acm [illegible]

Contents:

SEPTEMBER 1972 VOLUME 2 NUMBER [illegible]

SIGDA SPECIAL INTEREST GROUP ON DESIGN AUTOMATION

The goal of SIGDA is:
To enhance the utility of computers as engineering tools in the design, fabrication, and test of equipment and systems.

The objectives of SIGDA are:

1) To provide a means for the exhange of information concerning:
   a) Techniques
   b) Algorithms
   c) Computer Programs

   relevant to the computer assisted design of equipment and systems.

2) To publish quarterly newsletters containing news, technical information and opinions of interest to those involved in design and design automation.

3) To work closely with other ACM and professional committees and activities concerned with computer utilization as a design tool.

CHAIRMAN:

Charles E. Radke
IBM Corporation (B31/965)
P. O. Box 390
Poughkeepsie, New York 12602
(914) 463-4074

VICE CHAIRMAN:

Lawrence Margol
North American Rockwell
P. O. Box 3669
Anaheim, California 92803
(714) 632-8565

SECRETARY/TREASURER:

John R. Hanne
Texas Instruments (M/S 907)
P. O. Box 5012
Dallas, Texas 75222
(214) 238-3554

EDITOR:

Stephen Krosner
1581 NW 7th Street
Boca Raton, Florida 33432

MEMBERSHIP

SIGDA dues are $3.00 for ACM members and $5.00 for non-ACM members. Checks should be made payable to the ACM and may be mailed to the SIGDA Secretary/Treasurer listed above, or to SIGDA, ACM Headquarters, 1133 Avenue of Americas, New York, N.Y. 10036. Please enclose your preferred mailing address and ACM Number (if ACM member).

SIG/SIC FUNCTIONS

Information processing comprises many fields, and continually evolves new subsectors. Within ACM these receive appropriate attention through Special Interest Groups (SIGs) and Special Interest Committees (SICs) that function as centralizing bodies for those of like technical interests...arranging meetings, issuing bulletins, and acting as both repositories and clearing houses. The SIGs and SICs operate cohesively for the development and advancement of the group purposes, and optimal coordination with other activities. ACM members may, of course, join more than one special interest body. The existence of SIGs and SICs offers the individual member all the advantages of a homogeneous narrower-purpose group within a large cross-field society.

ACTIVITIES

1) Informal technical meetings at SJSS and FJCC.
2) Formal meeting during National ACM meeting + DA Workshop.
3) Joint sponsorship of annual Design Automation Workshop.
4) Quarterly newsletter.
5) Panel and/or technical sessions at other National meetings.

FIELD OF INTEREST OF SIGDA MEMBERS

Theoretic, analytic, and heuristic methods for:
   1) performing design tasks,
   2) assisting in design tasks,
   3) optimizing designs

through the use of computer techniques, algorithms and programs to:
   1) facilitate communications between designers and design tasks.
   2) provide design documentation,
   3) evaluate design through simulation,
   4) control manufacturing processes

CHAIRMAN'S MESSAGE - A SIGDA Business Meeting at ACM 1972

A small group of SIGDA members and interested parties met on Monday evening August 14th at the National ACM 72 Conference. I have reviewed the present status of SIGDA.

The financial status as reported was felt to be at a turning point. SIGDA's membership is stable and is growing (approximately 320 now). This was the first year in which SIGDA directly co-sponsored the DA Workshop (although $700 seed money was borrowed from Headquarters-ACM) M. J. Galey, Chairman of the 1973 DA Workshop indicates that approximately $1100 will be returned to SIGDA from the 1972 DA Workshop. This results in an approximate net return of $900. Unfortunately for the 1973 DA Workshop, our share of the seed is $1100 and again we will have to borrow $700 from ACM-Headquarters. This increase is because only SIGDA and IEEE Computer Society will be sponsors versus the previously three sponsors.

An inexpensive brochure on SIGDA, designed by John Rini, is scheduled to be printed in the next few weeks. The text is provided inside the front cover of the newsletter.

Headquarters-ACM has requested no deficit spending during our fiscal year, July 1, 1972 to June 30, 1973. Further, Hq-ACM has requested that SIG/SICs on a per member basis support one person in headquarters to handle SIG/SIC business. This may cost us an additional 30¢ per person. Instead of an increase in dues at this time (now $3 & $5) which would coincide with ACM's increase, I indicated that SIGDA will hold off on an increase. The additional $100 thus needed will be obtained from reduced expenditures and projected profits from DA Workshops as well as the assumed growth. It was felt that, given another 100 members, SIGDA would be able to publish three to four good Newsletters a year. Your efforts to get additional members in SIGDA are encouraged.

I indicated that the present officers' terms run through June 1973 and that I plan to have new officers introduced at the 1973 DA Workshop in June in Portland, Oregon.

In the past year, emphasis has been to let people know about SIGDA and the area of Design Automation outside of SIGDA. That is, to give DA and SIGDA exposure. That we have done with an increased number of technical sessions sponsored by SIGDA (SJCC 1972 and ACM 1972), and with a technical meeting and literature distributed at the DA Workshop 1972. This emphasis will continue and already we are planning a joint session with SIGGRAPH at ACM 73. Technical meetings at conferences will continue.

Obviously, the next thing to do is to push up the technical level of the newsletter. In order to meet the needs of different disciplines and the major areas of interest, I have asked Steve Krosner to be Chief Editor of the Newsletter. Steve has contributed to SIGDA as the representative to the 1972 DA Workshop Committee and as SIGDA session organizer at ACM 71 and ACM 72. He has a background in the development of digital computer design verification systems and is presently involved in supporting the marketing of special automated manufacturing systems for IBM. Walt Samek, who had previously asked to be relieved as Editor, has accepted the position of Associate Editor. Walt's background also ties development or design automation with manufacturing, but instead of the electronic computer area, Walt with Combustion Engineering of Windsor, Connecticut, represents the mechanical, metal fabrication, and piping areas of design automation. Jerry Paskusz and Larry Margol remain on as members of the Editorial Board.

