memory architecture is the crossbar. This type of network connects all processors to all memories simultaneously. Obviously, there would be no bus contention with this setup, but for most systems of any size, the cost would be prohibitive.

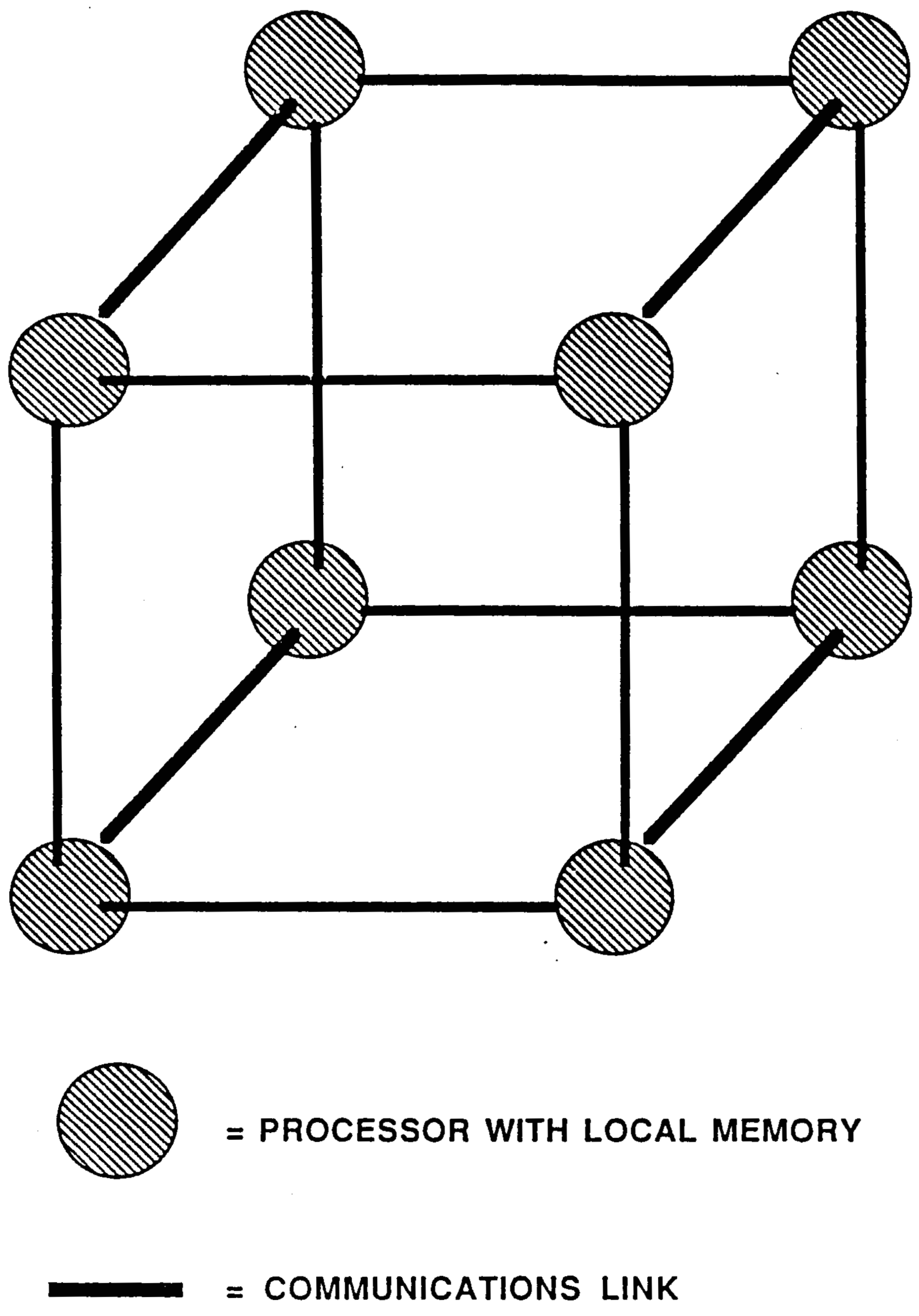


Fig.8 Three-Dimensional Hypercube

The other extreme possibility of an interconnection network for the shared memory architecture is the single shared bus. This network will not allow more than one processor at a time to access memory, which severely limits the number of processors which may be used. If a large number of processors are connected to a single shared bus, most of them will spend a large percentage of their time just waiting to access the bus. Obviously, this would be wasteful. Single shared bus networks are, however, relatively inexpensive to implement. Figure 9 shows a shared memory architecture which uses a single shared bus.

The two extremes of parallel computer architectures are the DMMP and the shared memory architectures. In the real world, some systems have been designed which are hybrids of these two extremes. For example, the Sequent Balance 8000 is classified as a shared memeory machine, but in actuality each processor contains a small amount of local memory. Because of this, the Balance 8000 is not a true shared memory machine, but for all practical purposes it is close enough from the programmers point of view.

C. The Sequent Balance 8000

The Sequent Balance 8000 is an example of a shared memory parallel computer which uses a single shared bus. It is versatile in that it may be used for dedicated parallel applications, as well as in a general purpose, multiuser environment. Since is release in 1984, the Sequent Balance 8000 has enjoyed relatively good commercial success. This section will look at the architecture and components of the Balance 8000[27]. An overall view of the structure of this machine may be seen in figure 10.

Balance 8000 Bus: The heart of the Sequent Balance 8000 is its single shared bus,

