Listed are abstracts from recent papers by IBM authors. Inquiries should be directed to the publication cited.

Abstracts

ALERT: A Program to Compile Logic Designs of New Computers, T. D. Friedman, *Digest of First Annual IEEE Computer Conference* **16C51,** 128–130 (September 1967). The ALERT program reads preliminary descriptions of new computers in a high level language and "compiles" them into logic designs. The input consists of a computer's architecture in a modified Iverson notation. From this, logic designs are generated in the form of Boolean equations. The program is described by summarizing in step by step fashion how a design is transformed automatically from Iverson notation into logic equations.

Analyzing Errors with the Boolean Difference, L. W. Bearnson, M. Y. Hsiao,* and F. F. Sellers, *Digest of the First Annual IEEE Computer Conference* **16C51,** 6–9 (September 1967). The Boolean difference is defined. It is shown through example how to use the Boolean difference to analyze the effect of errors on the outputs of logic circuits. Examples are given of error detection problems, analysis of redundant logic, and the generation of diagnostic sequences.

* On educational leave, University of Florida.

Application of Finite Geometry in File Organization for Records with Multiple Valued Attributes, S. P. Ghosh and C. T. Abraham, *IBM Journal of Research and Development* **12,** No. 2, 180–187 (March 1968). The schemes for organizing binary-valued records using finite geometries have been extended to the situation in which the attributes of the records can take multiple values. Some new schemes for organizing records have been proposed which are based on deleted finite geometries. These new schemes permit the organization of records into buckets in such a manner that, by solving certain algebraic linear equations over a finite field, it is possible to determine the bucket in which records, pertaining to two given values of two different attributes, are stored. Since the bucket identification required for the storage of record accession numbers is based on the combination of attribute values, the file does not require any reorganization as new records are added. This is a definite advantage of the proposed schemes over many key-address transformation procedures wherein the addition of new records may lead to either a drastic revision of the file organization or significant reduction of retrieval effectiveness. The search time for the new schemes are very small in comparison to other existing methods.

Architecture for an Extended Mission Aerospace Computer, F. H. Hardie and M. Ball, *Computer Design* **6,** No. 11, 34–36 (November 1967). A study of a computer subsystem for application in an extended Apollo mission and a follow-on effort in which an extended-mission computer was configured are reported. The proposed aerospace computer incorporates three primary operational modes within a single machine: (1) A reliable, long-term mission mode, featuring simplex operation and built-in spares. (2) A highly reliable mode for critical phases of a mission, during which continuous system operation is required, featuring triple modular redundancy and automatic switching. (3) A multiprocessing mode for handling and processing large quantities of data, featuring a partitioning of the computer into three simplex computers that can operate either independently or correlated.

Automated Fault Location on Processing Units, B. W. Clark, *Proceedings of the 1967 Automatic Support Systems Symposium for Advanced Maintainability,* St. Louis Section of IEEE, **2F1-2F10** (Nov. 7–9, 1967). This paper describes the automated production and use of sets of test patterns to check the logic of a processing unit. During the design of such a unit a Logic Master Tape is produced which contains a complete logical description of the unit, together with the physical location of each logic block. A set of computer programs has been written which produces, from this tape, a set of test patterns which, when applied to the unit, will detect and isolate a large percentage (>90%) of all logic failures. The patterns are applied to the unit by paths independent of the working logic of the units and the application of the tests is controlled by a small test equipment computer. This may either be built into the unit or may be external to it. While the application of each test is not dependent on the successful completion of other tests, the resolution to a replaceable unit does depend on test sequencing. In general, to obtain better resolution it is necessary to increase the number of tests in the series. Various experiments have been made to determine the trade-offs between these parameters. The indication of the suspect replaceable unit may be contained in a document, or it may be displayed directly in the maintenance console. Tests produced in this way have been used with considerable success on some models of the IBM SYSTEM/360, as well as on various military units. The programming system and the theory of application is independent of machine technology and architecture and the system can thus be used wherever it is necessary to repair a unit quickly without having to use highly trained personnel.