Between Steve and Walt and, of course, you - the members, we should be able to greatly "up" the newsletter. Remember, the deadline for the January 1973 Newletter issue is December 15, 1972. Send Steve Krosner your DA article reviews and other tidbits early.

Professor Steve Szyenda has agreed to actively represent SIGDA on the DA Workshop 73 Committee. The full committee is listed elsewhere in this newsletter. For those of you who attended the 1972 DA Workshop in Dallas, Steve welcomes your comments, both good and bad, about the Workshop. It is my desire to make the professional organizations which sponsor the Workshop more visible at the Workshop. Plan now to attend on June 25-27,1973, in Portland, Oregon.

If you are planning to attend the 1972 Fall Joint Computer Conference, remember that SIGDA will be having a meeting on one of the evenings. Larry Margol is arranging for a technical presentation at the meeting so check the coming events for information.

The business of SIGDA was finally exhausted and six hungry DA enthusiasts headed for one of Boston's fine seafood restaurants.

HELP!   HELP!   HELP!

Chuck has asked the Newsletter staff to continue trying to increase the technical level of the Newsletter. We can't do this alone. We need your help.

This issue contains our second technical paper. If you have a short paper of interest, send it along.

Read any good books lately? Why not write a short review for us?

Tending a course in DA? How about sending us a bibliography to publish.

Suggestions for improvement, modifications, or just comments? Drop us a note.

As Chuck said, SIGDA is growing---Let's keep the Newsletter growing also!!

Steve Krosner

HELP!   HELP!   HELP!

REPORTS OF TECHNICAL MEETINGS

SIGDA Technical Meeting Held at 1972 SJCC

A joint meeting between SIGGRAPH and SIGDA was held at the SJCC on Tuesday evening, May 17th. The meeting was organized by SIGDA and was well attended by over forty-five persons.

Bill Sass of IBM-Kingston, New York, talked on "Computer Graphics in DA". Bill's experience since the birth of graphic displays was apparent when he reminiscenced with an individual from UNIVAC about a forerunner to UNIVAC's first display. The applications that Bill discussed included logic input, simulation, test generation, diagnostics, imbedding of interconnection in printed circuit card design, and circuit design.

Although somewhere between Hq-ACM and the hotel management the slide projector and screen were lost, Bill was able to amply and verbally describe his talk very well. This could have proven embarassing for Bill since he was trying to show the need for graphic displays in design automation.

An interesting point brought out in the discussion following was that in many specific application areas, graphic devices can be tailored to that application as the design aid. A display with maximum capability is required because of the lack of clear specifications.

Project LOGOS Reported in DA Workshop 1972

Over seventy-five persons attended a SIGDA technical meeting held on June 27th at the DA Workshop in Dallas, Texas. John Henne arranged to have Professor Chuck Rose from Case Western Reserve University talk on their efforts on Project LOGOS.

LOGOS is intended to be an automated design environment in which designers of interactive display terminals through a combination of algorithms can perform a total design of the operating systems and hardware of a data processing system.

Reference is made to an article by C. Rose and J. Barden in the March 1972 issue of the SIGDA Newsletter as well as Proceedings of COMPCON 1972 Conference (September 12-14,1972), where a series of fine papers were presented.

COMING EVENTS

SIGDA Meeting at FJCC

Mr. Larry Margol, the Vice-Chairman of SIGDA, will conduct the meeting. Mr. R. P. Larsen of North American Rockwell will speak on DA of custom MOS devices. He will discuss a design automation system used for high volume custom MOS devices. This system has been evolving at North American Rockwell Microelectronics for several years, and has been actively used in the design process over this interval. Of particular interest, will be a layout program which automatically defines masks starting from logic equations. Also included will be a discussion of Interactive editing of designs that require manual intervention.

10th DA Workshop 1973

Sheraton Portland Hotel
Portland, Oregon
June 25-27, 1973

Design automation is taken to mean the use of computers as tools which aid the design process and is often extended to include areas such as testing, simulation and certain portions of manufacturing. Typical examples of Design Automation involve the application of one or more functions to a given design area.

In addition to the topics traditionally covered in DAWs of the past, the following topics are being added.

Design Automation for LSI (special problems, areas of changing emphasis,....)

Circuit Design Automation (tolerence studies, simulation, optimization techniques,....)

Software Design Automation (can DA techniques be applied to software systems? At what level? In what areas,....)

Computer Aided Manufacturing (Since we designed it by computer can we build it by computer?)

We are especially interested in soliciting papers on these topics for they seem to be areas where the next big payoff will occur.

Requirements:

If you plan to submit a paper, you should send three copies to the program chairman no later than January 2, 1973. (Rough Drafts are acceptable.)

Notification of acceptance will be sent to you during the first week of February 1973. After notification of acceptance, you will receive detailed instructions on the format to be observed in typing the final copy. To insure the availability of the Proceedings at the Workshop, your final manuscript will be due April 23, 1973.

Final papers should be no longer than 5000 words, and the presentation should be limited to 20 minutes. Projection equipment for 35mm slides and vuegraph (overhead projector) foils will be available for every talk. Please indicate what, if any, additional audio-visual aids you require.

Topics of Interest:

<u>Design Areas</u>

Manufacturing Process
Architecture
Mechanical
L S I
Electronic
Firmware
Software
Total Systems

<u>Functions</u>

Partitioning
Packaging Placement
Wiring
Analysis
Simulation
Design Verification
Testing/Quality Control
Interactive System
Design Language
Change Control
Theory

<u>Sponsors</u>

ACM (Association for Computing Machinery) Special

Interest Group of Design Automation

IEEE (Institute of Electrical and Electronics Engineers)Computer Society

Rough drafts are to be sent to the Program Chairman:

R. B. Hitchcock
IBM Watson Research Center
P. O. Box 218
Yorktown Heights, New York 10598

Accompanying the draft should be the full name, address and telephone number of the principal author, with whom all further direct communications will be conducted.