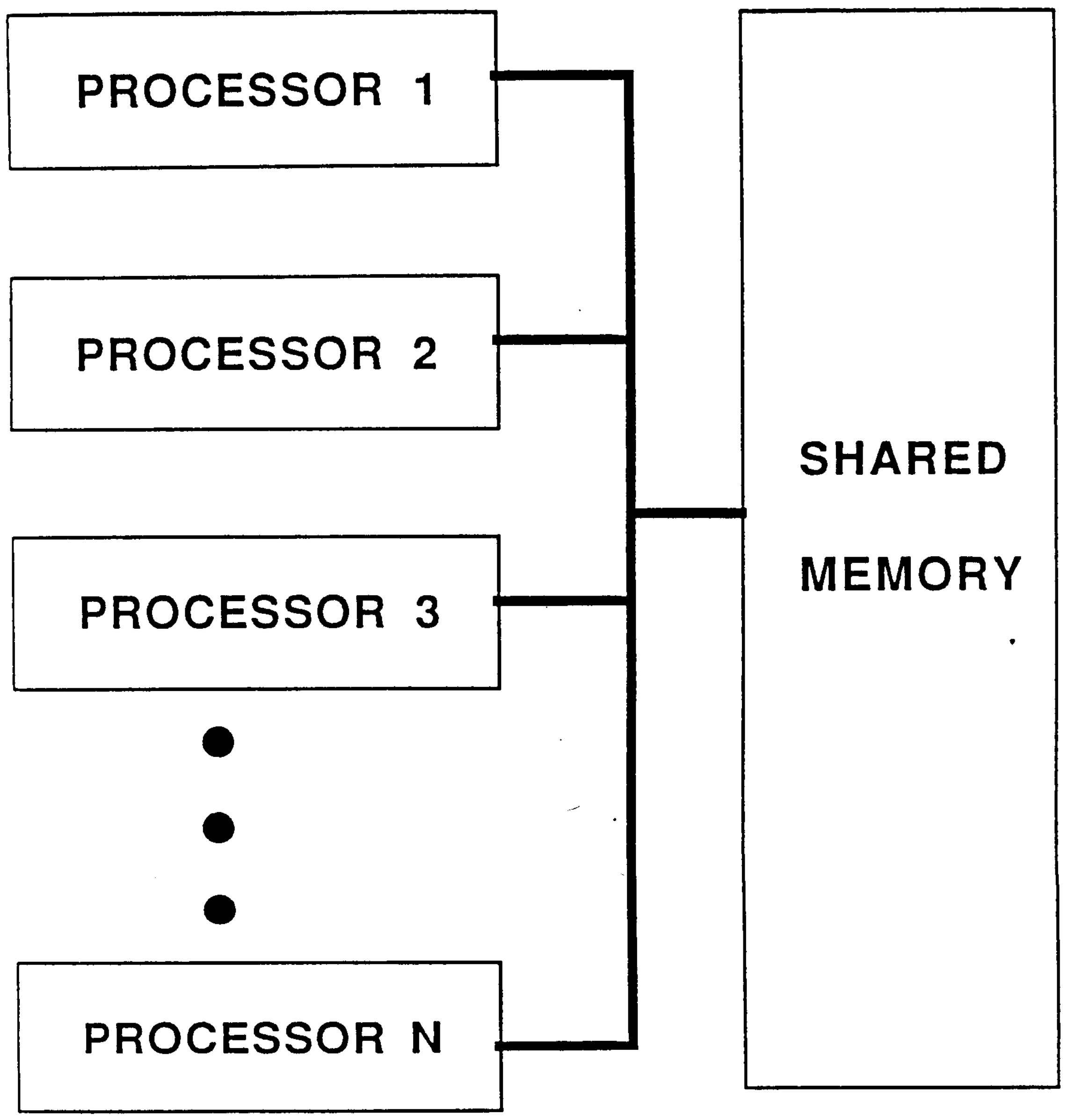


Fig.9 Shared Memory Architecture

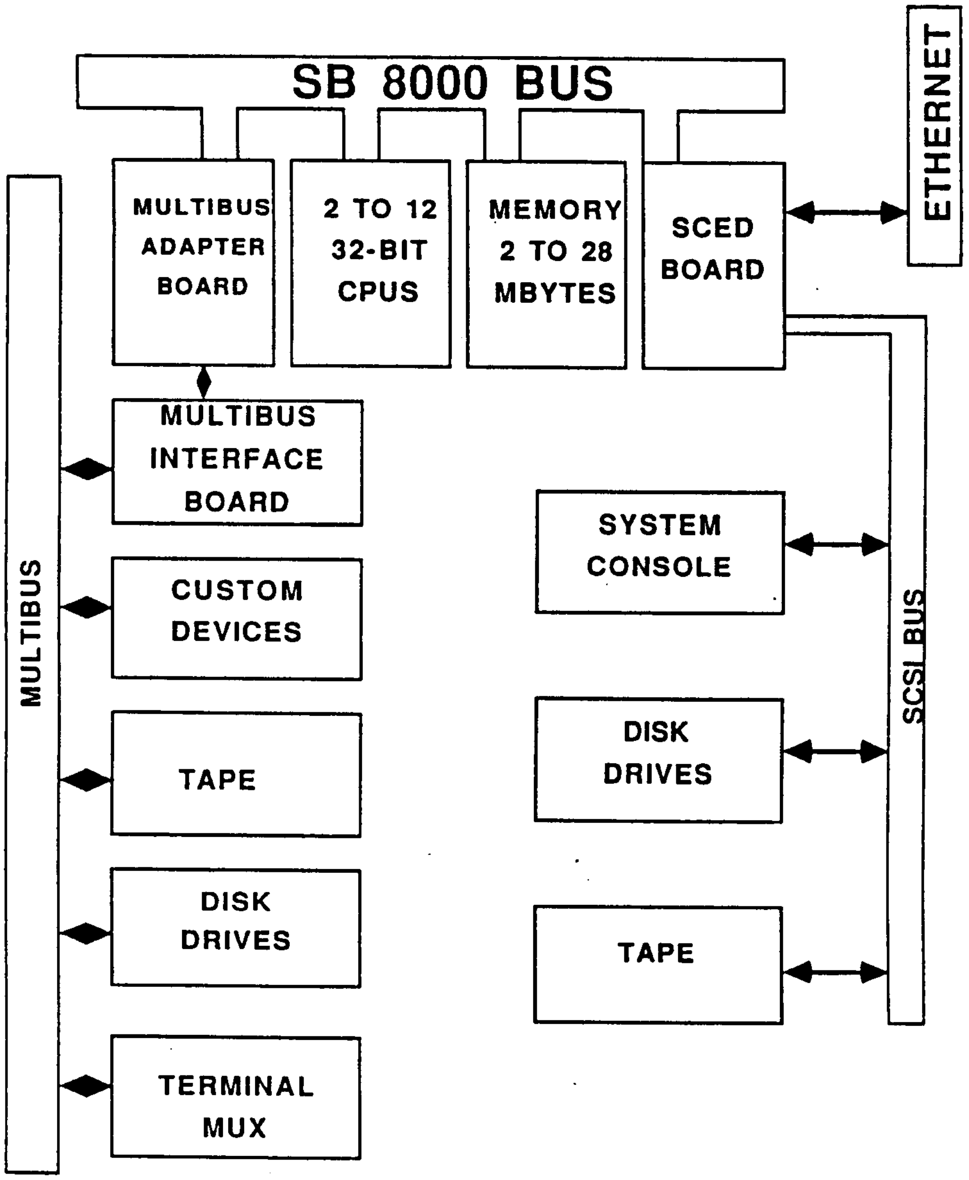


Fig.10 Sequent Balance 8000

called the SB8000. It is 32 bits wide and is used to connect the CPUs, memory, and I/O devices. Its simple structure eases the process of adding and removing components. More processors and memory may be added with minimal difficulty.

Data packets of 1, 2, 3, 4, and 8 bytes may be sent across the SB8000, which has a bandwidth of 40 Mbytes per second and a sustained data transfer rate of 26.7 Mbytes per second. This transfer rate is fairly high, but there are other machines with higher rates, such as the FX/8, produced by Alliant Computer Systems Corporation, which has a bandwidth of 188 Mbytes.

Processor Boards: The Balance 8000 contains between two and twelve National Semiconductor 32032 CPUs which are packaged two to a board. These chips run at 10 MHz. Recently, Intel 80386 boards have become available which make the machine run five or six times faster, but at a fairly high financial cost. The system used to run the parallelized match algorithm contains ten 32032 CPUs.

Each processor in the Balance 8000 has some added specialized hardware to make operations more efficient. The 8 Kbytes of cache memory available to each processor helps reduce bus contention, even though bus contention is still often a problem. Each processor also has another 8 Kbytes of local memory, as well as a floating point unit, and a memory management unit.

Memory Modules: Up to 28 Mbytes of physical memory may be used on the Balance 8000. This memory is located in up to four separate modules. Each of these memory modules contains 2 Mbytes of RAM used by the memory controller and an optional 2 to 6 Mbytes of extra memory. The reason the maximum total available is not 32 Mbytes is because one Mbyte of memory is reserved for use by up to four MULTIBUS adaptor boards.

Memory response time for a read request is 300 ns. By using the pipeline capabilities of the system, better performance than this may be achieved. Each process that is running has a limit of 16 Mbytes of virutal memory. This is obviously enough for most applications, but does present a problem for some very large programs since this limit is unchangeable.

Other Balance Components: The Sequent Balance 8000 also contains other components in its architecture. The System Link and Interrupt Controller (SLIC) is a single bit data path used for low level communications between system components. This bus is located within the SB8000.

Another Balance 8000 component is the Small Computer Systems Interface (SCSI) bus. This bus supports high volume, high speed data transfer between peripherals and memory. The SCSI is connected to the SB8000 through the SCSI/Ethernet/Diagnostics (SCED) board. The SCED also connects the Ethernet to the system, as well as provides a link for the system console.

As mentioned, up to four MULTIBUS adaptor boards may be used to connect external devices. Devices that may be attached include tape drives, printers, disk units, and terminals. Essentially any RS232-C compatible device may be connected through the MULTIBUS.

D. Summary

Parallel processing techniques and machines are currently very hot research topics. As more advanced parallel machines are designed and built, and as more sophisticated parallel programming techniques are discovered, problems which previously could not be handled with uniprocessor machines will be solved. One area being helped by parallel processing techniques is rule-based system shells. The following chapter looks at a particular rule-based system shell which does most of its matching in parallel. Significant speedups are achieved by dividing the work among the available processors. In this example, parallel processing is very valuable.

CHAPTER V

A PARALLELIZED MATCH ALGORITHM FOR USE IN A RULE-BASED SYSTEM SHELL

Two research topics which are currently receiving much interest are expert systems and parallel computing. Until recently, these two areas were fairly independent. This chapter discusses an attempt which was made to merge these research topics. A match algorithm which was designed for effective use on a true parallel machine and for rapidly changing match objects is described. The implementation details which were involved in incorporating this parallelized match algorithm into a rule-based system shell are also presented. The name given to the parallelized rule-based system shell is PMCLIPS. Test results of PMCLIPS are given which demonstrate the effectiveness of the parallelized algorithm in monitoring applications.

A. Background Information

Rule-based systems (RBS) have been used successfully in many applications, most noteably as interactive advisory systems. One area in which RBSs have not been utilized very successfully is in on-line monitoring problems. In applications which have the property of a rapidly changing fact base, such as in the monitoring problem, most RBSs slow down drastically. The reason for this can be traced back to the assumptions made when the matching algorithms for the RBSs were designed. Most matching algorithms used in RBSs assume the list of objects to be matched on each cycle will stay fairly constant. This assumption is true for most of the common problems handled by RBSs (i.e. interactive advisors). However, this assumption fails for monitoring applications. Even the most advanced matching algorithms used in RBS shells, such as the Rete match algorithm, suffer from performance loss in monitoring type problems. This is usually caused from an unnecessary storage of data.