Automatische Spracherkennung (Automatic speech recognition), E. H. Rothauser, *Nachrichtentechnische Zeitschrift* **20,** No. 7, 381–384 (July 1967). Existing machines can recognize carefully spoken single words almost without error, but the recognition of continuous speech is not yet possible. The methods developed for limited vocabularies are not easily generalizable. Renewed attention must, therefore, be given to the basic questions: a) What parameters are necessary and sufficient for the description of speech, and b) how can continuous speech be recognized on the basis of such parameters? To solve the latter problem, a variety of recognition methods will have to be combined in a hierarchical structure.

Some Bounds on the Storage Requirements of Sequential Machines and Turing Machines, R. M. Karp, *Journal of the Association for Computing Machinery* **14,** No. 3, 478–489 (July 1967). Any sequential machine M represents a function f_M from input sequences to output symbols. A function f is representable if some finite-state sequential machine represents it. The function f_M is called an nth order approximation to a given function f if f_M is equal to f for all input sequences of length less than or equal to n. It is proved that for an arbitrary nonrepresentable function f, there are infinitely many n such that any sequential machine representing an nth order approximation to f has more than $n/2 + 1$ states. An analogous result is obtained for two-way sequential machines and, using these and related results, lower bounds are obtained on the amount of work tape required by on-line and off-line Turing machines that compute nonrepresentable functions.

Computer Methods of Network Analysis, F. H. Branin, Jr., *1967 IEEE International Convention Record, Part 5, Circuit Theory,* 45–63. This is a tutorial paper which highlights the influence of the computer not only on the modus operandi of circuit design but also on network theory itself. It reviews the topological properties of linear graphs and describes a matrix-topological formulation of the network problem. In addition to the classical mesh, node, and cutset methods, a "mixed" method of analysis is described which is applicable to D-C, A-C, and transient problems and underlies the so-called state-variable approach to network analysis. Numerical methods of solving the network problem are discussed first in connection with linear and nonlinear D-C networks. A new approach to A-C analysis, using the mixed method and based on a numerical solution of the matrix eigenvalue problem, is described. The application of this method to the transient analysis of linear networks is also explained. Finally, the problem of instability in numerical integration of differential equations is discussed and several means of solving the problem are outlined.

Characterization of Magnetic Film Memories—A Step Toward Automated Design and Optimization, H. Chang and C. P. Wang, *Digest of the First Annual IEEE Computer Conference* **16C51,** 74–76 (September 1967). Generalized characterization of magnetic film memory systems is an essential step toward computerized memory design. This paper formulates basic relationships among all the major design parameters by identifying their underlying physical mechanisms and system functions common to all the magnetic film memories. These formulations provide the basis for total memory design and optimization. Several examples are given to illustrate the choice of design parameters and their influence on memory performance.

Common Mode Capacitance Cross Coupling in Memory Arrays, H. O. Leilich and T. R. Scott, *IEEE Transactions on Magnetics* **MAG-3,** No. 3, 495–500 (September 1967). Capacitance between intersecting word and bit lines of a memory array causes a special type of line-to-line interaction. Signal waveforms may be seriously distorted if a large number of common polarity signals occur simultaneously and the word and bit line delay times are comparable to the width of the signal. A 2-dimensional linear array model with distributed parameters was used, and a mathematical process for the determination of the wave coefficients according to the partial differential equation of the array and its boundary conditions is described. Typical cases were computed, interpreted, and compared to simple approximations which are useful as rules of thumb for the memory designer.

Data Link Control Procedures: What They Are and What They Mean to the User, H. F. Ickes, *Proceedings of 22nd National Conference Association for Computing Machinery* **P67,** 521–525 (August 1967). Part I of this paper presents an analogy between the characteristics of most computer input/output (I/O) operations and the characteristics of the data link control procedures being developed by the United States of America Standards Institute Task Group X3.3.4 (Data Transmission Control Procedures). The computer I/O operation characteristics consist of 1) control of data transfer by the computer, 2) starting of I/O devices, 3) determining status of I/O devices before operation, 4) formatting of data (e.g., printline, card, etc.), 5) data transfer checking (character parity and check characters), 6) error recovery (usually retry operation), 7) awareness of end of I/O operation, 8) halt I/O operation, and 9) discarding of records. Part II tells what the standardization of data link control procedures means to the user and what benefits can be derived from this standards effort.

