19940208 70

N94- 25352

PARALLEL PROCESSING METHODS FOR SPACE BASED POWER SYSTEMS

Final Report

NASA/ASEE Summer Faculty Fellowship Program--1993

Johnson Space Center

Prepared By:	F.C. Berry
Academic Rank:	Associate Professor
University & Department:	Department of Electrical Engineering Louisiana Tech University Ruston, Louisiana 71272
NASA/JSC	
Directorate:	Engineering
Division:	Propulsion And Power
Branch:	Power
JSC Colleagues:	Bob Hendrix
	Tom Jeffcoat
Data Submitted:	7-30-93
Contract Number:	NGT-44-001-800

ABSTRACT

This report presents a method for doing load-flow analysis of a power system by using a decomposition approach. The power system for the Space Shuttle is used as a basis to build a model for the load-flow analysis. To test the decomposition method for doing load-flow analysis, simulations were performed on power systems of 16, 25, 34, 43, 52, 61, 70, and 79 nodes. Each of the power systems was divided into subsystems and simulated under steady-state conditions. The results from these tests have been found to be as accurate as tests performed using a standard serial simulator. The division of the power systems into different subsystems was done by assigning a processor to each area. There were 13 transputers available, therefore, up to 13 different subsystems could be simulated at the same time.

This report has preliminary results for a load-flow analysis using a decomposition principal. The report shows that the decomposition algorithm for load-flow analysis is well suited for parallel processing and provides increases in the speed of execution.

INTRODUCTION

A research project, Advanced Electrical Power Management Techniques for Space Systems (ADEPTS), was started at the Johnson Space Center in 1986. The basic goal of ADEPTS was to automate the operations of a space based power system (SBPS) by using the technology of parallel and distributed processing.

From the ADEPTS project, three basic functions were identified which would form the basis of a management system for a SBPS. First was the monitoring of the power system that could be accomplished by the use of state estimation. Second was the scheduling of the generation to meet the required load of the SBPS that could be accomplished by unit dispatch. Third was the solution of the SBPS that could be accomplished by the use of load-flow analysis.

Methods like state estimation, unit dispatch, load-flow analysis, etc., are well-established tools that have been used heavily by the electric power industry since the introduction of the digital computer. However, the use of parallel processing is still relatively new when applied to the area of power systems [1]. In the last few years, a significant amount of research has been done in the area of parallel processing of power system problems. Most of this work has been in the development of algorithms. Actual testing on multiprocessor architectures is still near the beginning stages. The largest uncertainty today is the evolution of the hardware. For example:

1. Pipelined computers in which temporal parallelism is used to calculate overlapped computations. Early Cray-1 and Cyber 205 are good examples of these systems.

2. Array processors in which spatial parallelism is utilized through synchronized arithmetic logical units (ALU). The Illiac-IV is a good example of this system.

3. Parallel processing systems in which asynchronous parallelism is employed at the software level with the Cray X-MP and Cray 2 as typical examples.

Shared-memory and distributed processors are two basic systems of parallel computers that can offer increases in computational power but their acceptability and long term viability is unpredictable [1].

Because parallel algorithm development was much further along, ADEPTS chose to make use of the existing algorithms and focused on tailoring these algorithms to a

specific hardware design. To accomplish this, a hardware platform and software environment had to be decided upon. The INMOS T800 transputer was used as the basic hardware platform. The T800 transputer is a 32 bit reduced instruction set computer (RISC) running at 25 MHz [2]. The T800 transputer can communicate with other transputers or a host computer by a set of four bi-directional channels. By using these channels, a variety of hardware architecture can be investigated. Also, additional transputer modules can be easily added in the future to further enhance the system without making obsolete the existing transputers. Because of the difficulties encountered in developing parallel software, the Express® "operating system" was chosen. The Express package is best classified as a transparent parallel operating system with a set of tools and utilities for developing parallel programs [3, 4, 11].

A preliminary parallel architecture for a management system of a SBPS (Figure 1) was developed and tested using the power system of the Space Shuttle (Orbiter) as a model. The electric power distribution system that is present on the Orbiter is made up of three strings each having a fuel cell that provides DC voltage to those systems that require DC, and the inverters that convert the DC to AC for those elements that require AC. Figure 2 is a block diagram representation of the power system of the Orbiter.

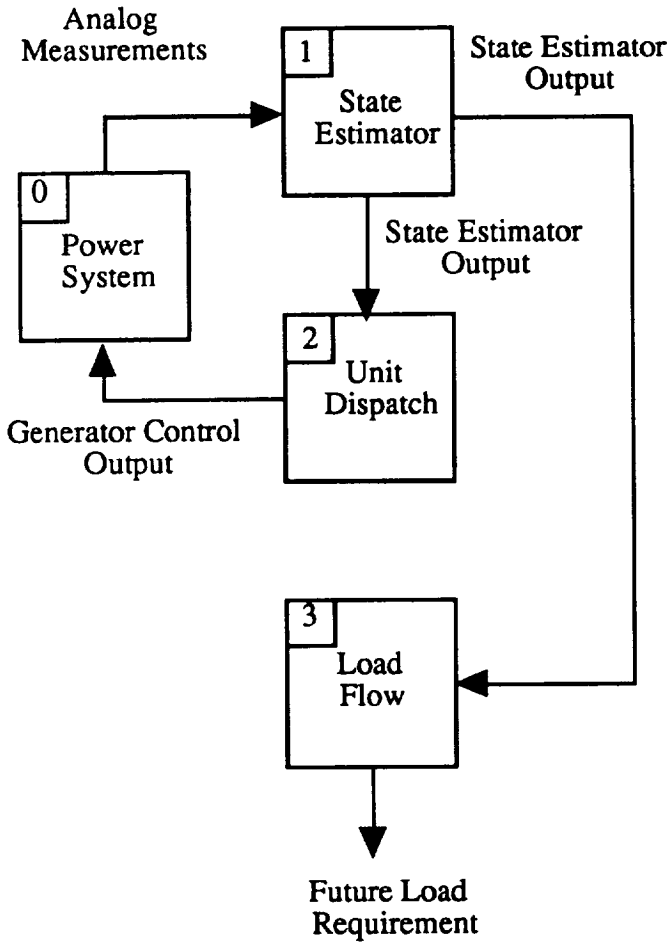


Figure 1: Block Diagram Of A Power Management System.