<u>ACM '73</u>

Plans include an interface session with SIGGRAPH on the use of graphics in design automation and a general SIGDA session. If you are interested in participating, drop a note to either Chuck Radke or Steve Krosner with your ideas.

<u>TECHNICAL PAPER</u>

<u>Modular Requirements for Digital Logic Simulation at a Predefined Functional Level</u>

Prepared By: Mr. C. W. Hemming and Mr. S. A. Szygenda

Simulation of digital logic provides a viable technique for development and diagnosis of digital systems. Simulation models currently employed are discussed with a summary of structure and timing techniques. A methodology for functional simulation in conjunction with gate level simulation is discussed, presenting a representative set of predefined functions, and introducing a measure for predefined function performance. Errors in design detectable at the functional level are categorized.

Key words and Phrases: Logic Design, Digital Simulation, Functional Simulation, Fault Simulation, Diagnosis of digital systems.
CR Categories: 4.9, 6.1, 6.3, 8.1

NEEDS FOR SIMULATION OF DIGITAL LOGIC

Design Verification

The value of simulation in the role of design verification has been repeatedly demonstrated. Design Verification is accepted by most to mean that the logic correctly performs the function the designer intended, including detection of races and hazards, within the limits of the simulator. Occasionally, a subset of the design verification problem is considered, where timing, race, and hazard analysis are not performed; this is called logic verification.

The economic value of adequate design verification has become more and more apparent in the past few years for several reasons: two of these reasons stand out.

The first is the trend to widen use of asynchronous design (23). Since digital systems are being designed which are both faster and cheaper, design becomes more difficult to accomplish; and, indeed, design errors in large asynchronous stems (such as hazards) may not be found until many units have been built and sold. Additionally, it has been suggested that asynchronous machines are more easily diagnosed and repaired than clocked systems (3,4).

The second reason emphasized is the demands of highly integrated systems. Since the development of a prototype of an integrated module requires that masks be made, chips cut, etc., it is an extremely expensive process; and repeated iterations, to correct design defects, are prohibitive. Consequently, the rate of integration of complex systems, especially highly asynchronous ones, has been slow. A simulator capable of alleviating these problems, TEGAS system, has been described by Szygenda, et. al. (19, 20).

Diagnosis

Once the validity of a design has been established, the manufacturer is confronted with a set of problems still not well understood. These include how one can verify that a machine is correctly assembled, how the machine can be correctly repaired if it is not assembled properly, and how to repair a machine that fails.

Much work has been done on diagnosis of combinational logic, with substantial success (16, 26). Sequential diagnosis is still rather primitive; consequently, the need for simulation to aid in diagnostic development. Two major objectives of the diagnostician are to 1) determine if a machine is functioning correctly, and, if not 2) isolate the error to the smallest replaceable (repairable) unit in the system. Although the problem of generation of an economically viable test set for the general sequential system is not solved, the use of simulation has aided the theory, and has provided insight to reasonably sized test sets for many individual modules. Indeed, for these reasons, the origins of simulation are closely intertwined with diagnosis (1, 15, 16, 17).

It might be argued that since technology already allows building a complete processor on a chip, the only valid concern is fault detection. Such an argument is incorrect for the following reasons.

1) A large number of digital systems that will be in service for many years to come are not built that way (IBM/360, for example).

2) The economics of building large scale general purpose computers with multiple small processors is not established.

3) Development of reliable systems, even if completely integrated, can be substantially enhanced by simulation.

4) Adoption of a standard function set such as on logic cards, is a common and useful technique expected to endure. New designs are currently being implemented with established functional modules.

When one considers design verification and diagnosis in depth, it becomes clear that simulation can be used effectively to enhance these efforts. The next section of this paper will be devoted to a discussion of techniques and implementations used in digital simulators. The remaining sections are concerned with module specifications and a functional partition to be used in an element level simulation environment as described next.

Internal Structures

Two structures dominate simulation efforts. The majority of the simulators reported are compiled. A compiled simulator translates the description of the system to be simulated into code, executable by the host computer. Thus the AND instruction of the host machine would normally be generated to simulate an AND gate. Where fault simulation is to be performed, additional codes are generated for the allowable faults. A fault is defined as a physical defect in the system, causing it to operate incorrectly. A failure is a manifestation of the fault. A stuck-at-"0" (short to ground, say) on the C line of Figure 1 would result in E being "1", consequently F becomes "1." Inputs A and B lose any control.

A typical compiled output for the following three gate system (using a hypothetical instruction set for clarity) is shown below.

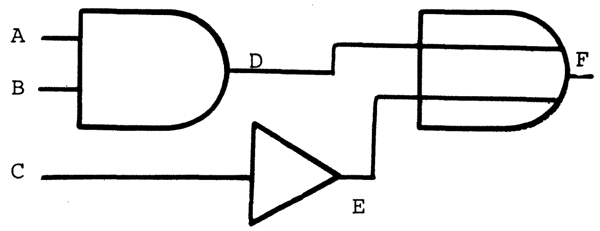

Figure 1 Example Digital Circuit

The code generated for the circuit shown in Figure 1 is as follows:

```
AND gate  CLEAR     / Clear Accumulator
          LOAD  A
          AND   B
          STORE D   / D is the AND of
                          A and B
          CLEAR
Inverter  LOAD  C
          COMP      / Accumulator con-
                      tains E now
          OR    D
OR gate   STORE F   / F is the logical
                      output
```

Note that both the inverter and the AND gate must be evaluated prior to evaluating the OR gate, forcing an ordering on the compiled code. This ordering requires that every input to a gate be determined (evaluated) before that gate is evaluated. The process by which the spatial relationship of the logic elements is determined and the resultant ordering imposed on the compiled code is called leveling.

Table driven simulators exhibit a somewhat different structure. A set of routines representing the various allowed functions is provided. The input system description is translated to tables which carry such information as 1) type of function (routines to be called), 2) logical interconnection of this function (its inputs and outputs), and 3) additional information such as propagation time, faults, etc. Excellent descriptions of table driven structure are given by Ulrich (22, 23).