Design and Use of Fault Simulation for Saturn Computer Design, F. H. Hardie and R. J. Suhocki, *IEEE Transactions on Electronic Computers* **16,** No. 4, 412–429 (August 1967). A system of IBM 7090 data processing system computer programs was developed for the purpose of normal and/or fault simulation of the Saturn computer. This paper will describe the design of the simulator and cite several applications in the development of the Saturn computer. The architecture, plus several important characteristics of the simulator, are presented. These include the Design Automation input interface, the logic selection procedure, failure injection, the compilation procedure, logical simulation and functional simulation. The ability to simulate up to 4000 Saturn instructions in either normal and/or fault environments (up to 33 faults per IBM 7090 run) will be demonstrated. Simulation of single, multiple, solid or intermittent faults, plus an automated statistical analysis of intermittent fault simulation results, will be presented. The IBM 7090 execution time of a compiled logic simulator can be prohibitive. To minimize running time several programming techniques were utilized, including logic block ordering (to allow single pass simulation), parallel fault simulation, stimulus bypassing, and functional simulation. These techniques are described. Several special forms of simulator output were developed. The use of this output and the applications of the simulator are presented, including design verification, test program evaluation, generation of a test point catalog, disagreement detector network evaluation, disagreement detector placement, intermittent failure analysis.

The Effects of Divided Attention on Visual Monitoring of Multi-Channel Displays, J. D. Gould and A. Schaffer, *Human Factors* **9,** No. 3, 191–202 (June 1967). This study investigated effects of divided attention on monitoring multi-channel alphameric displays for signals defined on the basis of the simultaneous values of all channels, i.e., multi-channel signals as opposed to single-channel signals. Variables investigated included (a) three methods of dividing attention (a short writing task, a long writing task, and blanking out the display), (b) number of channels monitored (4, 8, 12, and 16), (c) rate of display change (6 or 12 times per minute), (d) number of different signals simultaneously watched for (8 or 24), and (e) number of levels within channels (2 or 8). The main results were: (a) divided attention did not lead to a decrease in monitoring, compared to a control study without divided attention; (b) the rate of display change had the greatest effect upon performance, followed by the number of channels monitored; (c) even at the faster rate of display change, untrained subjects detected 80% or more of the signals when they monitored up to 12 channels; and (e) different methods used to divide attention affect performance differentially.

Experiments in Software Modeling. J. L. Kessler and D. Fox,* *AFIPS Conference Proceedings* **31,** 1967 Fall Joint Computer Conference, 429–436. This paper presents a summary of several experiments that were conducted to explore the feasibility of using a performance model of proposed software to make design decisions about that software. Some current simulation systems were used or considered but were not well suited for the purpose. Several modeling facilities specifically aimed at the software modeler have been identified. Some of these were implemented as FORTRAN subroutines, and further experiments were conducted. These led to the identification of additional desirable facilities now under development.

* Fox Computer Services, Inc.

Faster Printing Through Customized Chain Design, E. B. Eichelberger, W. C. Rodgers and E. W. Stacy, *IBM Journal of Research and Development* **12,** No. 2, 130–139 (March 1968). Many high-speed printers now in the field and under development use a constantly moving chain or train containing the characters required in the printing process. Generally they skip to the next line whenever the last character on a given line is printed. Since individual character usage varies widely, it may be possible to increase the printing speed by repeating high-usage characters more frequently on the chain than low usage ones. This paper presents an analytic method of accurately estimating the printing speed of a chain printer for any character arrangement and describes a technique for determining the number of copies each character should have on the chain so that the printer will operate at or near maximum speed. Using these methods, significant increases in printing speeds have been obtained for actual applications.

Finite-State Processes and Dynamic Programming, R. M. Karp and M. Held, *SIAM Journal of Applied Mathematics* **15,** No. 3, 693–718 (May 1967). This paper develops a formalism within which the application of dynamic programming to discrete, deterministic problems is rigorously studied. The two central concepts underlying this development are discrete decision process and sequential decision process. Discrete decision processes provide a convenient means of problem statement, while monotone sequential decision processes (which are finite automata with a certain cost structure superimposed) correspond naturally to dynamic programming algorithms. The representations of discrete processes by monotone sequential decision processes are characterized, and this characterization is used in the deviation of dynamic programming algorithms for a variety of problems.

Hardware: A Matter of Logic, Memory, and Timing, T. J. Harrison, *Control Engineering* **14,** No. 11, 74–79 (November 1967). Effective control programming does not require in-depth knowledge of computer circuits. But understanding how hardware responds to software is important in writing programs that will get improved process results from control computers. Therefore, the basic building blocks of a digital computer from a systems engineering viewpoint are presented.