Besides not performing well in monitoring type problems, the Rete match algorithm has another inherent limitation in that it is serial in nature. As described in Chapter III, the Rete match algorithm is not well suited for implementation on a true parallel computer. As a result, the potential speed increase of a parallel implementation is not available to the Rete match algorithm as it stands. Considering these limitations, it is obvious that the Rete match algorithm needs to be significantly redesigned in order for it to be effective in a RBS shell designed to handle monitoring problems in a parallel fashion. The following section discusses such a parallelized match algorithm.

B. PMA, a Parallelized Match Algorithm

This section describes a parallelized match algorithm called PMA which was designed to allow the RBS shell it is used in, PMCLIPS, to operate in a parallel fashion. PMA's design was influenced by the Rete match algorithm, and some of the internal data holding structures of the two algorithms are very similar. However, the flow of data through these structures differs greatly, as will be demonstrated.

Flowgraph Description: Figure 11 shows a flowgraph which describes at a fairly high level how PMA works. As mentioned in Chapter III, RBSs cycle through three steps: "select," "act," and "match." The flowgraph in figure 11 includes the "select" and "act" descriptions in addition to the "match" description since these directly affect the "match" actions handled by PMA. The most important

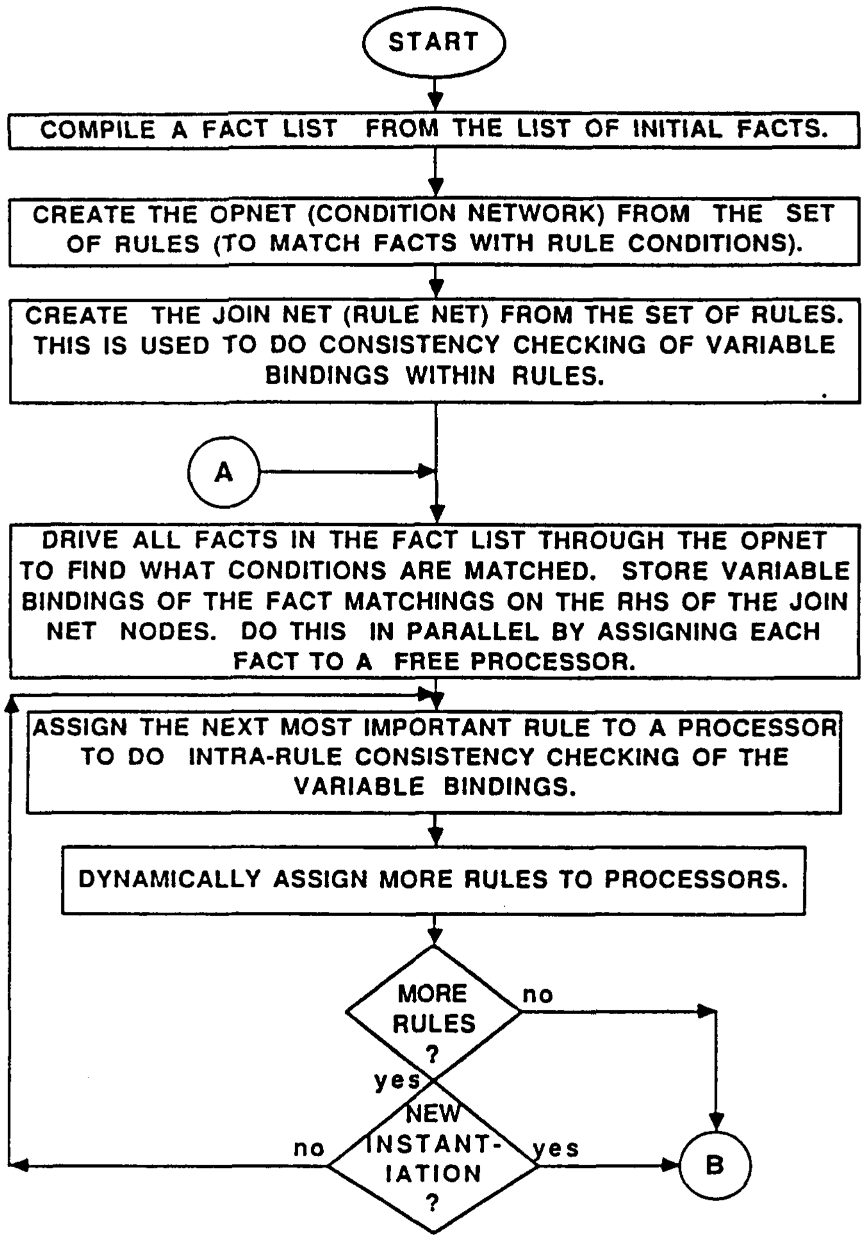

START
COMPILE A FACT LIST FROM THE LIST OF INITIAL FACTS.
CREATE THE OPNET (CONDITION NETWORK) FROM THE SET OF RULES (TO MATCH FACTS WITH RULE CONDITIONS).
CREATE THE JOIN NET (RULE NET) FROM THE SET OF RULES. THIS IS USED TO DO CONSISTENCY CHECKING OF VARIABLE BINDINGS WITHIN RULES.
A
DRIVE ALL FACTS IN THE FACT LIST THROUGH THE OPNET TO FIND WHAT CONDITIONS ARE MATCHED. STORE VARIABLE BINDINGS OF THE FACT MATCHINGS ON THE RHS OF THE JOIN NET NODES. DO THIS IN PARALLEL BY ASSIGNING EACH FACT TO A FREE PROCESSOR.
ASSIGN THE NEXT MOST IMPORTANT RULE TO A PROCESSOR TO DO INTRA-RULE CONSISTENCY CHECKING OF THE VARIABLE BINDINGS.
DYNAMICALLY ASSIGN MORE RULES TO PROCESSORS.
MORE RULES ?
no
yes
NEW INSTANT-IATION ?
no
yes
B

Fig.11 PMA Flowgraph

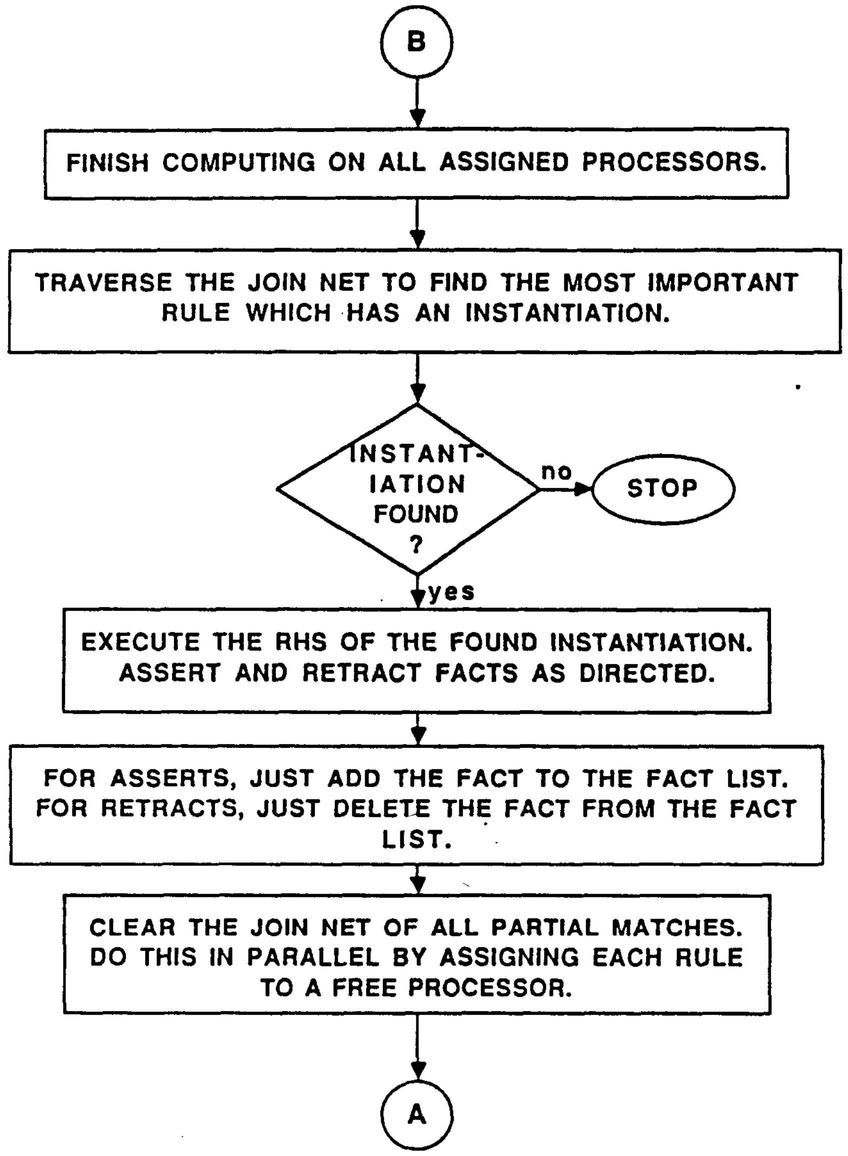

B
FINISH COMPUTING ON ALL ASSIGNED PROCESSORS.
TRAVERSE THE JOIN NET TO FIND THE MOST IMPORTANT RULE WHICH HAS AN INSTANTIATION.
INSTANT-IATION FOUND ?
no
STOP
yes
EXECUTE THE RHS OF THE FOUND INSTANTIATION. ASSERT AND RETRACT FACTS AS DIRECTED.
FOR ASSERTS, JUST ADD THE FACT TO THE FACT LIST. FOR RETRACTS, JUST DELETE THE FACT FROM THE FACT LIST.
CLEAR THE JOIN NET OF ALL PARTIAL MATCHES. DO THIS IN PARALLEL BY ASSIGNING EACH RULE TO A FREE PROCESSOR.
A

Fig.11 Continued

step from a performance point of view is the match step, since this is where the majority of the run time is spent. The description given of PMA overlaps the discussion of PMCLIPS which is provided later, since the two are so extensively intertwined.