The simulators discussed in the literature appear in relative agreement on the following attributes of the various structures. Compiled simulators are assumed to be faster, allow more elements per simulation run (if elements are elementary gates), and are generally difficult to implement or change; and, since a spatial ordering is imposed, do not consider timing. Table driven simulators allow relatively complete timing analysis (and consequently hazard analysis), can be written in a high level language, and are assumed to be somewhat slower and restricted in element count. Fault simulation is the process of simulating the behavior under fault conditions. A class of well defined faults is stuck-at-"1" (S-A-1) and stuck-at-"0"(S-A-0) faults.

Techniques have been developed to increase the operating speed of both table driven and compiled simulators. The technique known as stimulus bypassing (6,8) is associated with compiled simulators. A preliminary section of code is associated with each function for which stimulus bypassing might be effective. If the output of the function is not going to change for the current inputs, the section of code associated with the function is not executed (bypassed). For example, an RS flip-flop with both inputs zero could be bypassed (8). The timing gains due to stimulus bypassing have not been reported.

A different concept, known as selective trace (20, 23), is associated with table driven structures. Selective trace is based on the observation that if a gate's output does not change when evaluated, then the fan-out of this gate is unaffected by the excitation that caused evaluation of the current gate. Hence the current gate's output is not followed. This procedure has been reported to yield an 88% improvement in running time (22), and results with TEGAS2 have indicated an order of magnitude savings.

Zero Delay Simulation

Zero delay simulation is based on the Huffman model of sequential circuits. A detailed and readable description of the Huffman model may be found in Miller (14). The essentials of the Huffman model are presented in Figure 2.

The system is evaluated in "passes." A pass consists of applying the current values of the primary and secondary input vectors to the combinational logic, producing the new values of the primary and secondary output vectors. The secondary input vector is updated to correspond to the secondary output vectors, and the combinational logic reevaluated. The process is continued until a stable condition occurs.

A race occurs, in this model, if two or more secondary outputs change together. A critical race occurs if a race may cause an incorrect state to be assumed as the stable state. Of course, races in sequential networks are normally due to various delay conditions present in the system, which are not accounted for in the Huffman model, since timing is not considered. Leveling requires an element be evaluated after every input to that element is evaluated as previously discussed, but within the context of a pass as discussed above (25).

A number of simulators have been implemented using the zero delay model, but most do no race analysis. A major historical importance (17, 18). Other zero delay simulators have been reported, and are summarized in Table 1.

Simulator	Year Pub.	Compiled or Table Driven	Delay	Fault Simulation	Auto Fault Insert	Race-hazard Analy.	Spike Analy.	Max Elements	Simulator Time	Selective Trace	Implemented	Comments
Seshu & Freeman	62	Compiled	Z	Parallel	Yes	Race Only	No	-	1 min. per 10 failures	No	Yes	Primarily a heuristic program to isolate faults in sequential logic, by generating its own test vectors
Seshu 1604	64	Compiled	Z	Parallel	Yes	Race Only	No	300	-	No	Yes	4 strategies for test vectors, as above
Ulrich	65	Table Driven	A	No	No	No	No	8,000 Note 2	-	Yes	Yes	Macro elements can be defined. Claims to 25X improvement with selective trace
Saturn-Hardie & Suhockie	67	Compiled	Z	Parallel	No	Note 3	No	-	2 clocks /sec	Stimulus bypass on macro only	Yes	Instruction bypass requires hand leveling-Functional macros
1107 Yetter	67	Compiled	Z	Parallel	Yes	No	No	10,000	.45 usec component	No	Yes	No functional-has memory
Chang & Manning	68	Compiled	Z	Level	-	No	No	-	-	No	No	
Hays	69	Table Driven	Z	No	No	No	Note 1	3,000	-	No, but cks inputs	Yes	Wired ANDS & ORS
Jephson, et. al.	69	Table Driven	Z	No	No	Yes	Note 5	-	-	Yes	Yes	3 valued
McKay	69	Hardware & Software	-	Not Clear	Not Clear	No	No	Predict 36,000	-	Yes	No	
Ulrich	69	Table Driven	A	No	No	No	Note 4	-	-	Yes	Yes	Claims 7X improvement
Brewer	70	Compiled	-	-	-	-	-	-	-	-	-	
Cohen	70	Compiled	Z	Yes	No	No	No	-	-	Stimulus bypass	Gate Only	Functional for combinational only;table lookup
Lamp-Walford	70	Table Driven	U	Deductive Method	Yes	-	No	-	-	Yes	Yes	Interactive and batch
Szygenda Rouse & Thompson	70	Table Driven	A	Yes	Yes	Yes	Yes	5,000	-	Yes	Yes	

Abbreviations: A, assignable delay; Z, zero delay; U, unit delay.
Note 1: Checks that propagation occurs in less than specified time.
Note 2: The equation used to calculate this value in his paper does not consider overhead for table driven.
Note 3: Determines illegal input conditions; no race hazard.
Note 4: Input changes shorter than a certain minimum duration are not propagated.
Note 5: Not as defined within this work.

Table 1. Digital Simulator System Summary.

Delay Models

A serious shortcoming of zero delay simulators is that the Huffman model is an inaccurate representation of digital systems, particularly asynchronous ones. The major inaccuracy is the strong dependence of a simulated system's operation on both local and global timing. The distinction between these is not absolute, but rather depends on temporal distances through which feedback lines must propagate for system stability to occur. If timing could be taken into account, a far more accurate prediction of races can be made, hazards could be detected, and spikes could be detected. A spike is the condition where a module's inputs are changed faster than the propagation delay of the module.

The simplest technique employed to represent timing is to assign a constant delay of one time increment to each element in the system. Such a model is known as an unit delay model. A nice feature of the unit delay model is that the same leveling techniques discussed in the zero delay case may be used, except the procedure is time dependent rather than space dependent. In an actual implementation any differences in combinational logic disappear if hazard analysis is not done. For sequential, however, the technique is far more powerful than a Huffman model, since races can be analyzed to some extent (accurate race analysis cannot be performed since in reality all elements do not switch with unit delay.) However, spike analysis cannot be performed.