Preliminary tests of the architecture of Figure 1, using the Orbiter power system, showed that the load-flow program took from 20% to 50% of total processing time depending on the size of the electrical circuit that the load-flow program was solving. To improve the overall performance of the architecture of Figure 1, a different architecture for the load-flow was investigated.

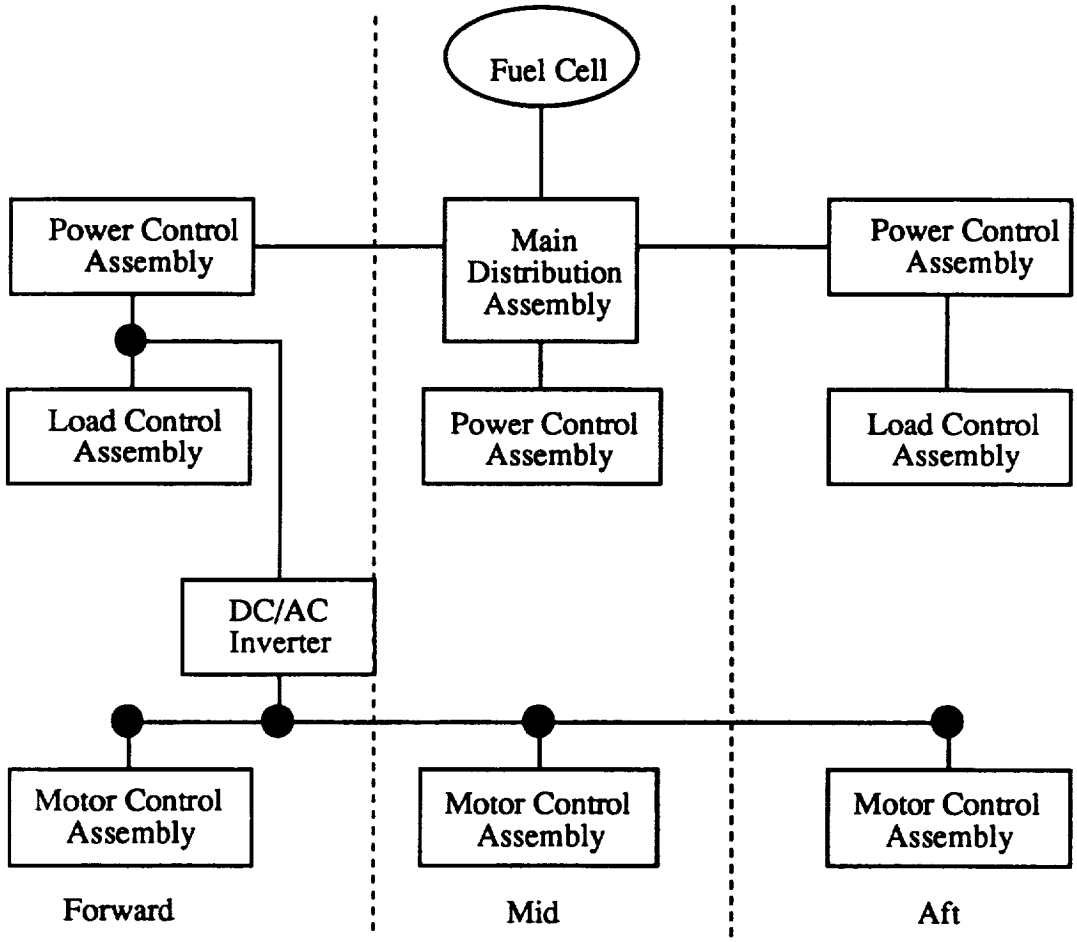


Figure 2: Power Distribution System for the Space Shuttle (1 of 3 Strings).

RELATED RESEARCH

A general definition of load-flow is given: Load-flow is the steady-state solution of equations that describe a power system. These equations can be nonlinear and algebraic. They can be nonlinear because they express powers as a function of voltages. They are algebraic because they describe the steady-state instead of the transient behavior of the power system. [5]

Rafian, Sterling, and Irving [6], presented a method for load-flow analysis for power systems by tearing the network into a number of independent subsystems. These subsystems could then be solved in parallel resulting in a saving of time for on-line control. This method of decomposed load-flow analysis was based on the fast decoupled load-flow algorithm. The decomposed load-flow technique that was presented would

normally give the greatest reduction in time, if the system could be divided such that the subsystems were interconnected only by a few tie lines (separating nodes).

Rafian, Sterling, and Irving [7], present a mathematical technique for the decomposition of a set of nonlinear algebraic and differential equations which result from a corresponding dynamic model of an electric power system. This method of load-flow analysis for power systems again tears the network into a number of independent subsystems. These subsystems could then be executed in parallel. This method of decomposed load-flow analysis was based on the Newton-Raphson load-flow algorithm. It was also shown that this decomposed load-flow algorithm was suitable for implementation on parallel processors with a 32-bit word capability. This decomposition technique resulted in significant savings in the simulation elapsed time.

Wang, Xiang, Wang, and Huang [8], presented two algorithms for a parallel solution of the reduced gradient optimal power flow. The parallel solution method for optimal power flow was accomplished by tearing the network into a number of independent subsystems. These subsystems were then solved in parallel by using what the authors called a two-level computer network.