PMA expects two inputs: a list of constant objects (a fact list) and sets of patterns (conditions). PMA's job is to continually match all the constant objects with the sets of patterns as the constant objects are changed. PMA itself does not change the list of constant objects – this is performed by the act step of the RBS cycle. However, PMA's actions help the act step know what changes to make.

The first task performed by PMA is the creation of an internal software structure to hold the initial constant object list which is provided to PMA. This object list will change through time with elements being added and deleted according to the act step of the RBS cycle. A simple linked list works well for storing the object list.

Next, PMA takes the provided sets of patterns to create an opnet (condition net). In a RBS, the sets of patterns correspond to the condition sets of rules. Separate groups of patterns are provided to PMA, which are used to build the opnet, as well as the join net described later. The opnet is designed to make the process of matching individual constant objects with a set of patterns more efficient.

Patterns may contain variable elements which match zero or more elements inside a single constant object, as in the Rete match algorithm described in Chapter III. Because of this flexibility, the matching process may mushroom if excessively general patterns are to be matched. Figure 6 shows an example opnet. Its physical structure is the same as that used in the Rete match algorithm, but how and when it is used differs in PMA from the Rete match algorithm.

The opnet is basically a compiled indexing structure which is helpful in reducing the amount of searching which is required in matching constant objects with variable patterns. Objects are inserted at the head of the opnet structure and compared, element by element, with the nodes of the opnet. When a mismatch is found, a redirection in the search is made to effectively cut off branches which are unfruitful. Whenever a complete successful match is found between an object and pattern, the opnet provides a pointer over to the join net so that the match information can be temporarily stored. It is possible for a single object to match several different patterns since variable elements are allowed within pattern descriptions. How PMA uses the opnet is shown later.

The next step taken by PMA is to build the join net from the set of patterns it is given. The join net used by PMA is similar to the join net used by the Rete match algorithm, but the flow of data is quite different. Figure 12 shows an example join net which is the PMA version of figure 7. The example CLIPS program of Chapter III would generate a PMA join net which looks like figure 12. Figure 4 gives the source code of the example program.

The join net is used to hold matching information between individual objects and patterns. It is also utilized in making sure that variable bindings within a set of patterns (i.e. conditions in a rule) are consistent. That is, only one constant element may be bound to a specific variable name. The set of patterns is satisfied only if all patterns match an object in the list of objects, and if only one constant element is bound to each pattern variable.

The join net in figure 12 depicts a set of nodes (squares) which represent individual patterns from a group of patterns. The right-hand side (RHS) of a

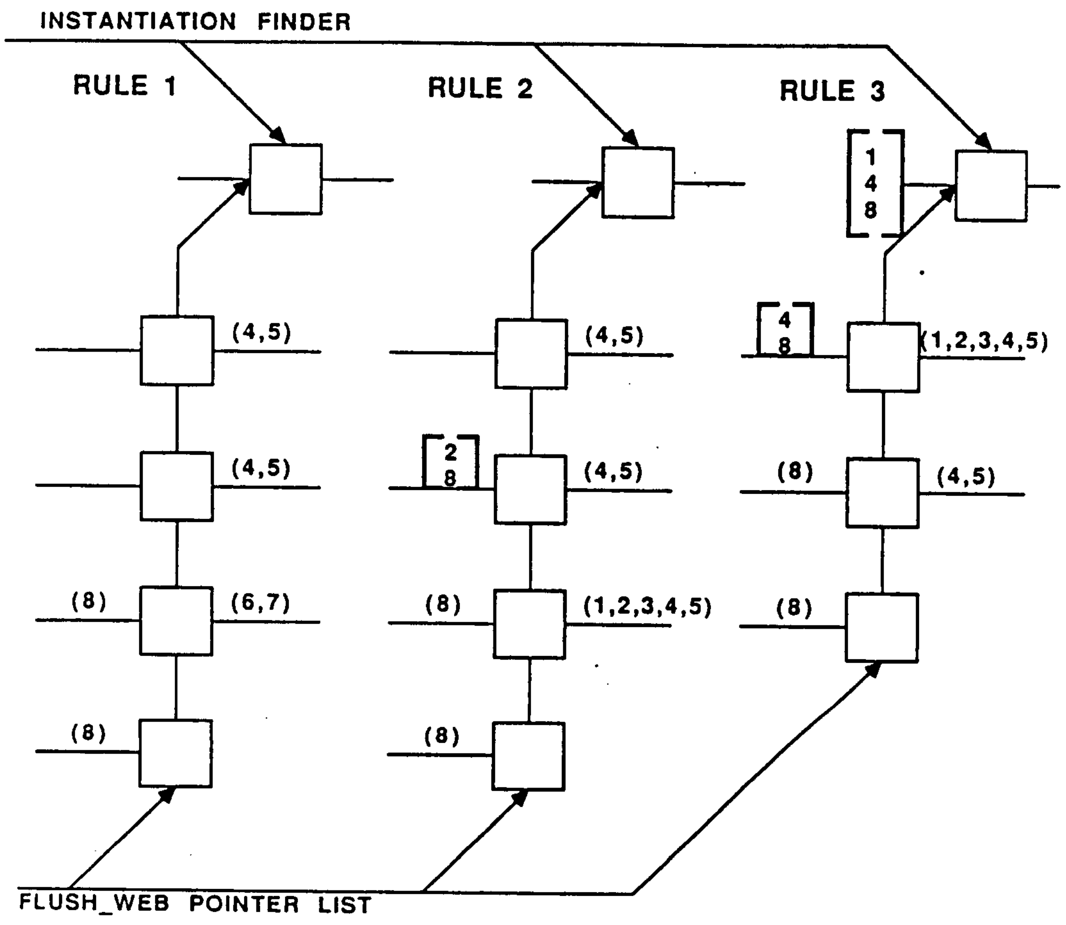


Fig.12 PMA Join Net

node holds information about matches between objects and the particular pattern represented by the node. This location is pointed to by the opnet when a successful match is made when going through the opnet. The left-hand side (LHS) of a node is used to hold the set of variable bindings which are consistent up to that point. Variable binding information on the LHS and RHS are compared to see if a consistent set can be created. If one can be, it is sent up to the next node. When and how this is done is a key difference between PMA and the Rete match algorithm. Whenever a set of consistent variable bindings reaches the last (top) node in a set of patterns, then that set is said to be satisfied and an instantiation exists. The flow followed by PMA will soon be expounded further.

After the list of objects, the opnet, and the join net are created, PMA then drives all the constant objects through the opnet to find which patterns are matched. As mentioned, all match information is stored on the RHS of the join net nodes pointed to by the opnet. PMA is more efficient than the Rete match algorithm in performing this operation because it is able to feed constant objects through the opnet in parallel. The Rete match algorithm can only compare one object at a time, and when it does, it continues on with that one object taking it as far as possible in trying to find new instantiations. That is, the Rete match algorithm does not drive all objects to the RHS of the join net at once as PMA does. This is due to the basic difference in the data flow of the two algorithms. The Rete match algorithm is designed for efficient use on a uniprocessor machine and is unable to take advantage of multiple processors as PMA is.

PMA performs this first phase of parallel operation by using dynamic scheduling discussed in Chapter IV. Each individual constant object is assigned to a processor whenever one becomes available. When a processor receives an object to drive through the opnet, it does so, and when it finishes it returns to be assigned a new object to push through. Different objects will in general take different amounts of time to push through the opnet, so by using dynamic scheduling, processors will not be waiting around idle. All the processors will stay busy.