The next stage in refinement of the model is the association of a representative delay with each element or element type in the system. Such a simulator implements an assignable delay model. Whereas compiled simulators are typically zero delay, assignable delay simulators are normally table driven. Since each element has an associated delay, the ability to accurately model races and hazards is greatly enhanced, and spikes may be detected. Techniques to minimize the scheduling overhead have been implemented in the TEGAS 2 system (29).

Another feature of some simulators is 3 valued simulation (11,20). Three-valued simulations allow the association of an unknown state with element outputs. Table 1 presents a summary of major simulators described in open literature.

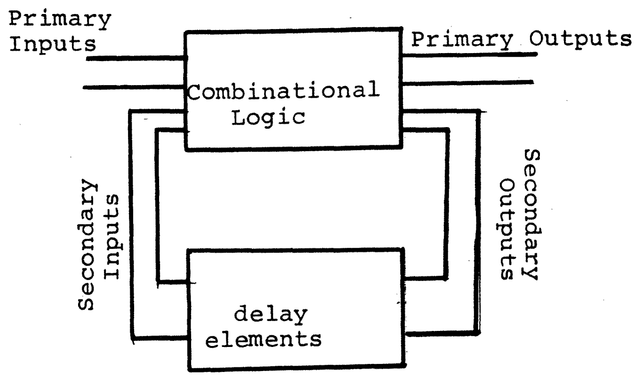

Figure 2. The Huffman Model of a Digital System

Limitations of Gate Level Simulators

As is evident from Table 1 and the preceding discussion, gate level simulators have constraints imposed in terms of the maximum number of gates which can be simulated, the validity of the model (particularly as regards timing considerations), and the requirement that MSI-LSI functional models be expanded to gate level representations for description of the system.

FUNCTIONAL SIMULATION

Past work in functional simulation has been rather infertile in terms of functions above flip-flops, with minor exceptions (including the extensions to the TEGAS2 system).

The earlier digital simulators mentioned the use of functional "macros" which were groups of code, inserted in the compilation, to simulate the functional element. The most complex function mentioned is the flipflop (8). Also, certain packaged elements were added to some implementations of the Seshu simulators.

Later, specific efforts were made by Chang and Manning (3,4) and by Cohen (6) to delineate systems oriented toward functional simulation.

Chang and Manning's work requires that systems be partitioned into well defined sections and described in terms of 1) multioutput combinational networks, 2) sequential circuits, and 3) register bus systems. Combinational networks are minimized by a Quine-McCluskey algorithm and simulated at the gate level. Sequential circuits are simulated by storing the flow table. Register-bus systems are simulated by storing the flow table. Register-bus systems are simulated by host machine memory data transfers. There is no discussion of spikes, race, or hazard analysis, fault insertion or timing in the model. The system has not, apparently, been implemented.

The second system was discussed by Cohen. The logic considered is restricted to zero delay combinational logic, which is quite restrictive.

It is desirable at this point to summarize the status of digital system simulation. Gate level simulators apparently have the potential now to accurate timing modeling, race and hazard analysis, parallel fault simulation, and efficient activity monitoring to avoid simulating needlessly. Functional simulation, on the other hand, has been limited to minor extensions of gate level simulation, with little consideration of variable length modules, techniques for handling general case functions not previously defined, or the effect of faults in the system. Speed and storage improvement due to functional extensions have received no analysis except for the storage analysis of Cohen's (6) table lookup. A thorough treatment of functional simulation along with synthesis and analysis of techniques which could be used for functional simulation has been considered by Hemming (27).

REQUIREMENTS FOR COMPLETE SYSTEM SIMULATION

In this section, the problems associated with using functional simulation for design verification are discussed, and the desirability of functional simulation is presented, along with the functions selected and the properties of these functions.

MSI-LSI Considerations

The current tendency in digital design is to buy as much of the system prefabricated as possible. Indeed, this has been the philosophy for a long time; until recently, though, the biggest practical prefabricated unit was on the order of a master-slave JK flip-flop, with several per package. Now however, dual 100-bit shift registers may be purchased, as well as many other complex functions.

While it is possible to model such a device at the gate level, or develop a flow table for such a device, the absurdity of imposing such requirements on the designer are apparent, especially when one considers the size of the flow table for even 36 bit counters. From a design verification viewpoint, the desired logical model would be a software package which evaluates the specified function and provides the appropriate outputs at the correct times.

Clearly, the advent of medium and large scale integrated circuits technologies imposes demands on system simulators previously unconsidered. Thus one of the requirements of functional simulation is to provide means for rationally describing systems composed of such functional modules. The techniques to be described are effective in this respect.

Artificial Boundary Problem

With the introduction of timing into the individual model, it is impractical, for the general case, to localize the activity of a network such that the network is logically partitioned under every input vector and for every simulation interval. Consequently, the incorporation of individual element timing in the system emphasizes the requirement for retaining the complete system description throughout the simulation pass. Paging techniques, as discussed by Szygenda (20), have presented a feasible solution. However, by incorporating the functions to be described, the total paging requirements can be reduced through increased storage efficiency, yielding improved simulation speeds.

Description of Systems and Systems Perspective

A major consideration in using a computer system is generating a complete error free description of the simulated system to the simulator. Experience has shown that with the TEGAS2 system, about 2*n cards are required to describe an n gate system, with n in the 100-500 gate range. Hence, effort required to produce a correct description of large system is considerable, and the time devoted to preprocessor checking is expensive. Indeed, it might well be argued that difficulty in producing correct descriptions could inhibit experimentation with the design, resulting in a poorer design. These considerations alone give impetus to a study of functional simulation strategies. With these considerations, we next discuss a set of functions and the implications of the selected functions.