Berry and Cox [9], presented a coordination decomposition method for load-flow analysis using the Gauss-Siedel algorithm. This coordination decomposition method was accomplished by the tearing of the electrical network into a number of independent subsystems which could then be solved in parallel. However, this load-flow algorithm was only simulated on a serial computer and no true parallel processing took place.

Taoka, Iyoda, Noguchi, Sato, and Nakazawa [10], presented a Gauss-Siedel method for doing load-flow analysis on a hyper cube computer. The algorithm that was presented consisted of solving the power flow equations in an iterative manner in order to minimize the communication between nodes.

LOAD-FLOW

Load-flow is the name given to a network solution that shows currents, voltages, and power flow at every bus in the system. In the load-flow problem, a relationship between voltage and current at each bus exists and the load-flow calculation must solve for all voltages and currents such that these relationships are satisfied. As such, the load-flow gives the electrical response of the transmission system to a particular set of loads and generator unit outputs. Therefore, load-flow is the solution of an electrical network that gives the values of currents, voltages, and power flow at every bus (node) in the electrical power system.

To start the load-flow solution process, a set of linear equations is formulated for an electrical network. This linear set of equations is generally of the node form as represented by equations (1).

$$\begin{bmatrix} I_1 \\ I_2 \\ . \\ . \\ I_i \end{bmatrix} = \begin{bmatrix} Y_{11} & Y_{12} & . & . & Y_{1j} \\ Y_{21} & Y_{22} & . & . & . \\ . & . & . & . & . \\ . & . & . & . & . \\ Y_{i1} & . & . & . & Y_{ii} \end{bmatrix} \begin{bmatrix} E_1 \\ E_2 \\ . \\ . \\ E_i \end{bmatrix} \qquad (1)$$

Although currents entering the nodes from generators and load branches are not known, they can be written in terms of P and E.

$$I_i = \frac{P_i}{E_i} \tag{2}$$

Since currents entering the nodes are considered positive, then power flow into the node is also considered positive, and power dissipated in the resistors would be entered as a negative number.

Substituting equation 2 into equation 1 gives the following result.

$$\begin{bmatrix} \frac{P_1}{E_1} \\ \frac{P_2}{E_2} \\ \cdot \\ \cdot \\ \frac{P_i}{E_i} \end{bmatrix} = \begin{bmatrix} I_1 \\ I_2 \\ \cdot \\ \cdot \\ I_i \end{bmatrix} = \begin{bmatrix} Y_{11} & Y_{12} & \cdot & \cdot & Y_{1j} \\ Y_{21} & Y_{22} & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ Y_{i1} & \cdot & \cdot & \cdot & Y_{ii} \end{bmatrix} \begin{bmatrix} E_1 \\ E_2 \\ \cdot \\ \cdot \\ E_i \end{bmatrix} \tag{3}$$

$$\begin{aligned} \frac{P_1}{E_1} &= Y_{11}E_1 + Y_{12}E_2 + \ldots + Y_{1j}E_i \\ \frac{P_2}{E_2} &= Y_{21}E_1 + Y_{22}E_2 + \ldots + Y_{2j}E_i \\ \cdot \quad & \quad \cdot \quad \cdot \quad \cdot \\ \cdot \quad & \quad \cdot \quad \cdot \quad \cdot \\ \frac{P_i}{E_i} &= Y_{i1}E_1 + Y_{12}E_2 + \ldots + Y_{1i}E_i \end{aligned} \tag{4}$$

The set of equations (4) may be solved for E_1 through E_i by the use of an iteration technique.

$$\begin{aligned} E_2 &= \frac{1}{Y_{22}}\left\{\frac{P_2}{E_2} - [Y_{21}E_1 + \ldots + Y_{2j}E_i]\right\} \\ \cdot \quad & \quad \cdot \quad \cdot \quad \cdot \\ \cdot \quad & \quad \cdot \quad \cdot \quad \cdot \\ E_i &= \frac{1}{Y_{ii}}\left\{\frac{P_i}{E_i} - [Y_{i1}E_1 + \ldots + Y_{(i-1)(j-1)}E_{(i-1)}]\right\} \end{aligned} \tag{5}$$

Note that E_i has been written in terms of itself and the other voltages. Also, one equation has been deleted, this equation or bus is designated as the swing or slack bus. The swing or slack bus is generally a generator that is the first to react to changes in the system. Generally the voltage at the slack bus is specified.

DECOMPOSITION METHOD

The decomposition method for load-flow analysis was started by dividing the electrical power system into a number of smaller systems or subsystems. These subsystems can then be solved independently in parallel [13]. To demonstrate the decomposition method for load-flow analysis, an example will be presented.

The electrical power system of Figure 3 was decomposed by using the admittance matrix (Table 1). The admittance matrix was transformed so that each of the subsystems would appear sequentially; that is, the nodes were renumbered for the electrical circuit. This renumbering caused the network equations for each subsystem to follow the

dynamic equations for that same subsystem. Hence, for the power system of Figure 3 there were 4 different subsystems. In the admittance matrix, the diagonal elements Y_{ii} are the summation of all admittances that surround the ith node, where $Y_{il} = Y_{1i}$ and is the negative of the branch. The admittance for Figure 3 is given in Table 1.