When PMA finishes pushing all the constant objects to the RHS of the join net nodes in parallel, it then goes on to the second phase of parallel operation. This phase employs dynamic scheduling as does the first parallel operation. In this step, free processors are continually assigned to sets of patterns until no more sets of patterns are left to be processed, or until a new instantiation has been found. Sets of patterns with higher importance (salience) are processed first.

When a processor receives a set of patterns to work on, it begins operation by looking at the bottom node which corresponds to the first pattern in the set of patterns. If this node has no variable bindings attached to it, then no more processing may be done and the processor is finished with that set of patterns. If, however, at least one set of variable bindings exists, then one is extracted and moved up to the LHS of the next node up on the list of nodes. At this point, consistency checking is performed between the variable bindings on the LHS and RHS of that node to see if a new consistent set of variable bindings may be constructed. If so, then they are shipped up to the next node for a similar operation to be performed. This process continues until the top node is reached, which means an instantiation has been found, or until no new set of consistent variable bindings can be generated for shipping up to the next node.

If an instantiation is discovered, then a special check is made to determine what to do next. PMA contains a structure which holds history information about the last n instantiations which were used to change the list of objects. The value of n is variable and is set by the user to any desired value. If the newly discovered instantiation is identical to one of the instantiations in the history structure, then the new instantiation is ignored and not used. Processing continues until another instantiation is found which is new, or until all of the variable binding sets at the bottom node are exhausted. The history feature is useful in guarding against undesired looping errors during run time operation.

If the discovered instantiation has not been used during the last n cycles, then it is stored. At this point, the processor is finished working on the set of patterns. Since only instantiations of lower importance could possibly be found in the rest of the sets of patterns, no more processors are assigned to process these sets of patterns. This observation saves a lot of processing time in general.

Whenever a new instantiation is not discovered, control goes back to the bottom node so that the process can start over with a new set of variable bindings which are attached to the node. If no more variable bindings exist, the processor is finished working on that set of patterns and returns to the main process to be reassigned to a new set of patterns. If another processor working in parallel has already found an instantiation, then the free processor will not be assigned a new set of patterns to work on.

The dynamic scheduling employed allows several sets of patterns to be checked for instantiations concurrently. One special consideration which must be insured is that whenever a processor is assigned to a set of patterns, it must not be interrupted until it finishes by finding a new instantiation or by running out of variable bindings to work with. This is important because the sets of patterns are fed out in descending order of importance. If a processor working on a set of patterns with a relatively low level of importance finds a new instantiation, another processor still working on a set of patterns with a higher level of importance must be allowed to finish or else an instantiation of lower importance may be picked during the select step. It is assumed that the most important instantiation will be selected on each cycle, and in PMA it is.

The Rete match algorithm also uses a join net to find instantiations, but when it finds one, it puts it on an agenda ranked by salience value. PMA does not have to do this since only one instantiation is needed per cycle. The Rete match algorithm inherently excludes the repeated firing of identical instantiations by its structure, but PMA achieves the same effect with slightly more flexibility by allowing the user to specify how far back previous instantiations are to be checked. PMA and the Rete match algorithm are quite different in this respect.

At this point, PMA steps aside momentarily to allow the select operation to be performed. This job can be done very quickly since PMA sets up the data in a convenient form. The top nodes in the sets of patterns in the join net are checked in descending order of importance. The first set of patterns found which has an instantiation stored in it is the one chosen since it is the most important one. If no instantiation is found, then the program stops. This operation is very different than the Rete match algorithm's use of a potentially long agenda.

The act step is performed next. The actions to be taken are dictated by the selected instantiation. Typically a series of assertions and retractions are made. In the Rete match algorithm, an assertion involves a long sequential series of steps which entails going through the opnet, join net, and agenda as far as possible. However, in PMA, an assertion involves simply the addition of a new object to the object list, which takes very little time.

In the Rete match algorithm, a retraction takes a lot of time since every

ORIGINAL PAGE IS
OF POOR QUALITY

pattern which matches the object being retracted must be located. Once these patterns are located, a linear search through the join net and agenda must be made so that all partial matches and instantiations which use the object may be deleted from the system. When several retractions are required per cycle, as in monitoring problems, much time is consumed. This is the place where most of the time savings of PMA come from. For retractions, PMA simply deletes the object from the object list. The price it pays in reasserting each object on each cycle is not as severe as the price paid by the Rete match algorithm in retracting large numbers of objects per cycle.

After the act step is finished, the final portion of PMA is executed. This is the third and final distinct parallel section of code which is run. The job that is performed is essentially a cleanup operation. The entire join net is cleared of all structures that were placed on it during the other two parallel execution phases. Clearing of structures refers to returning dynamically allocated memory for reuse at a later time. This job is dynamically scheduled since the time needed to clear one set of patterns may be very different than the time needed to clear another set of patterns. As in the second phase of parallel execution, free processors are assigned to sets of patterns. When a processor finishes with one set, it goes back to be assigned to a new set to clear. When all join net nodes are clear, control returns to the first phase of parallel execution in which all constant objects are driven through the opnet, and the cycle continues.

C. Implementation Details of PMCLIPS

PMCLIPS is a modification of CLIPS 3.11. CLIPS was developed at NASA and uses a version of the Rete match algorithm in performing matching operations.

Matching is the main portion of CLIPS which was reworked in creating PMCLIPS, although the select and act functions are also quite different. In this section, the implementation details of PMCLIPS are given, along with the differences between PMCLIPS and CLIPS.

Operational Information: The first two letters of PMCLIPS refer to the program's designed purpose of being used on a parallel computer for monitoring applications. The version of PMCLIPS which has been implemented runs on the Sequent Balance 8000 which can contain up to 12 processors. The Balance 8000, as described in Chapter IV, uses a shared memory architecture and a single shared bus. Other shared memory machines could be used to run PMCLIPS with minor adjustments in many cases. The Appendix contains the main portion of the run time code of PMCLIPS. Code used for parsing and other static functions do not vary much between CLIPS and PMCLIPS and are not included in the Appendix.

When writing rule sets to run on PMCLIPS, the syntax followed is identical to that followed in writing rule sets for CLIPS, with two minor exceptions. The first exception is that in CLIPS, the importance (salience) of a rule is indicated by assigning a numerical value to the salience slot of a rule definition. In PMCLIPS, the relative importance of a rule is determined by its position within the set of rules. Rules which are listed earlier have more importance than rules which come later. This means if during a cycle two or more rules have valid instantiations, the rule with the highest indicated importance will be the one to fire, since only one rule may fire per cycle in RBSs.

The second syntactical difference between CLIPS and PMCLIPS concerns what is allowed as the first condition in a set of rule conditions. In CLIPS, any type of condition may occupy the first slot; however, in PMCLIPS, the first condition may not be a negated pattern. That is, in PMCLIPS, a rule should not be written so that the first pattern is satisfied only when no fact in the fact base is found to match the pattern. As mentioned, this is a very minor difference.

The use of PMCLIPS is interactive in nature. Once the program has been started, it prompts the user for all the inputs it needs. The first input PMCLIPS expects is the name of a file which contains a rule set to be loaded in for eventual running. At this point, the user has some options. He may reset the rule set to prepare it for running or he may use some other user interface feature. It is possible to display the current facts and rules in the system at any time between cycles. Its is also possible to interactively toggle switches which cause extra information about what is happening during run time to be printed on the screen, such as what facts are currently being added or deleted, and which rules are having instantiations produced.

When a reset is invoked by the user, PMCLIPS then asks for the number of processors to use. Any number of processors may be chosen, as long as the number of processors on-line is not exceeded. In actual practice, it is advisable to not use more than one less than the number of processors in the system. Leaving one processor free for the system to use for miscellaneous operations helps reduce scheduling problems.

Once the number of processors to use is input by the user, PMCLIPS then asks for the number of previous rule firings to store in a history structure. This history structure, which in software is a bi-directional ring, holds the rule name and associated facts of the previous n rules which fired. The user supplies the value of n, and it will vary according to the type of rule set to be run. Usually n will be set to zero since most rule sets do not try to loop.

After a reset, PMCLIPS is ready to run. The run command is then given with an optional number of cycles to go through. If the number of cycles to execute is not supplied, the system runs until no more instantiations can be found.

Parallel Coding: By examining a portion of the coding of PMCLIPS, the manner in which parallelism is utilized may be seen. Figure 13 shows cycle_reset(), a high level function which is run concurrently by all available processors during run time.

Cycle_reset() contains three main sections which are executed in parallel. The first section is used to remove all bindings from the join net. This piece of code is run until all rules in the join net have been cleared of unnecessary software structures. The m_next() funtion call is used to prevent different processors from doing the same job. Every time m_next() is called, it returns an integer one greater than the integer it returned the last time it was called. By using this mechanism, calls to flush_web() can be differentiated according to the argument passed to it. The argument corresponds to the rule to be cleared of structures. This is one way of achieving dynamic scheduling. Processors execute the flush_web() code until finished and then return to find which rule to flush next. When all rules have been flushed, processors wait until all the other processors are finished. The m_sync() function is responsible for synchronizing the processors as described.