Information Theory and Systems Concepts in Reprography, H. J. Zweig and C. E. Nelson, *Proceedings of the Second International Congress on Reprography,* Cologne, Germany (October 25, 1967). Several concepts of information theory are reviewed with reference to their application in optics and photography. Some applications of recently developed concepts are given, and the importance of augmenting these concepts with those derived from psychometrics as well as systems analysis was stressed.

Investigations in the Design of an Automatically Repaired Computer, W. G. Bouricius, W. C. Carter, J. P. Roth, and P. R. Schneider, *Digest of the First Annual IEEE Computer Conference* **16C51,** 64–67 (September 1967). To achieve ultra-reliability, an Automatically Repaired Computer (ARC) organization is being investigated. The computer consists of a number of suitably interconnected functional units, each consisting of a set of identical modules capable of performing the required function. The ARC is organized to detect module malfunctions and, after diagnosis, to perform automatic repair. Methods for designing ultra-reliable module switching, reconfiguration within modules and addressing storage after an error are presented. A reliability model plus an on-line program to perform the calculations allow engineer interactive control to achieve a balanced computer design.

Language for Modeling and Simulating Dynamic Systems, R. J. Parente and H. S. Krasnow, *Communications of the ACM* **10,** No. 9, 559–567 (Sept. 1967). The advent of a new generation of computers, coupled with the continued growth of interest in simulation as a tool for decision making, has helped to motivate the development of the language discussed in this paper. This language is indebted to prior languages for some basic concepts of discrete system modeling, which they first clarified and embodied; however, the language results from the postulation of a single simulation system that facilitates both the modeling and experimentation aspects of a simulation study. An introduction to the language is presented, rather than a formal definition. Emphasis is placed upon those features which contribute most directly to the overall power of the language, with only limited attention being given to the many details that must also be provided. Most of the discussion is based upon a sample simulation study.

A Microprogrammed Implementation of EULER on IBM/SYSTEM/360 Model 30, H. Weber, *Communications of the ACM* **10,** No. 9, 549–558 (September 1967). An experimental processing system for the algorithmic language EULER has been implemented in microprogramming on an IBM SYSTEM/360 Model 30 using a second Read-Only Storage unit. The system consists of a microprogrammed compiler and a microprogrammed String Language Interpreter, and of an I/O control program written in SYSTEM/360 machine language. The system is described and results are given in terms of microprogram and main storage space required and compiler and interpreter performance obtained. The role of microprogramming is stressed, which opens a new dimension in the processing of interpretive code. The structure and content of a higher level language can be matched by an appropriate interpretive language which can be executed efficiently by microprograms on existing computer hardware.

The Organization of Computations for Uniform Recurrence Equations, R. M. Karp, R. E. Miller, S. Winograd, *Journal of the Association for Computing Machinery* **14,** No. 3, 563–590 (July 1967). A set of equations in the quantities $a_i(p)$, where $i = 1, 2, \cdots, m$ and p ranges over a set R of lattice points in n-space, is called a system of uniform recurrence equations if the following property holds: If p and q are in R and w is an integer n-vector, then $a_i(p)$ depends directly on $a_j(p - w)$ if and only if $a_i(q)$ depends directly on $a_j(q - w)$. Finite-difference approximations to systems of partial differential equations typically lead to such recurrence equations. The structure of such a system is specified by a dependence graph G having m vertices, in which the directed edges are labeled with integer n-vectors. For certain choices of the set R, necessary and sufficient conditions on G are given for the existence of a schedule to compute all the quantities $a_i(p)$ explicitly from their defining equations. Properties of such schedules, such as the degree to which computation can proceed "in parallel," are characterized. These characterizations depend on a certain iterative decomposition of a dependence graph into subgraphs. Analogous results concerning implicit schedules are also given.

Pattern Recognition and Eye-Movement Parameters, J. D. Gould, *Perception and Psychophysics* **2,** No. 9, 399–407 (1967). Pattern perception was studied by recording eye movements while *S*'s visually scanned nine simultaneously presented patterns of asterisks for target patterns. Pattern parameters studied were: similarity of target patterns to non-target patterns (absolute difference in the number of elements), number of target elements, and frequency of targets. Systematic correlations between the first two pattern parameters and eye-movement parameters were found. Mean duration and mean number of fixations on targets and also on non-targets increased with increased similarity. Mean duration and mean number of fixations increased only on targets with an increase in the number of target elements. Non-target patterns were perceived more quickly than targets. Fixations of longer duration were required to perceive the original target than to identify the other target patterns subsequently. The eye-movement results provide the basis for developing inferences about higher order processing of visual stimuli.