PREDEFINED MODULES FOR SYSTEM SIMULATION

A number of factors are important when the basic decisions are made about a functional module set. In view of the desire for the best results most economically obtained, the fundamental considerations include such constraints as:

1) Accuracy of representation of the physical function which is being modeled
2) Storage requirements
3) Execution time
4) General applicability to digital systems
5) Ease of use

However, other factors are of importance when considerations of fault insertion and test pattern generation are introduced. Optimization of a procedure for design verification can result in a system unsuitable for fault insertion, particularly where sequential circuits are concerned. The consequences of such conflicting requirements are considered later.

The Normal Fundamental Mode Model

There are two broad categories of sequential circuits, so called fundamental mode and pulse mode (12). The mechanisms used to evaluate the models are sufficiently different that incorporating both modes in one model is unrealistic. The model adopted is the normal fundamental mode model, discussed by Unger (21). This constraint allows one input change only, and only when memory elements are stable; so an input that changes from 0 1 cannot change from 1 0 until the system is stable. The requirement that only one input change at a time is often unnecessary, and when this requirement is dropped the model is called the fundamental mode model.

To illustrate the differences between fundamental mode and pulse mode, the following example is helpful. Consider the circuit in Figure 3.

Analysis of this circuit produces the state transition maps shown in Figure 4. The X's shown in the figure denote states which cause oscillation to occur and are illegal. The important thing to note is that the state (A,B,C,D) = (1,1,0,1) is illegal in fundamental mode, but not in pulse mode and other differences appear. Consequently, tables generated for this circuit would be different depending on the mode of simulation desired. Similarly, the evaluation equations would be different. If the illegal input conditions are detected and evaluation stopped unless the inputs are valid, the system could be evaluated as in equations (1) and (2) for pulse mode:

$$C = DA + C\overline{B} \qquad (1)$$

$$D = \overline{C} \qquad (2)$$

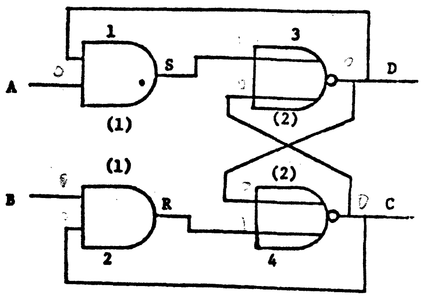

Figure 3. Mode Representation Example

(A)

Pulse mode transition map

AB CD	00	01	11	10
00	X	X	X	X
01	01	01	10	10
11	X	01	X	10
10	10	01	01	10

(B)

Fundamental mode transition map

AB CD	00	01	11	10
00	X	01	X	10
01	01	01	X	10
11	X	01	X	10
10	10	01	X	10

Figure 4. Transition Maps for Mode Representation Example.

however, for fundamental mode, the evaluation would use equations (3) and (4).

$$C = A + C\overline{B} \qquad (3)$$

$$D = \overline{C} \qquad (4)$$

The critical factor is the duration of the input changes. As a result of the requirement to differentiate between modes in the development of the modules which follow, the normal fundamental mode model is the basis for consideration of timing in the system. Also, the procedure to convert gate level representations to tables is dependent on this requirement.

Requirements of a Predefined Function Set

The criteria for selection of a set of functions is as follows:

1) Completeness
2. Broad applicability
3) Function delay time
4) Fault propagation

The first requirement is completeness, by which we mean that any arbitrary digital function may be constructed from AND gates, OR gates and INVERTERS (12). These essential functions, then, become the base for a complete set. Next, we consider the additional functions desired for broad application.

When the applicability and ease of use of the function set are considered, we depart from the realm of mathematical completeness and enter the domain of engineering judgment. Within this context, broad applicability may be interpreted as follows:

a) the set represents the major physical functional types found in digital systems, except main memory;

b) the functions are not constrained to implementation on a particular computer;

c) the functions produce a contribution improving overall system simulation beyond the level attained with gate level simulation.

The first consideration, that the major function types be represented, requires further clarification. Recall that the objective is to represent the logical characteristics of the function performed, not necessarily the implementation of the function itself (for the implementation, a gate level description would be correct). Consequently, the difference between ripple carry, carry lookahead, conditional sum, and the various other methods of implementing an adder are not considered logically significant at the adder terminals, except as follows:

1) propagational delay;
2) occurrence of multiple transitions on the outputs
3) propagation of faults.

Non-logical characteristics, such as the voltage levels required for a zero, are not considered here.

With the preceding considerations, a study of general machine characteristics was performed to identify suitable extensions to this set, and the following set of functions was developed.
<u>Combinational</u>: Adder, Decoder, Encoder, Shifter, And, Or, Inverter.
<u>Sequential</u>: Counter, Register

Since the table driven structure is adopted as the internal representaton of these functions, a separate propagation delay may be associated with each occurrence of a function in the simulated system, retaining the first logical property, propagation delay. The delay is assumed to be the same for every output, such that all outputs are updated after the defined delay time for the function. If it is necessary to associate separate delays with each output, a tabular technique could be used; alternately, that function could be simulated at the gate level.

The second logical property, noticeable at the pins, is multiple transitions prior to stability. This is actually hazard analysis, and is implementation dependent; consequently, it is not included in these functions. A procedure has been developed for building a tabular description of a gate level function which detects multiple outputs (27).

The second requirement, for broad applicability of functions, was a degree of machine independence. Since the vast majority of processors are binary, the primary problem is associated with word length. Three factors are of concern; the word length of the hose (the machine performing the simulation), the word length of the simulated machine, and the relationship of the two.

A basic function length of 4 was selected. Implementation of these functions requires a host machine word length of at least 8 bits. This allows automatic transfers from one machine to another, irrespective of simulated machine or host machine word length (with an 8 or more bit host).