The next parallel job to be done is the pushing of all the facts through the opnet. The facts are kept in a linked list so an extra variable is needed to keep track of which facts have already been assigned and which still need to be assigned to processors. The functions s_lock() and s_unlock() are used to make sure only one processor at a time runs the code between these two calls. This part of the code is known as a critical region. In this case, only one processor at a time may

```
/*******************************************************************/
/* CYCLE_RESET: This is the function called to initiate all the   */
/*   parallel processing.  The first step is to dynamically       */
/*   allocate processors to flush the join net of all structures */
/*   currently attached (from a previous cycle).  When all rules  */
/*   in the join net are flushed, the processors synchronize and  */
/*   and then are dynamically fed facts to push through the       */
/*   opnet.  After another synchronization, the processors are    */
/*   then dynamically assigned rules in the join net to drive the */
/*   facts through to do intra-rule consistency checking.         */
/*******************************************************************/
void cycle_reset()
  {
   int loc_num_rules;
   struct fact *fact;
   int local_finish;
   int rule_num;

   if (watch_rules)
     {
      m_lock();
      printf("Process %d\n", myid);
      m_unlock();
     }

   loc_num_rules = num_rules;

   /*========================================*/
   /* Remove all bindings from the join net. */
   /*========================================*/

   while ((rule_num = m_next()) <= loc_num_rules)
     flush_web(rule_num);

   m_sync();
```

Fig.13 Cycle_Reset Code

```
  /*================================================*/
  /* Loop through each fact in the fact list and   */
  /* push it through the opnet.                    */
  /*================================================*/

  local_finish = FALSE;

  while (!local_finish)
    {
     s_lock(&lock_com1);
     if (traverse != NULL)
       {
        fact = traverse;
        traverse = traverse->next;
        s_unlock(&lock_com1);
        compare(fact,fact->word,new_patn_list,1,1,NULL,1);
       }
     else
       {
        s_unlock(&lock_com1);
        local_finish = TRUE;
       }
    }

  m_sync();

  /*=========================================================*/
  /* Drive each variable binding at the empty nodes through */
  /* the join-net until an instantiation is found or else   */
  /* no more bindings are left.                              */
  /*=========================================================*/

  while ((cont_step == TRUE) &&
         ((rule_num = m_next()) <= loc_num_rules))
    step_drive(rule_num);

}
```

Fig.13 Continued

update the variable pointer which tells which facts have already been assigned. If this were not protected, two processor could potentially try to change the pointer simultaneously with unknown results.

After all facts have been driven through the opnet simultaneously, the processors are resynchronized using m_sync(). The last parallel job is to do consistency checking of variable bindings in the join net, which is handled by step_drive(). Dynamic scheduling is again used with different rules going to different processors. As in the first parallel section, m_next() is used to count down the number of rules to assign, as well as to keep two processors from working on the same rule.

As this code shows, PMCLIPS makes extensive use of parallelism, whereas CLIPS, as it stands, is not capable of doing the same. Roughly speaking, what has been done is CLIPS's long serial flow has been chopped up so that each piece may be executed in parallel in PMCLIPS. The test results given in the next section show that this restructuring of CLIPS is very worthwhile in certain types of applications.

D. Test Results

Table II provides a comprehensive listing of the performance results of several different rule sets run on PMCLIPS and CLIPS. For PMCLIPS runs, results are reported for runs corresponding to the number of processors used. Between one and nine processors were used for PMCLIPS, but CLIPS could only use one processor because of the way it is designed.

The values in the table are normalized by rule set to show the ratio of the execution time of a run to the fastest run time achieved. For example, looking at

TABLE II.
Relative Performance of CLIPS and PMCLIPS

	RELATIVE EXECUTION RUN TIMES OF PMCLIPS AND CLIPS NORMALIZED BY FASTEST PMCLIPS RUN TIME IN EACH ROW									
	PMCLIPS (Number of Processors Used)									CLIPS
Rule Set	1	2	3	4	5	6	7	8	9	Serial
POWERMON	3.76	2.09	1.53	1.26	1.12	1.06	1.00	1.00	1.00	1.41
MAB	4.17	2.33	1.67	1.33	1.17	1.17	1.00	1.17	1.17	0.83
MAB5	4.38	2.38	1.77	1.46	1.31	1.15	1.08	1.00	1.08	4.62
MAB10	4.45	2.41	1.77	1.41	1.23	1.09	1.14	1.00	1.00	5.64
MAB15	4.05	2.22	1.59	1.27	1.14	1.03	1.03	1.00	1.03	5.73
MAB20	4.06	2.23	1.63	1.37	1.29	1.15	1.10	1.00	1.06	6.04
MAB25	4.06	2.21	1.61	1.39	1.23	1.13	1.09	1.00	1.09	6.34
MAB30	4.12	2.25	1.64	1.36	1.19	1.22	1.13	1.00	1.15	6.79

the MAB5 rule set, it is seen that the fastest run time is achieved by PMCLIPS using eight processors. The time required by CLIPS to run this rule set is 4.62 times as long as the time required by PMCLIPS using eight processors. In this example, PMCLIPS using one processor is still faster that CLIPS.

The MAB rule set is derived from the classic AI monkey-and-banana problem. In this problem, a monkey is placed in a room with a ladder, a banana, and assorted couches, pillows, keys, and chests. The goal of the monkey is to move objects around and open chests until the banana is acquired and eaten. The rule set written to perform this operation contains 33 rules with an average of 3.2 conditions per rule. PMCLIPS was designed to be more efficient with rule sets which have a higher number of conditions. For this reason, CLIPS is able to outperform PMCLIPS in the MAB rule set. The fastest PMCLIPS run is about 17% slower than CLIPS, which is designed to handle this type of rule set.

The rule sets named MABn, where n is a number, are modifications of MAB. For example, MAB5 runs the same sequence of steps that MAB does, but each rule has an additional five conditions to match. MAB10 has ten extra conditions, and so on. The data in Table II backs up the claim that PMCLIPS is better suited for rule sets with a large number of conditions than is CLIPS. In all MABn rule sets, PMCLIPS runs faster than CLIPS, even when only one processor is used. When more than one processor is used, the speed difference increases further.

The other rule set tested was POWERMON. This rule set has 23 rules which contain an average of 10.7 conditions per rule. An algorithm which was developed by a power systems group at Texax A&M is implemented in this rule set. POWERMON's job is to monitor the energy levels in different frequency bands of electrical power lines. Disturbances and events are recorded as part of the monitoring process. Simulated data was input to POWERMON producing the results seen in Table II. When PMCLIPS is used with four or more processors, it runs faster than CLIPS. Compared with PMCLIPS using seven or more processors, CLIPS requires about 41% more time to run. This is not quite as good as the 6.79 factor difference in time in the MAB30 rule set, but is still significant.

Another piece of information which may be extracted from the data in Table II is how the number of conditions in a rule set affects the relative performance of CLIPS and PMCLIPS. Figure 14 shows a plot of the ratio of CLIPS run time to the fastest PMCLIPS run time versus the average number of conditions in a rule set. By using the MAB rule sets, this plot may be generated. The conclusion reached by examining this graph is that the more conditions present, the better PMCLIPS does relative to CLIPS. The improvement tends to level off at high values, but is still present.

Another graph is shown in figure 15. This plot demonstrates the effects of adding more processors to a PMCLIPS run. MAB5 is chosen as the example, but all test runs look very similar. In this plot, MAB5's normalized run time is plotted versus the number of processors used. The PMCLIPS run time for one processor is normalized to 100. The CLIPS run time is drawn on the plot as a straight line for comparison.

As the plot shows, adding a second processor cuts the run time almost in half. And adding a third processor significantly improves performance. However, after four or five processors have been added, the extra processors are almost not worth adding. This result is not surprising since the Sequent Balance 8000 has only a single bus. When several processors try to access memory, all but one must wait, which wastes time. This is a major problem with having just one bus.