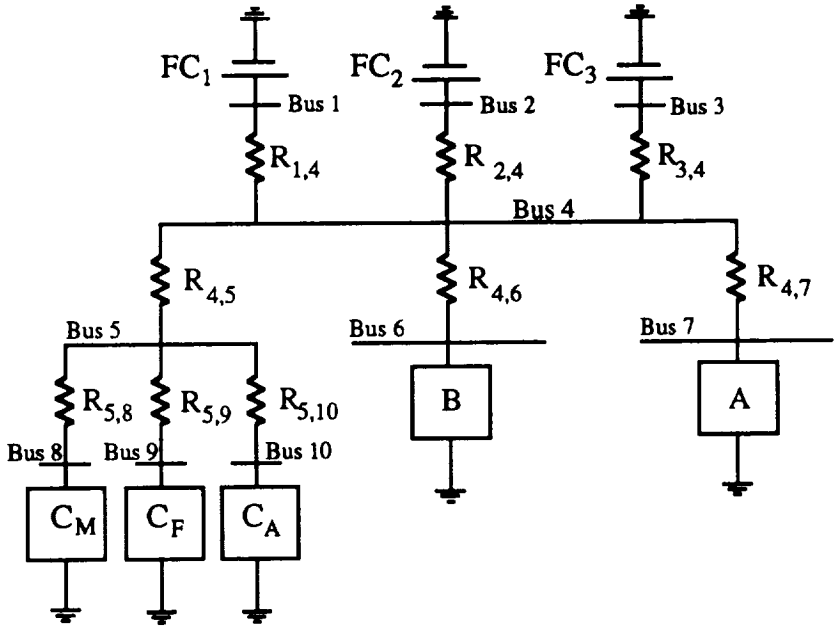


Figure 3: Space Shuttle Power Distribution System.

Table 1: The Admittance Matrix.

	1	2	3	4	5	6	7	8	9	10
1	$Y_{1,1}$			$-Y_{1,4}$						
2		$Y_{2,2}$		$-Y_{2,4}$						
3			$Y_{3,3}$	$-Y_{3,4}$						
4	$-Y_{4,1}$	$-Y_{4,2}$	$-Y_{4,3}$	$Y_{4,4}$	$-Y_{4,5}$	$-Y_{4,6}$	$-Y_{4,7}$			
5				$-Y_{5,4}$	$Y_{5,5}$			$-Y_{5,8}$	$-Y_{5,9}$	$-Y_{5,10}$
6				$-Y_{6,4}$		$Y_{6,6}$				
7				$-Y_{7,4}$			$Y_{7,7}$			
8					$-Y_{8,5}$			$Y_{8,8}$		
9					$-Y_{9,5}$				$Y_{9,9}$	
10					$-Y_{10,5}$					$Y_{10,10}$

After the renumbering of Table 1 is completed the admittance matrix of Table 2 is created. The admittance matrix from Table 2 can be expressed as the sum of two matrices.

$$[Y] = [Y_l] + [Y_c] \tag{6}$$

Where $[Y_l]$ are a number of block diagonal submatrices that represents a matrix for a specific subsystem. From the $[Y_l]$ matrix, it can be seen that four different sets of equations can be produced, one set for each subsystem as shown in Figure 4 and Table 3.

Table 2: The Admittance Matrix Renumbered.

	1	2	3	4	5	6	7	8	9	10
1	$Y_{1,1}$			$-Y_{1,4}$						
2		$Y_{2,2}$		$-Y_{2,4}$						
3			$Y_{3,3}$	$-Y_{3,4}$						
4	$-Y_{4,1}$	$-Y_{4,2}$	$-Y_{4,3}$	$Y_{4,4}$	$-Y_{4,5}$				$-Y_{4,9}$	$-Y_{4,10}$
5				$-Y_{5,4}$	$Y_{5,5}$	$-Y_{5,6}$	$-Y_{5,7}$	$-Y_{5,8}$		
6					$-Y_{6,5}$	$Y_{6,6}$				
7					$-Y_{7,5}$		$Y_{7,7}$			
8					$-Y_{8,5}$			$Y_{8,8}$		
9				$-Y_{9,4}$					$Y_{9,9}$	
10				$-Y_{10,4}$						$Y_{10,10}$

Table 3: The Admittance Matrix $[Y_1]$.

	1	2	3	4	5	6	7	8	9	10
1	$Y_{1,1}$			$-Y_{1,4}$						
2		$Y_{2,2}$		$-Y_{2,4}$						
3			$Y_{3,3}$	$-Y_{3,4}$						
4	$-Y_{4,1}$	$-Y_{4,2}$	$-Y_{4,3}$	$Y_{4,4}$						
5					$Y_{5,5}$	$-Y_{5,6}$	$-Y_{5,7}$	$-Y_{5,8}$		
6					$-Y_{6,5}$	$Y_{6,6}$				
7					$-Y_{7,5}$		$Y_{7,7}$			
8					$-Y_{8,5}$			$Y_{8,8}$		
9									$Y_{9,9}$	
10										$Y_{10,10}$