Peripheral Data Traffic in Modern Data Processing Systems, K. Ganzhorn and W. Walter, *Jahrbuch des elecktrischen Fernmeldewesens* 1967, 9–86 (1967). The article describes the information interchange between the central processing unit and the peripheral I/O devices by means of data channels. The structure, data flow and control principles of the two basic types, the selector channel, and the multiplexor channel are described. Data channels are control and data links. They communicate with I/O devices via a standard interface and associated control units. The control unit executes decentralized control of I/O devices and acts as a buffer compensating for differences in flow-rate, speed, coding, etc., between the channel and the I/O devices. In this way, a wide variety of I/O devices can be hooked up to the central processor. The principles of operation and peculiarities of the most important I/O devices are discussed; punched card and paper tape equipment, document readers, printers, I/O devices for teleprocessing and process control, optical display units with light pen, audio response units as well as external storage.

The Process Concept as a Basis for Simulation Modeling, G. P. Blunden and H. S. Krasnow, *Simulation* **9,** No. 2, 89–93 (August 1967). There are many systems whose behavior cannot readily or naturally be described by sets of differential equations. These range from computing systems, described in terms of the transportation, storage, and transformation of data, to transportation systems, described in terms of the movement of vehicles over networks of roads or rails. A body of digital computer techniques have emerged for the modeling and simulation of such systems, known loosely as discrete simulation to distinguish it from analog or continuous simulation. Recent

years have seen the extension of these techniques to increasingly comprehensive and detailed models applied to the study of large and complex systems. This progress has been facilitated by improvements in the supporting technology, ranging from faster and larger computers to more general and complete simulation languages. This paper first reviews some aspects of creating computer-based system descriptions and the functions performed by discrete simulation languages. It then suggests and outlines an approach based upon a unification of the handling of dynamic process and static entity.

Real-Time Definable Languages, A. L. Rosenberg, *Journal of the Association for Computing Machinery* **14,** No. 4, 645–662 (October 1967). The closure properties of the class of languages defined by real-time, on-line, multitape Turing machines are proved. The results obtained are, for the most part, negative and, as one would expect, asymmetric. It is shown that the results remain valid for a broad class of real-time devices. Finally, the position of the class of real-time definable languages in the "classical" linguistic hierarchy is established.

Real-Time Hardware-in-the-Loop Simulation Verifies Performance of Gemini Computer and Operational Program, J. L. Gross, *Simulation* **9,** No. 3, 111–118 (September 1967). An IBM Gemini Mission Verification Simulation (MVS) facility exercises and verifies the performance of the actual Gemini digital computer and its operational programs through real-time, hardware-in-the-loop simulations of Gemini missions. This paper gives a system-type functional description of MVS and describes the methodology used to verify performance of the actual computer through comparisons and analyses of mission profile data generated both by MVS and all-digital reference simulations. A description of the various types of mission profile data obtained together with a discussion of its significance and use is included. The error analysis techniques used to establish the expected deviations between mission profile data generated by MVS and the reference all-digital simulations are described.

Simulation of Electron Beam Control Systems Using DSL/90, M. Dost and R. R. Barber, *Simulation* **9,** No. 5, 237–247 (November 1967). The general-purpose digital simulation language DSL/90 was used to simulate two nonlinear feedback systems. The systems, required to control an electron beam in an IBM photo-digital mass memory are briefly described and their mathematical models are developed. One system regulates the filament current, the other periodically focuses the beam on the target. The system block diagrams are transcribed into sets of connection statements and associated control statements in step-by-step fashion to illustrate the simplicity and utility of programming engineering problems of this type.

The Structure of a General Design Automation System, J. Kurtzberg, *Digest of the First Annual IEEE Computer Conference* **16C51,** 50 (September 1967). A design automation system can be considered in terms of (1) the translation of system specification into equations and timing charts, (2) network design automation which deals with the physical realization of the logic and development of the computer backplane, (3) records automation which is concerned with documentation for wiring and maintenance purposes, and (4) clerical tests for design and circuitry inconsistencies. This last function is usually embedded among the design tasks. These general areas of design automation will be discussed, and the structure and design flow of a general design automation system will be presented along with an examination of specific tasks and computer procedures for handling them.