The selection of a basic 4 bit function requires that a technique be

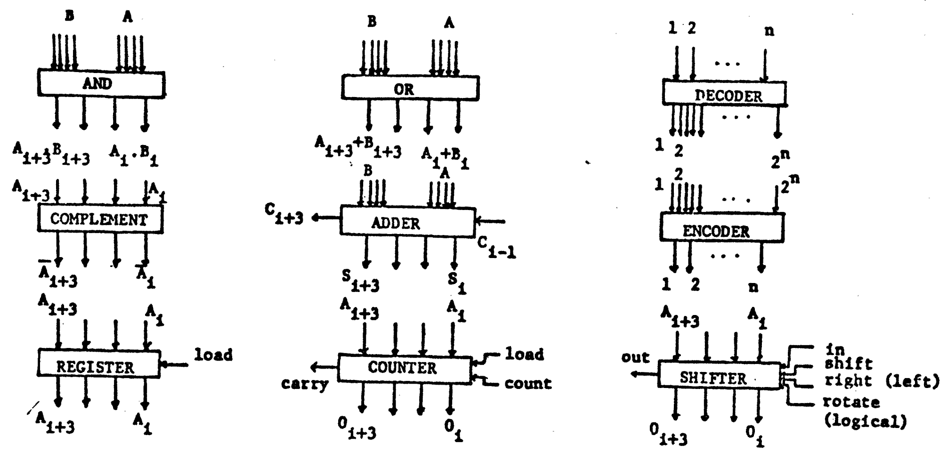

Figure 5. User View of Predefined Functions

adopted to provide longer lengths. The method developed is defined as linear concantenation. Linear concantenation is the process of extending a function by repeating its basic length; no other parameter is changed. Additionally, linear concantenation requires that the relationship between the number of inputs and the number of outputs be linear.

Since the encoder and decoder do not satisfy the linear relationship required, linear concantenation cannot be used. By supplying decoders with 2 to 6 inputs, a set of output counts can be covered. That is, every output count from 4 to 64 is available, directly or indirectly, by selecting one of these functional encoders or decoders.

Finally, the requirement to represent individual flip-flops, the requirements of fault insertion, and the preservation of complex timing make it necessary to add individual gates to the set. The elementary gates types provided in TEGAS2 as shown in Table 2, fill out the set.

In summary, a set of functions has been defined which represents a significant portion of the identifiable functions in general purpose computers.

Propagation delay may be modeled due to the simulator structure assumed. Variations in word length have been considered, and the process of linear concantenation allows indefinite extension of applicable functions. Combined with the elementary gates from TEGAS2, complete digital systems may be representated within the limits of the number of elements which can be handled, wich is limited by the host memory size.

N-Input AND Gate	N-Input OR with fault
N-Input OR	Insertion
INVERTER	N-Input NAND with fault
Primary Input	Insertion
Primary Output	N-Input NOR with fault
N-Input NAND	Insertion
N-Input NOR	Single Input INVERTER
Printer	with Fault Insertion
Delay	Bus Register
N-Input Exclusive	OR/1's Complement 3 Bit
JK Clear FF;J,K	Adder
Clock,Clear	Combinational Encoder
JK FF; J,K,Clock	Combinational Encoder
SR FF;Clock,S,R	Bus Builder
T FF;ClockT	Clock
D FF;Clock, D	JK FF;J,K,Clock
N-Input AND with	
fault	Clear, Set
Insertion	SR FF; S,R (no clock)

Table 2 Logical Types Available in TEGAS2

Functional Element Procedures

Procedures for predefined elements been developed (27). These reflect the 4 bit width previously discussed, and require a host machine with a minimum 8 bit word width. These routines require an add instruction as well as a logical shift left and logical shift right in the host.

When properties of the simulated machine exist in the host, it would be advisable to modify these if the simulated machine is to be simulated often. Such modifications make the simulator more specialized, and, since generality is of interest here, these modifications are not considered in depth. However, parallel fault insertion is very dependent on width, and need be considered for the general case.

Timing and Storage Requirements

The timing and storage requirements for functional evaluation, using the functions presented, may be compared to a representative gate level implementation for the function. Using typical representations, this comparison is performed analytically. The basic four bit functions are considered. Linear concantenation would not change the result, since the procedures would be evaluated twice and the number of gates would double.

Storage requirements are based on the count of the separate inputs and outputs required. For the gate level representation, this includes all internal variables. Functional representation, of course, has no explicit internal variables.

Timing analysis is very dependent on fault insertion considerations and is not included in this paper.

The modified improvement index (MII) defined as:

$$MII = \frac{B}{A}\frac{D}{C} \quad (.1)$$

where A = simulation time for the function description
B = simulation time for the gate level description
C = storage required for the functional description
D = storage required for the gate level description

and the result is multiplied by 1/10 to reflect improvements due to selective trace. The average of the MII is 2.6, indicating potential improvements in storage requirements and simulation speed for functional simulation.

CONCLUSIONS

These methods provide a profitable extension to the gate level simulation strategy. The functions are complete, have broad application, model the basic elements of a digital system, and can model certain critical parameters such as delay and input error conditions. Several predefined functions have had preliminary testing in the TEGAS2 system. A structural example network yielded a simulation time improvement of 3, somewhat as expected from the modified improvement index, for a fault free simulation. Parallel fault simulation at the gate level imposes storage considerations which deter functional simulation. A modified storage strategy is being investigated for TEGAS2 system.

Finally, it is apparent that no predefined set will be adequate, in the general case, to define a system without resorting to elementary gates at various points. The inclusion of all functions would be exorbitant and it really is not necessary. For special functions, such as the AB flip flop shown in Figure 3, the actual simulation at the gate level would be performed. Alternately, table driven techniques (27) might be used. A second deficiency in the predefined set previously discussed is that the interconnection between functions are simple busses, and consequentially every output appears simultaneously after the specified module delay. In certain instances this model may not be adequate. For example, a carry completion adder has timing which is dependent on the applied data. This problem may also be approached with table driven techniques.

The utilization of predefined functions capitalizes on the components of the host structure. Use of this knowledge can produce an increased improvement index with no loss of simulator generality, with a resultant faster and more powerful digital simulation system.