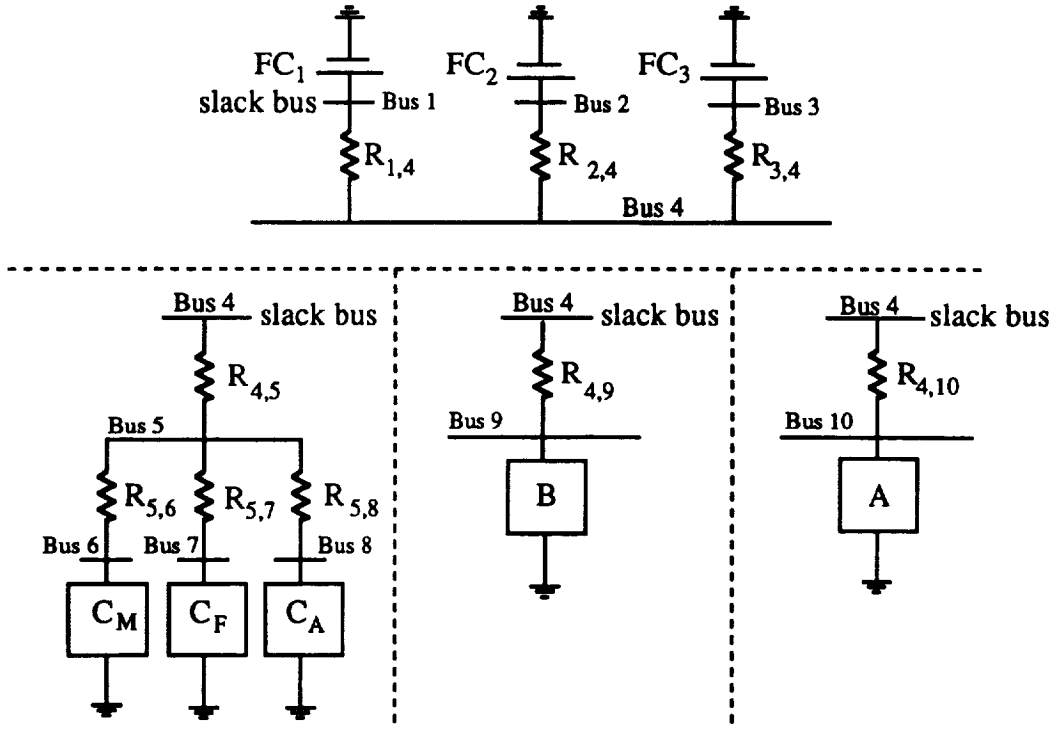
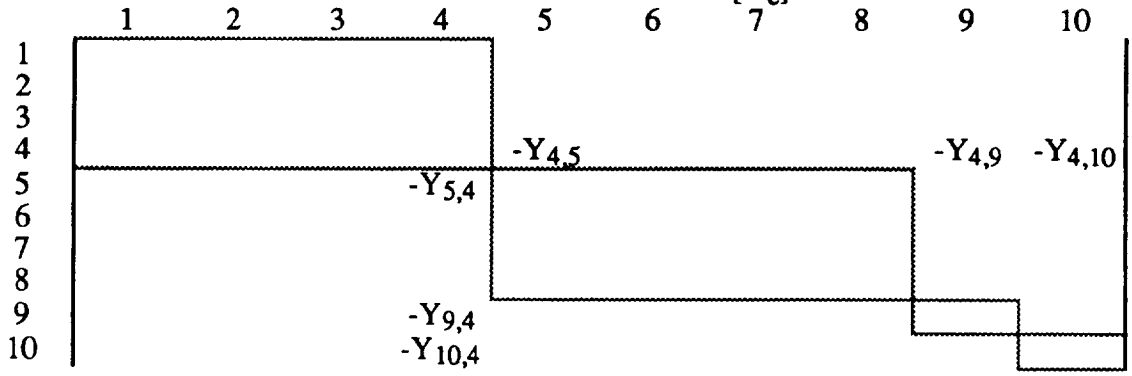


Figure 4: Decomposed Space Shuttle Power Distribution System.

Each of the individual subsystems shown in Figure 4 and Table 3 can be solved independently. These subsystems require their own slack bus and one of these slack buses will serve as the slack bus for the entire system.

The remaining elements, $[Y_c]$ of the admittance matrix are the nonzero off diagonal elements that are the separating nodes of one subsystem with respect to the node voltages in another subsystem. Table 4 shows the $[Y_c]$ matrix.

Table 4: The Admittance Matrix $[Y_c]$.



These nonzero off diagonal elements act as sending or receiving nodes for power flow from one subsystem to another. Therefore, these nonzero off diagonal elements can be referred to as voltage and power (EP) separating nodes. Finally, these nonzero off diagonal elements are directly related to the number of common nodes in different subsystems.

From the $[Y_c]$ matrix, it can be seen that at node 4 the system was divided. The $[Y_c]$ matrix is the coordination matrix and generally this matrix will contain the separating nodes. The function of the coordination matrix is to determine the amount of ΔP and ΔE that the separating nodes contribute to each subsystem.

PERFORMANCE ANALYSIS

To test the decomposition method for doing load-flow analysis, simulations were performed on power systems of 16, 25, 34, 43, 52, 61, 70, and 79 nodes. Each of the power systems was divided into subsystems and simulated under steady-state conditions. The results from these tests have been found to be as accurate as tests performed using a standard serial simulator. The division of the power systems into different subsystems was done by assigning a processor to each area. Table 5 provides the results of timing tests for each of the power systems. The times reported in Table 5 are the times associated with the solution of the load-flow problem (Node Time) and part of the host I/O (Host I/O). The Node Time includes internode communication time, calculation time, and part of the host I/O time. The Host I/O time reported in Table 5 is that part of the total host I/O time required to print the result of the load-flow calculation to the screen and the time to create two files, which contains the data of the timing tests that are reported in the following tables.

Table 5: Timing Data

		16 buses	25 buses	34 buses	43 buses
1	Node Time ms	56.77	94.59	146.88	211.9
Transputer	Host I/O ms	115.25	116.35	115.46	115.6
2	Node Time ms	33.41	44.29	51.97	71.6
Transputer	Host I/O ms	112.45	123.20	122.88	113.7
4	Node Time ms	33.02	44.67	48.19	54.9
Transputer	Host I/O ms	102.46	122.82	122.94	123.2
10	Node Time ms	*	*	*	44.4
Transputer	Host I/O ms	*	*	*	128.2
13	Node Time ms	*	*	*	25.9
Transputer	Host I/O ms	*	*	*	110.5

The results of the timing tests that are reported [illegible] running each program 10 times and examining the timi[illegible] recorded when the program finishes [3, 4, 11]. In order to [illegible] like speedup and efficiency, the times reported for the sin[illegible] "best case" and the times reported for the multiprocessor [illegible] By reporting the "best case" times for the single processor [illegible] times for the multiprocessor programs would provide the [illegible] and efficiency for the multiprocessor programs.