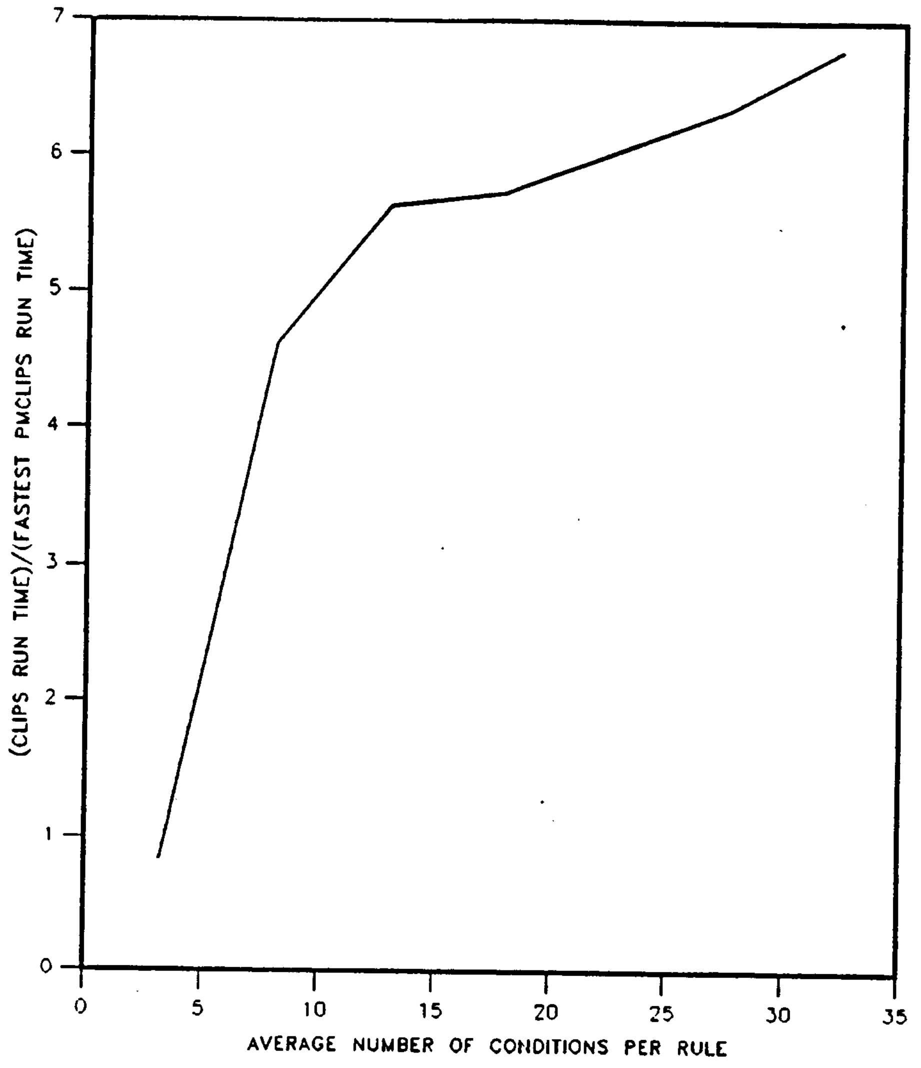


Fig.14 PMCLIPS Relative Performance Versus Number of Conditions

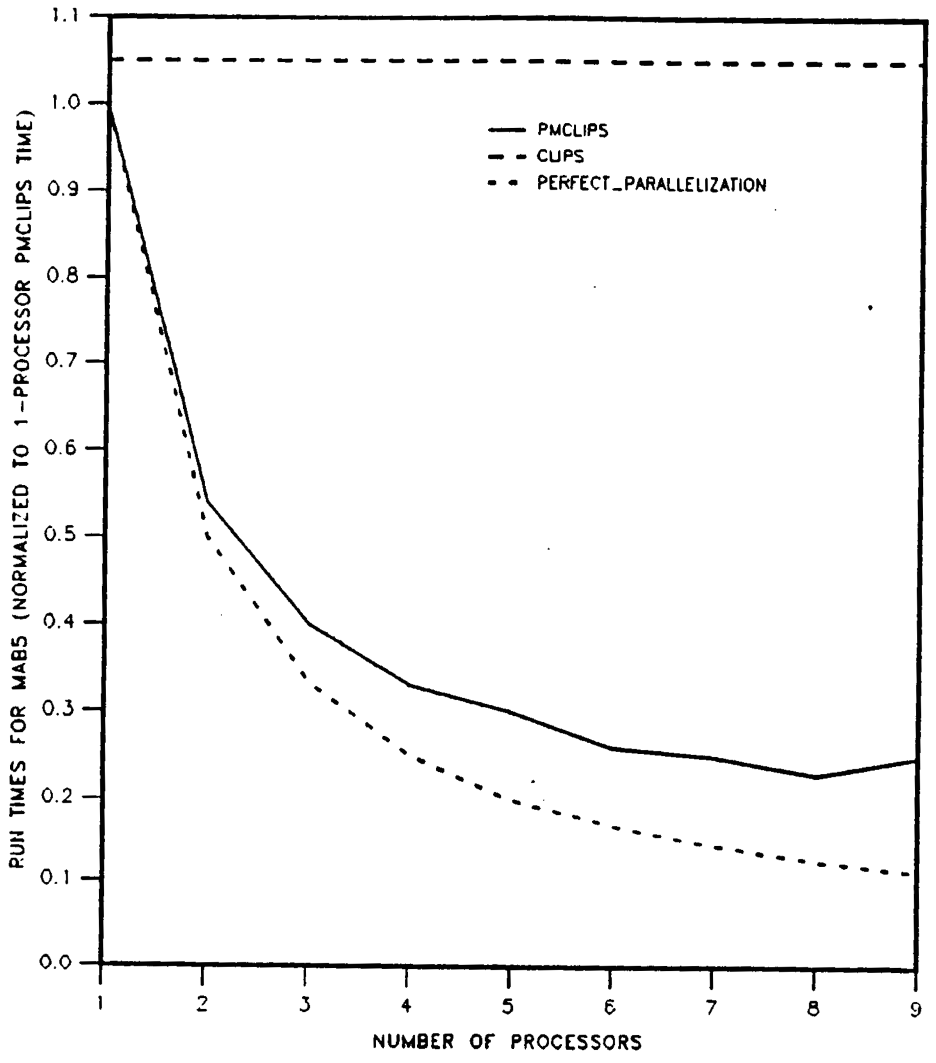


Fig.15 PMCLIPS Performance Versus Number of Processors

Overall, the test results speak very well for PMCLIPS. Its design goal of improving CLIPS for rule sets with several conditions has been met. PMCLIPS obviously makes use of parallelism, whereas CLIPS does not. In the extreme example of MAB30, where the improvement in the time factor is 6.79, PMCLIPS looks very good.

CHAPTER VI

SUMMARY AND CONCLUSIONS

The objective of this research as stated in Chapter I was the development of a matching algorithm to be used in a rule-base system shell. This matching algorithm was to have two specific properties. First, it was to be capable of taking advantage of a multiprocessor computer. Secondly, it was to make the rule-based system shell it was used in run faster than a standard rule-based system shell for problems in which match objects change rapidly.

Chapter II showed the present status of systems which claim to be real-time expert systems. The interest of industry and academia in this area was discussed. Techniques which could be used in attempting to integrate real-time systems and expert systems were also examined. The objective of this research, which has been reiterated above, ties in directly with the goal of developing real-time expert systems. By effectively utilizing parallelism in speeding up a rule-based system shell, a new tool is created by which more real-time expert systems may be developed. Any speedup whatsoever which can be achieved in a rule-based system shell results in a product which is then able to handle certain problems which before were incapable of being solved.

In developing the new parallelized match algorithm, PMA, a foundation was needed to begin the work and also to have a reference against which comparisons could be made. The match algorithm which was used as a starting point was the Rete match algorithm described in Chapter III. This match algorithm is considered by many to be state of the art. It has two main weaknesses, however, since it does not take advantage of parallelism, and since it slows down when used with a rapidly changing fact base. Monitors have the property of utilizing a rapidly changing fact base, so a replacement for the Rete match algorithm for this type of problem was needed.

The technique which was followed in attempting to speed up PMA for monitoring problems was parallelism. Chapter IV discussed ways in which parallelism may be employed effectively in different types of problems. Various methods of parallel programming were discussed. The method most used in PMA is dynamic scheduling, which entails some extra overhead in allowing processors to communicate, but also has advantages in that it is able to keep all available processors busy working. Chapter IV also looked at different parallel hardware architectures, since this is a major factor in determining how well a particular parallel application will perform. The target machine of the parallelized rule-based system shell was the Sequent Balance 8000, therefore, this shared memory architecture machine was looked at in more detail.

CLIPS, a rule-based system shell developed by NASA, was extensively modified in producing a new rule-based system shell called PMCLIPS. Chapter V gives the details of PMCLIPS, as well as the matching algorithm used in it, PMA. The syntax of PMCLIPS rule sets is essentially identical to the syntax of CLIPS rule sets, but the internal operation of the two is quite different. CLIPS is very sequential in its operation, whereas PMCLIPS saves much time by splitting up its work among available processors.