FUNCTION	COMPARATIVE GATE LEVEL TYPE	A FUNCTION TIME	B GATE LEVEL TIME	C FUNCTION STORAGE	D GATE LEVEL STORAGE (1 bit/word)	MODIFIED IMPROVEMENT INDEX (B/A)*(D/C)*.1
Adder	Ripple carry adder	4	48	3	67	7.7
Decoder 2input, 4output	direct boolean	3	16	2	8	2.0
Encoder 4input 2output	direct boolean	3	8	2	6	.6
Shifter	single gates (logical)	3	3	1.2		
And, Or	direct boolean	4	16	3	12	1.6
Inverter	direct boolean	3	12	2	8	1.6
Counter	4 flipflops	4	16	2	10	2.0
Register	4 flipflops	2	16	2	10	4.0

Table 4: Storage and Time Comparisons
Functions and Equivalent Gate Representations.

REFERENCES

1. Armstrong, D. B. "On Finding a Nearly Minimal Set of Fault Detection Tests for Combinational Logic Nets." IEEE Trans. on Elec. Computers, vol, EC-15, no. 1, Feb. 1966, pp. 66-73.

2. Breuer, M. A. "Functional Partioning and Simulation of Digital Circuits." IEEE Trans. on Computers vol. C-19, Nov., 1970, pp. 1038-46.

3. Chang, H. Y., Manning, E. G., and Metze, G. Fault Diagnosis of Digital Systems. 1970, John Wiley & Sons, New York.

4. Chang, H. Y., and Manning, E. G. "Functional Techniques for Efficient Digital Fault Simulation." Digest of the First IEEE Computer Group Conference, 1967.

5. Chu, Y. Digital Computer Design Fundamentals. 1962, McGraw-Hill, New York.

6. Cohen, D. J. Computer Based Fault Analysis of Digital Systems. Research Report CSRR2020, University of Waterloo, Dept. of Applied Analysis and Computer Science, Waterloo, Canada, 1970.

7. Gschwind, H. W. Design on Digital Computers. 1967, Springer Verlag, New York.

8. Hardie, F. H., and Suhockie, R. J. "Design and Use of Fault Simulation for Saturn Computer Design." IEEE Trans. on Elec. Computers, vol. EC-16, August 1967, pp. 412-29.

9. Hays, G. G. "Computer-Aided Design: Simulation of Digital Design Logic." IEEE Trans. on Computers, vol. C-18, no. 1, January 1969, pp. 1-10.

10. Husson, S. S. "Microprogramming Manual for the IBM System 360 Model 50," IBM TR 00.1479-1, IBM Systems Development Division, Poughkeepsie, New York, 1967.

11. Jephson, J. S.; McQuarrie, R. P.; and Vogelsberg, R. E. "A Three-Value Computer Design Verification System." IBM System Journal, no. 3, 1969, pp. 178-80.

12. McCluskey, E. J. Introduction to the Theory of Switching Circuits. 1965 McGraw-Hill, New York.

13. McKay, A.R. "Comment on Computer-Aided Design: Simulation of Digital Design Logic". IEEE Trans. on Computers, September 1969, p. 862.

14. Miller, R.E. Switching Theory 2 vols. 1965, John Wiley & Sons, New York.

15. Poage, J.F. "Derivation of Optimal Tests to Detect Faults in Combinational Logic." Symposium on Mathematical Automata, pp 483-528, 1962 Brooklyn Polytechnic Press, Brooklyn.

16. Roth, J.P. "Diagnosis of Automata Failures: A Calculus and a Method. IBM Journal, July 1966, pp 278-291.

17. Seshu, S. "On an Improved Diagnosis Program." IEEE Trans. on Elec. Computers, vol. EC-14, February 1965, pp. 76-9.

18. Seshu, S., and Freeman, D.N. "The Diagnosis of Asynchronous Sequential Switching Systems." IEEE Trans. on Elec. Computers, August 1962, pp. 459-65.

19. Szygenda, S.A.; Rouse, D.; and Thompson, E. "A Model and Implementation of a Universal Time Delay Simulator for Large Digital Nets." AFIPS Proceedings of the SJCC. May 1970, pp. 207-16.

20. Szygenda, S. A., "TEGAS2 - Anatomy of a General Purpose Test Generation and Simulation System for Digital Logic." Proceedings of the 9th Annual Design Automation Workshop June, 1972, Dallas, Texas.

21. Unger, S.H. Asynchronous Sequential Switching Circuits. 1969 John Wiley and Sons, New York.

22. Ulrich, E.G. "Exclusive Simulation of Activity in Digital Networks." Communications of the ACM, vol. 12, no. 6, February 1969, pp. 102-110.

23. Ulrich, E. G. "Time-Sequential Logical Simulation based on Circuit Delay and Selective Tracing of Active Network Paths.: Proceedings of the 20th ACM National Conference, August 1965.

24. Walford, R.B. "The Lamp System". Proceedings of Workshop on Fault Detection and Diagnosis in Digital Circuits and Systems, Lehigh Univ., December, 1970.

REFERENCES (cont'd)

25. Yetter, I. H. "High Speed Fault Simulation for Univac 1107 Computer System." Proceedings of the 23rd ACM National Conference, 1968, pp. 265-277.

26. Szygenda, S. A. and Goldbogen, G. C. Implementation and Extension of Multi-Dimensional Path Sensitizing in a Simulation and Diagnostic System," Proceedings Seventh Annual Allerton Conference on Circuit and System Theory, University of Illinois, Monticello, Illinois, October 1969.

27. Hemming, C. W. "Functional Simulation Techniques for design verification and fault insertion in electronic digital systems." Ph.D. Dissertation, CS/OR Center, Institute of Technology, Southern Methodist University, Dallas, Texas, September 1971.

28. Szygenda, S. A., Hemming, C. W., and Hemphill, J. M. "Time Flow Mechanisms for Use in Digital Logic Simulation," Proceedings of the Fifth Annual Conference on Applications of Simulation," (ACM) New York, N. Y., December, 1971.

29. Szygenda, S. A., et al., "Implementation of Synthesized Techniques for a Comprehensive Digital Design, Verification and Diagnosis System," Fifth International Conference on System Sciences, University of Hawaii, January, 1972.