Again from Table 5 the Node Times did not change [illegible] these numbers are the actual times. The Host I/O numbers ch[illegible] next. Being consistent with the idea of the "worst case" [illegible] the best "best case" time for the single processor program[illegible] time for the multiprocessor programs. By using the best "[illegible] processor programs and the worst "worst case" time for [illegible] would provide the worst "worst case" for the speed[illegible] multiprocessor programs. This adjustment to Table 5 is give[illegible]

Table 6: Adjusted Time

		16 buses	25 buses	34 buses	43 buses
1	Node Time ms	56.77	94.59	146.88	211.90
Transputer	Host I/O ms	99.84	99.84	99.84	99.84
2	Node Time ms	33.41	44.29	51.97	71.68
Transputer	Host I/O ms	128.26	128.26	128.26	128.26
4	Node Time ms	33.02	44.67	48.19	54.91
Transputer	Host I/O ms	128.26	128.26	128.26	128.26
10	Node Time ms	*	*	*	44.42
Transputer	Host I/O ms	*	*	*	128.26
13	Node Time ms	*	*	*	25.92
Transputer	Host I/O ms	*	*	*	128.26

From Table 6, Table 7 was created. Table 7 is the total "be[illegible] processor program and the total "worst case" time for each mu[illegible]

Table 7: "Worst Case" Total Time

		16 buses	25 buses	34 buses	43 buses	52 buses	61 buses	70 buses	79 buses
1	Node Time ms	56.77	94.59	146.88	211.90	233.02	252.99	275.14	297.28
Transputer	Host I/O ms	99.84	99.84	99.84	99.84	99.84	99.84	99.84	99.84
	Total Time ms	156.61	194.43	246.72	311.74	332.86	352.83	374.98	397.12
2	Node Time ms	33.41	44.29	51.97	71.68	82.68	91.14	98.88	104.38
Transputer	Host I/O ms	128.26	128.26	128.26	128.26	128.26	128.26	128.26	128.26
	Total Time ms	161.67	172.55	180.23	199.94	210.94	219.40	227.14	232.64
4	Node Time ms	33.02	44.67	48.19	54.91	57.92	61.06	63.81	66.24
Transputer	Host I/O ms	128.26	128.26	128.26	128.26	128.26	128.26	128.26	128.26
	Total Time ms	161.28	172.93	176.45	183.17	186.18	189.32	192.07	194.50
10	Node Time ms	*	*	*	44.42	44.99	45.63	49.02	49.54
Transputer	Host I/O ms	*	*	*	128.26	128.26	128.26	128.26	128.26
	Total Time ms	*	*	*	172.68	173.25	173.89	177.28	177.80
13	Node Time ms	*	*	*	25.92	26.56	26.88	26.94	26.82
Transputer	Host I/O ms	*	*	*	128.26	128.26	128.26	128.26	128.26
	Total Time ms	*	*	*	154.18	154.82	155.14	155.20	155.08

Speedup is a measure of the application software utilization of a multiple processor system. However, the original purpose of the speedup was to compare the time of the fastest serial program, T^1, with the time of the parallel equivalent of the same program, T(p). [15] This definition of speedup has a different meaning, e.g.,

$$S(p) = \frac{T^1}{T(p)} \tag{7}$$

The definition of the speedup from equation 7 is rarely used because of the difficulty in measuring T^1. Instead, T(1) is used as an approximation of T^1. Therefore, the speedup is estimated from the measurement for T(1) and T(N) [15] and is defined as follows:

$$S(p) = \frac{T(1)}{T(p)} \tag{8}$$

The speedup of the different multiprocessor programs is given in Table 8. Again these are the "worst case" values for the speedup.

Table 8: Speedup Results

		16 buses	25 buses	34 buses	43 buses	52 buses	61 buses	70 buses	79 buses
1	Total Time ms	156.61	194.43	246.72	311.74	332.86	352.83	374.98	397.12
Transputer	Speedup	NA	NA	NA	NA	NA	NA	NA	NA
2	Total Time ms	161.67	172.55	180.23	199.94	210.94	219.40	227.14	232.64
Transputer	Speedup	0.96	1.12	1.36	1.55	1.57	1.60	1.65	1.70
4	Total Time ms	161.28	172.93	176.45	183.17	186.18	189.32	192.07	194.50
Transputer	Speedup	0.97	1.12	1.39	1.70	1.78	1.86	1.95	2.04
10	Total Time ms	*	*	*	172.68	173.25	173.89	177.28	177.80
Transputer	Speedup	*	*	*	1.80	1.92	2.02	2.11	2.23
13	Total Time ms	*	*	*	154.18	154.82	155.14	155.20	155.08
Transputer	Speedup	*	*	*	2.02	2.14	2.27	2.41	2.56

Processor efficiency measures the contribution of each processor to the parallel solution when p processors are employed. That is, E(p) equals the average efficiency per processor when the problem is run with p parallel processors [15].

$$E(p) = \frac{S(p)}{p} \tag{9}$$

Therefore an efficiency rating of 100% means the program runs in linear speedup time, while an efficiency of 0% means the processor is of no use in solving the problem in parallel [15].

The efficiency of the different multiprocessor programs is given in Table 9. Again these are the "worst case" values for the efficiency.