The test results presented in Chapter V show how effectively PMCLIPS works for monitoring problems. In one optimal example, PMCLIPS runs 6.79 times faster than CLIPS. In that example, each rule has several conditions, which is what PMCLIPS was designed to handle more efficiently. In a more realistic example, PMCLIPS was 41% faster, which in some applications is very significant.

When extra processors were used in PMCLIPS runs, the response times notably increased for the first few processors added. After five or six processors were added, the increase in response time was not very appreciable. The reasoning behind this is fairly obvious when the architecture of the underlying computer is examined. The Sequent Balance 8000, which was the machine used, is a shared memory machine with only a single bus. The shared memory architecture seemed to be well suited for PMA, but having only a single bus restricted the speedup which may have been achieved. When more than one processor needed to access memory, all but one had to wait until the bus was free. When several processors were working, much time was wasted.

Future work in this research topic could focus on how to improve PMCLIPS's performance further. One area which would appear to be promising is the effects of different parallel machine architectures on run time. For example, it would seem that a shared memory architecture which had a multiple bus interconnection network could potentially allow many more processors to operate with less bus contention. This would result in a much better response time. Also, PMCLIPS, as it stands, contains many user friendly features inherited from CLIPS which possible could be eliminated to improve the run time.

This work achieved its objective of developing a parallelized match algorithm for use in a rule-based system shell which works better than standard rule-based system shells for monitoring type applications. Not all real-time expert system problems are solved by this work, but some applications not handled previously are closer to a solution. In this sense, PMA and PMCLIPS provide a meaningful step in the advancement of real-time expert systems.

REFERENCES

[1] Charles L. Forgy, "Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem," *Artificial Intelligence*, vol. 19, pp. 17-37, Sept. 1982.

[2] Stephen J. Andriole, *Applications in Artificial Intelligence*. Princeton, NJ: Petrocelli Books, Inc., 1985.

[3] Mark H. Houck, "Designing an Expert System for Real-Time Reservoir System Operation," *Civil Engineering Systems*, vol. 2, pp. 30-37, Mar. 1985.

[4] Gerald Jay Sussman, et. al., "Scheme-79 – Lisp on a Chip," *IEEE Computer*, vol. 14, pp. 10-21, July 1981.

[5] Edwin M. Drogin, "AI Issues for Real Time Systems," *Defense Electronics*, vol. 15, pp. 71-79, Aug. 1983.

[6] Hidehiko Tanaka, "A Parallel Inference Machine," *IEEE Computer*, vol. 19, pp. 48-54, May 1986.

[7] Kenneth W. Goff, "Artificial Intelligence in Process Control," *Mechanical Engineering*, vol. 10, pp. 53-57, Oct. 1985.

[8] Cindy A. O'Reilly and Andrew S. Cromarty, "'Fast' Is Not 'Real-Time': Designing Effective Real-Time AI Systems," in *Proceedings of SPIE 548 – Applications of Artificial Intelligence II*, Arlington, VA, 1985.

[9] D. C. Evers, D. M. Smith, and C. J. Staros, "Interfacing an Intelligent Decision-Maker to a Real-Time Control System," in *Proceedings of SPIE 485 – Applications of Artificial Intelligence*, Arlington, VA, 1984.

[10] R. L. Ennis, J. H. Griesmer, S. J. Hong, et. al., "A Continuous Real-Time Expert System for Computer Operations," *IBM Journal of Research and Development*, vol. 30, pp. 14-17, Jan. 1986.

[11] Steven Henkind, Louis Teichholz, and Malcolm Harrison, "Intensive Care Unit Monitoring Using a Real-Time Expert System," in *Proceedings of IEEE Computers in Cardiology*, Salt Lake City, UT, 1984.

[12] P. A. Sachs, A. M. Paterson, and M. H. M. Turner, "Escort – An Expert System for Complex Operations in Real Time," *Expert Systems*, vol. 3, pp. 22-29, Jan. 1986.

[13] M. Lattimer Wright, et. al., "An Expert System for Real-Time Control," *IEEE Software*, vol. 3, pp. 16-24, Mar. 1986.

[14] David Leinweber and John Perry, "Controlling Real-Time Processes on the Space Station with Expert Systems," in *Proceedings of SPIE 729 - Space Station Automation II*, Cambridge, MA, 1986.

[15] Joe K. Clema, Richard Werling, and Alhad Chande, "Expert Systems for Real Time Applications," in *Proceedings of IEEE NAECON*, Dayton, OH, 1985.

[16] Carla M. Wong, Richard W. Crawford, John C. Kunz, and Thomas P. Kehler, "Application of Artificial Intelligence to Triple Quadrupole Mass Spectrometry (TQMS)," *IEEE Transactions on Nuclear Science*, vol. 31, pp. 804-810, Feb. 1984.

[17] N. K. Gupta and R. E. Seviora, "An Expert System Approach to Real Time System Debugging," in *Proceedings of the First IEEE Conference on Artificial Intelligence Applications*, Stanford, CA, 1984.

[18] William R. Nelson, "REACTOR: An Expert System for Diagnosis and Treatment of Nuclear Reactor Accidents," in *Proceedings of the National Conference on Artificial Intelligence*, Pittsburgh, PA, 1982.

[19] Chris Culbert, *CLIPS 3.0 Reference Manual.* Clearlake, TX: NASA Artificial Intelligence Section, Johnson Space Center, 1986.

[20] Bruce K. Hillyer and David Elliot Shaw, "Execution of OPS5 Production Systems on a Massively Parallel Machine," *Journal of Parallel and Distributed Computing*, vol. 3, pp. 236-267, Jan. 1986.

[21] Robert J. Douglas, "A Qualitative Assessment of Parallelism in Expert Systems," *IEEE Software*, vol. 2, pp. 70-81, May 1985.

[22] Sequent Computer Systems, Inc., *Balance Guide to Parallel Programming.* Beaverton, OR: Sequent Computer Systems, Inc., 1986.

[23] Laxmi N. Bhuyan, "Interconnection Networks for Parallel and Distributed Processing," *IEEE Computer*, vol. 20, pp. 9-12, June 1987.

[24] Carl D. Howe and Bruce Moxon, "How To Program Parallel Processors," *IEEE Spectrum*, vol. 24, pp. 36-41, Sept. 1987.

[25] Alan H. Karp, "Programming for Parallelism," *IEEE Computer*, vol. 20, pp. 43-57, May 1987.

[26] Paul Wiley, "A Parallel Architecture Comes of Age," *IEEE Spectrum*, vol. 24, pp. 46-50, June 1987.

[27] Floyd Davenport, *The Concurrent Environment of the Sequent Balance 8000.* College Station, TX: Computer Science Department, Texas A&M University, 1987.

APPENDIX

```
/*********************************/
/* PMCLIPS Version 1.0   10/1/87 */
/*********************************/

/*****************************************************************/
/* PMCLIPS is a modification of CLIPS 3.11.  CLIPS was developed  */
/* by NASA and uses a version of the Rete match algorithm in its  */
/* matching techniques.  PMCLIPS uses a different matching algo-  */
/* rithm which was designed to be more efficient when used with   */
/* rule sets which contain rules with a relatively large number   */
/* of conditions.  The matching algorithm used in PMCLIPS also    */
/* has the quality of being naturally adaptable to a parallel     */
/* processing implementation.  In fact, this version of PMCLIPS   */
/* is designed to run on a true parallel machine, the Sequent     */
/* Balance 8000.  Other parallel machines could be used to run    */
/* this code with minor adjustments in many cases.                */
/*                                                               */
/* In using this program, an input rule set has the exact same    */
/* syntax as it does for CLIPS, with a few very minor excep-      */
/* tions.  First, the salience of the rules in PMCLIPS is deter-  */
/* mined by the physical ordering in the rule set instead of the  */
/* value of the salience variable for the rule.  Secondly, the    */
/* first condition of a rule cannot be a "not" condition.  These  */
/* differences are obviously minor.                               */
/*                                                               */
/* Another external difference the user will notice is that when */
/* the reset command is given, PMCLIPS will prompt for how many   */
/* processors to use.  CLIPS obviously does not ask this because  */
/* it was designed to run on one processor.  PMCLIPS also prompts */
/* the user for the number of previous rule firings to save.      */
/* This allows run-time characteristics to be modified to match   */
/* the particular rule set being used.  Many rules sets do not    */
/* need any previous rule firings to be saved to match against,   */
/* but some rule sets do.  The saved rule firings are matched     */
/* against to see if they have recently fired.  If they have,     */
/* then a different instantiation is found, if one exists.        */
/*                                                               */
/* PMCLIPS author: Grady Cook                                     */
/*****************************************************************/

#include <stdio.h>
#include "clips.h"

#define TRUE 1
#define FALSE 0
#define OFF 0
#define ON 1
#define LHS 0
#define RHS 1

#define BINDS_MATCH 1
```