SYSTEM/360 Coding Techniques, R. H. Williams, *Data Processing* **9,** No. 8, 22–24 (August 1967). This article is a collection of coding techniques which can be used to make SYSTEM/360 programs more efficient, along with some recommendations on the use of symbols. Some of these techniques might be presented in the usual course of study, but many of them are unlikely to be found in any of the available documentation. This article will be especially helpful to those trying to learn the SYSTEM/360 on their own, but should also be useful to the trained (or even experienced) SYSTEM/360 programmer. The first section consists of coding techniques which allow the programmer to utilize the SYSTEM/360 hardware better. The second section describes some conventions in the use of symbols for more effective utilization of assembler facilities.

Techniques for the Optimal Design and Synthesis of Switching Circuits, G. D. Hachtel and R. A. Rohrer,* *Proceedings of the IEEE* **55,** No. 11, 1864–1877 (November 1967). The variational approach to the optimal design of high speed switching circuits is explored. The approach implements the variational calculus to obtain an expression for the vector sensitivity of a scalar performance function (e.g. delay, or switching time) to changes in the vector of design parameters. Gradient methods are established for using the vector sensitivity to iteratively update the parameter vector and obtain an optimal design. It is shown that the variational approach retains, typically, an M-fold computational advantage over conventional stop-and-repeat methods in determining the sensitivity of a scalar performance function to M design parameters. The approach is shown to be well adapted for incorporation into package analysis programs with matrix formulations, and vested with sufficient generality to be applicable to a wide range of switching circuit problems (e.g. Low-power or Large scale integrated circuits). It is further shown that subsumed in the general class of nonlinear parameter-value synthesis problems is the class of delay-minimization problems, and that the switching time minimization problem is a special case of the classical "Time-Optimal" problem.

* University of California.

Toward the Future Integrated Library System, R. W. Alexander, *Proceedings of 33rd Conference of FID and International Congress on Documentation* **II b 4,** 1–14 (October 1967). While the contents of future libraries are being amassed in publications and organized by documentalists for retrieval, the traditional forms of library operation are being shattered and recast. Reshaping is made necessary by the pressures of information supply and demand; the integrated systems evolving are made possible chiefly because computer services are becoming practical for libraries. With an active data base of all records (from purchasing to circulation), the library can take advantage of on-line mass storage, of versatile and economical terminals, and of the cost saving inherent in time-sharing. Thus implemented, computer-based systems can control the clerical burden, sustain and speed up established services, and provide personalized service keyed to individual interests. In making the transition, the system analyst and the librarian must identify essential library elements and discard those needed only to hold the old system together. The question is first "what shall we do?" and only then, "how shall we do it?" Reviewing these pre-design stages, this paper goes on to outline specific considerations for a system planned around an on-line real-time computer facility. It defines such a facility and describes appropriate terminals; it also discusses the software problems of file organization, integrity protection, and a programming structure sufficiently flexible to coordinate all the diverse tasks of the library.

The Yale-IBM Nuclear Physics Data Acquisition System, H. L. Gelernter,* J. Birnbaum, M. Mikelsons, J. D. Russell, F. Cochrane, D. Groff, J. F. Schofield, and D. A. Bromley,** *Nuclear Instruments and Methods* **54,** No. 1, 77–90 (September 1967). A collaborating group of Yale and IBM physicists have begun a sequence of particle-gamma correlation experiments at the Yale

Emperor tandem Van de Graff accelerator, where all data acquisition, analysis, display, and experimental control functions are to be performed within the framework of a general integrated hardware-software nuclear physics data acquisition system designed for the IBM SYSTEM/360 Model 44 scientific computer. The system, developed jointly by IBM Research and the Yale Nuclear Structure Laboratory, is intended to enable physicists to assemble the required hardware and software for any of a wide variety of experimental configurations with a minimum of attention to detail, and to change configuration with ease upon completion of a given experiment.

---

* Presently at the State University of New York at Stony Brook, New York.
** Yale Nuclear Structure Laboratory, New Haven, Connecticut.