Table 9: Final Speedup And Efficiency Results

		16 buses	25 buses	34 buses	43 buses	52 buses	61 buses	70 buses	79 buses
1	Speedup	NA	NA	NA	NA	NA	NA	NA	NA
Transputer	Efficiency	NA	NA	NA	NA	NA	NA	NA	NA
2	Speedup	0.96	1.12	1.36	1.55	1.57	1.60	1.65	1.70
Transputer	Efficiency	48.0%	56.0%	68.0%	77.5%	78.5%	80.0%	82.5%	85.0%
4	Speedup	0.97	1.12	1.39	1.70	1.78	1.86	1.95	2.04
Transputer	Efficiency	24.2%	28.0%	34.7%	42.5%	44.5%	46.5%	48.7%	51.0%
10	Speedup	*	*	*	1.80	1.92	2.02	2.11	2.23
Transputer	Efficiency	*	*	*	18.0%	19.2%	20.2%	21.1%	22.3%
13	Speedup	*	*	*	2.02	2.14	2.27	2.41	2.56
Transputer	Efficiency	*	*	*	15.5%	16.4%	17.4%	18.5%	19.6%

FAULT-TOLERANCE

Fault tolerance is a class of methods to achieve dependable and reliable computing systems. Dependability is that property of a computer system that allows reliance to be justifiably placed on the service a system delivers [12, 14]. Fault tolerance is the ability to deliver a service in spite of faults. Fault tolerance is generally obtained from redundancy. This redundancy can be accomplished by masking (static) redundancy or dynamic redundancy [12, 14].

Masking (static) redundancy uses extra components to obscure, or mask, the effect of a faulty component [12, 14]. This technique is regarded as static because once the redundant copies are connected their interconnections remain fixed. The error resulting from faulty components are masked by the presence of other copies of those components. An example of masking redundancy is the computer system for the Space Shuttle. There are four different computers on the Space Shuttle. Three of these computers execute the same program using the same data while the fourth computer runs a different program which performs the same function as the other three. The results of the three computers running the same program are compared or voted on, if any one of the result from one of the computers is inconsistent with that of the others, it is masked out.

The reason that the fourth computer has a different set of software that performs the same function as the other three is to grade against an error in programming. This is a frorm of dynamic redundancy. Dynamic redundancy refers to systems which can be reconfigured in response to a fault. If the fourth computer results were different from the results of the other three computers on the Space Shuttle, the fourth computer would take

over and mask out the others, thereby reconfiguring the computer system from a multiprocessor to a single processor computer system.

Another fault tolerant technique is fault detection, but fault detection provides no tolerance to faults, rather it gives warning when they occur [12, 14].

The reason for this introduction to fault tolerant system was to show how they could be applied to this research. The goal of this research is not to design a fault tolerant computer system but to apply parallel and distributed processing to construct an electric power management system (EPMS) for space based power systems. In the development of this EPMS, two different architectures for doing the decomposition method for load-flow were tested. These results have been presented in Tables 5 through 9. They were the 10 and 13 transputer systems and are presented in Figures 5 and 6.

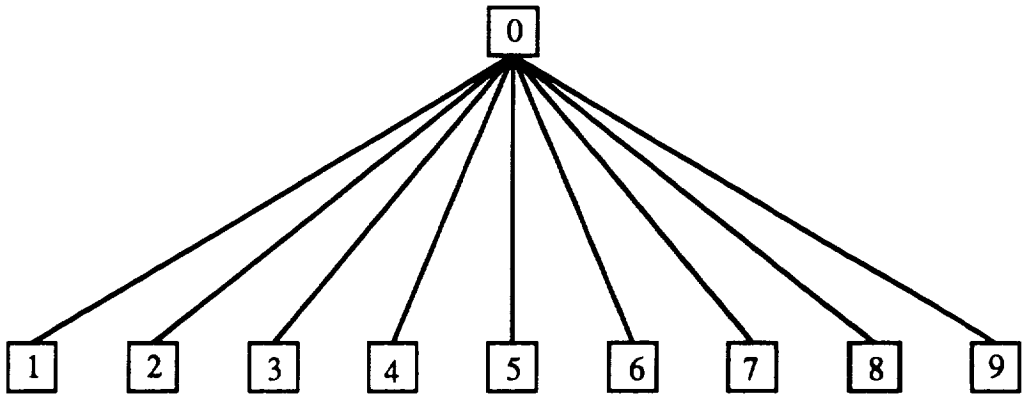


Figure 5: 10 Transputer System

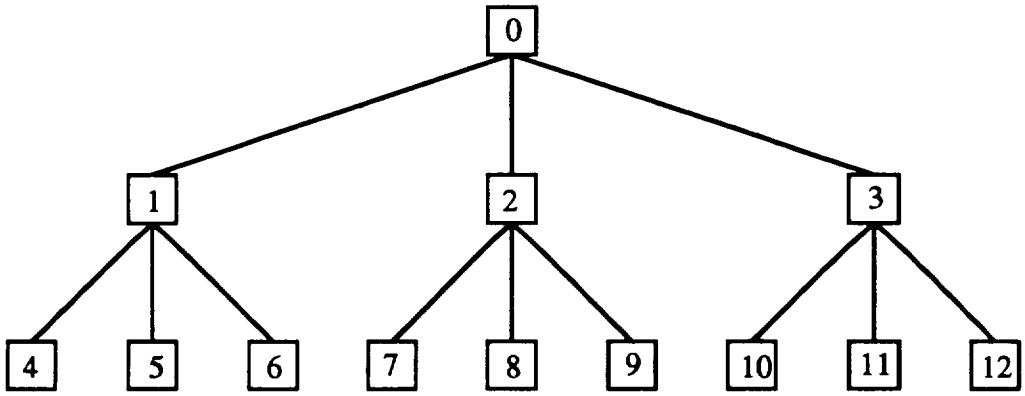


Figure 6: 13 Transputer System

From Figures 5 and 6 it can be seen that processor 0 is the critical processor. If processor 0 were to fail, both systems in Figure 5 and 6 would fail. However if processor 1, 2, or 3 were to fail in Figure 6, the processors which feed into any of these processors would be removed from service. An alternative system which is given in Figure 7 would use 12 processors. The system in Figure 7 would operate the same as the system in Figure 5 with the exception that processors 10 and 11 would perform the same function as processor 0. This would provide a static redundancy for processor 0 and would eliminate the risk of a fault on the middle layer of processors in Figure 6. The system in

Figure 7 was not tested so no timing data are available at this time, but the point is made that fault tolerance can be addressed in this design.

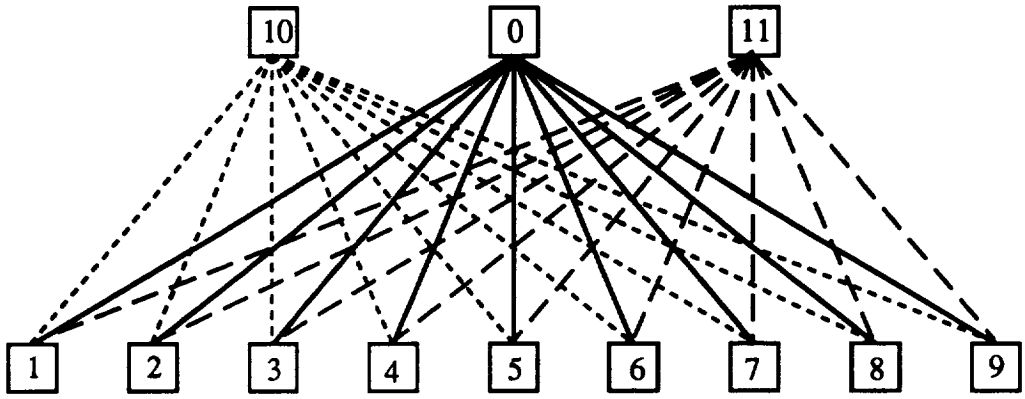


Figure 7: Proposed Fault Tolerant System

CONCLUSIONS

This paper has presented preliminary results for a load-flow analysis using a decomposition principal. It has been shown that the decomposition algorithm for load-flow analysis is well suited for parallel processing and provides increases in the speed of execution. Also two different hardware structures for doing the decomposition method were presented (Figures 5 and 6) and they were analyzed for speedup and efficiency. A fault tolerant design was also proposed (Figure 7) which made use of static redundancy.

Further work will be conducted on the decomposition principal by the addition of more transputers to the present system. This will be done to see whether or not further decomposition by the addition of more processors will be of any benefit. Also with the addition of more transputers, a hyper cube architecture could also be investigated; this approach was presented by Taoka, Iyoda, Noguchi, Sato, and Nakazawa [10].

REFERENCES

[1] IEEE Committee Report, "Parallel Processing in Power Systems Computation," *IEEE Transactions On Power Systems*, Vol. 7, No. 2, May 1992, pp. 629-637.

[2] A.M. Tyrrell and J.D. Nicoud, "Scheduling And Parallel Operations On The Transputer," *Microprocessing and Microprogramming*, Vol 26, 1989, pp. 175-185.

[3] Express 3.0, Introductory, Guide Macintosh, ParaSoft Corporation, 2500, E. Foothill Blvd., Pasadena, CA 91107, 1988, 1989, 1990.

[4] Express 3.0, User's Guide, Version 3.0, ParaSoft Corporation, 2500, E. Foothill Blvd., Pasadena, CA 91107, 1988, 1989, 1990.

[5] Y. Wallach, Calculations And Programs For Power System Networks, Prentice-Hall, Inc., Englewood Cliffs, New Jersey 07632, 1986.

[6] M. Rafian, M.J.H. Sterling, and M.R. Irving, "Decomposed Load-Flow Algorithm Suitable For Parallel Processing Implementation," *IEE Proceedings*, Vol. 132, Pt. C, No. 6, November 1985.

[7] M. Rafian, M.J.H. Sterling, and M.R. Irving, "Parallel Processing Algorithm For Power System Simulation," *IEE Proceedings*, Vol. 135, Pt. C, No. 6, November 1988.

[8] L. Wang, N. Xiang, S. Wang, and M. Huang, "Parallel Reduced Gradient Optimal Power Flow Solution," *Electric Power Systems Research*, Vol. 17, 1989, pp. 229-237.

[9] F.C. Berry and M.D. Cox, "A Technique For Load Flow Analysis On A Power System," *IECEC Proceedings*, Washington, D.C., 1989.

[10] H. Taoka, I. Iyoda, I., H. Noguchi, N. Sato, and T. Nakazawa "Real-Time Digital Simulation For Power System Analysis On A Hyper Cube Computer," *IEEE Transactions On Power Systems*, Vol. 7, No. 1, Feb. 1992, pp. 1-7.

[11] F.C. Berry, "Express, A Tool for Teaching Parallel Processing," *Computers In Education Journal*, Computers In Education Division Of ASEE, Accepted, 1992.

[12] D.P. Siewiorek, "Architecture of Fault-Tolerant Computers: An Historical Perspective," *Proceedings Of The IEEE*, Vol. 79, No. 12, Dec. 1991, pp. 1710-1734.

[13] H.M. Chen and F.C. Berry, "Parallel Load-Flow Algorithm Using A Decomposition Method For Space Based Power Systems," *IEEE Transactions on Aerospace and Electronic Systems*, Accepted, 1993

[14] L.J.M. Nieuwenhuis and G.D. Blom, "Fault Tolerant Computing With Transputers And Occam," *Real-Time Systems With Transputers*, H. Zedan, Ed., 1990, IOS Press, pp. 108-118.

[15] T.G. Lewis and H. El-Rewini, Introduction To Parallel Computing, Prentice Hall, Englewood Cliffs, New Jersey 07632, 1992.

[16] F.C. Berry, R.F. Gasser Jr., and H.M. Chen, "Course Grain Parallel Processing for the Management of a Space Based Power System," *IEEE Transactions on Aerospace and Electronic Systems*, Accepted, 1993.