LABORATORY FOR
COMPUTER SCIENCE

MASSACHUSETTS
INSTITUTE OF
TECHNOLOGY

MIT/LCS/TM-166

REPORT ON THE WORKSHOP
ON
SELF-TIMED SYSTEMS

Randal E. Bryant

May 1980

545 TECHNOLOGY SQUARE, CAMBRIDGE, MASSACHUSETTS 02139

Report on the

Workshop on Self-Timed Systems

Randal E. Bryant

Massachusetts Institute of Technology
Laboratory for Computer Science

[illegible] Massachusetts 02139

CONTENTS

1. Introduction

The Workshop on Self-Timed Systems was held at the MIT Endicott House on July 8-12, 1979. This workshop served to bring together experts in the field of self-timed systems to review and assess the state of the art and to chart directions for future research. For the purpose of the workshop, self-timed systems were defined to include any system composed of a set of modules which communicate asynchronously. The modules, however, may themselves be implemented either synchronously or asynchronously. It is hoped that the advent of custom LSI parts offers a new opportunity for the application of self-timed principles. The strong bias of conventional, standard IC parts toward clocked systems may no longer limit the practicality of self-timed designs.

Funding for the workshop was provided by the Department of Energy under contract number DE-AC02-79ER10473.

This report consists of abstracts submitted by the participants and then editted to give a more uniform style of presentation and level of detail. The speakers have not had a chance to review the editted abstracts and correct any errors which may have occurred. Any opinions expressed in the abstracts are those of the speakers and not necessarily of their institutions or of the sponsoring institution.

I would like to acknowledge the assistance of Clement Leung in preparing the bibliography for this report, and of Jack Dennis, Clement Leung, and J. Dean Brock for reviewing preliminary versions of this report. The guidance and ideas for this workshop were provided by Jack B. Dennis.

2. Summary of Presentations

Opening Remarks

1. Jack Dennis, MIT

Welcome to the Workshop on Self-Timed Systems. Little work on self-timed digital systems has been done since many of us got together at Woods Hole nearly ten years ago. It seems that available commercial devices have such a strong bias toward clocked systems that it has not been possible to build practical systems in this form. However, the prospect of design and manufacture of custom LSI devices changes this picture dramatically and offers a new opportunity to explore the possible advantages of using self-timed principles in digital system design. We have organized this workshop to assess this question, to ascertain the state of the art in principles and methodologies for self-timed system design, and to identify profitable areas for further research. It is good to see you again.

Session on System Design Methodologies

Chairman: J. Dennis

1. L. R. Marino, San Diego State University

Methods for Handling Asynchronous Inputs to Sequential Networks

Asynchronous inputs are a source of difficulty in designing both synchronous and asynchronous sequential networks. Different methods can be applied depending on the type of system and the nature of the application.

For synchronous networks, an input that is asynchronous with respect to the network clock may result in failure to satisfy the setup requirements of the state register. Conventional methods for handling such inputs involve the use of a synchronizing flip flop, but it may have an extended resolution time due to the metastable state. The probability of network failure as a result of extended resolution times may be made arbitrarily small only by making the sampling interval arbitrarily large and thus making the recognizable frequency of the asynchronous input arbitrarily small. Hence this is a reasonable approach only when asynchronous input changes are relatively infrequent.

In a recent paper by Wormald [62], it was suggested that the extended resolution time of synchronizers may be prevented by using a Schmitt trigger in the feedback loop of the latch. It can be shown using a phase plane analysis that this is not the case. The metastable state exists in spite of the hysteresis of the Schmitt trigger, and trajectories that pass close to this state will have extended resolution times.

Asynchronous sequential networks may be designed to process asynchronous inputs directly as described by Unger [58], provided that runt pulses occur neither in the input variables nor in the state variables. Theoretically, inertial delay elements may be used to enforce this constraint. Inertial delays can also be used to eliminate extended resolution times of synchronizer flip flops in synchronous networks. However, an inertial delay that can completely eliminate runt pulses has yet

to be designed, and it appears unlikely that it is possible to do so.

Finally, it is possible to use synchronous design techniques and yet avoid completely the failures due to extended resolution times of synchronizers by using the combined clock-synchronizer circuit of Pechoucek [43]. Similar circuits were independently designed by M. Stucki and C. Seitz.

H. T. Agerwala, IBM, Yorktown Hts., NY

A Design Methodology for Digital Systems Using Petri Nets

I have developed a class of Petri Nets called Synthesis Petri Nets as a design language for digital systems. It has been shown that these nets can describe any sequential or combinational circuit and have the capability to describe concurrent operations. Digital system descriptions can be synthesized automatically from Synthesis Petri Nets. The procedure follows the top-down approach and allows the description of a system at any level of detail. The gate level is obtained through the use of the transition gate, a new gate proposed for the implementation of Petri Nets. The advantages of the transition gate over three other existing approaches are that it can be used to implement larger classes of Petri Nets and that circuits implemented with it exhibit better fault response properties. The fault response properties refer to the percentage of faults that can be detected and isolated. I have developed fault detection and isolation techniques for the design method.

The method presents the following advantages:

1. A single language is used through all the stages of the design procedure. This allows the use of a single simulator for design verification at any stage and facilitates functional level simulation.

2. Circuits designed by the method are speed independent, hence they do not exhibit races or hazards. Design verification through simulation of speed independent circuits require only three-valued, unit delay simulators. These simulators are much simpler than the five valued accurate delay simulators required for design verification in conventional circuits. Basic ideas for the development of a Petri Net simulator have been developed.

The method presents one disadvantage: In circuits designed by the method, 85.7% of the internal faults (internal to the chip) and all pin faults representable as stuck-at faults are automatically detectable, are fairly easy to isolate, and cannot produce wrong states or false output signals. Nevertheless, the remaining 14.3% of the faults could exceed the number of distinguishable stuck-at faults exhibited by an equivalent conventional circuit. I have developed some implementation approaches to ease this situation.

III. C. Seitz, Caltech

Self-Timed System Concepts

In my view of self-timed systems, only events within an *equipotential region* may be assumed to obey a particular time ordering unless they are causally related. For signals propagating beyond an equipotential region, an arbitrary delay must be assumed. The equipotential region is a formal expression in the self-timed discipline of the importance of relating logical and physical locality. Signalling between local elements is both faster and simpler.

Self-timed systems are composed of a set of elements, each of which must be contained within an equipotential region. The specification of an element includes both *functional* relations, or constraints on outputs, and *domain* relations, or constraints on inputs. A system is defined recursively as either an element or as a legal interconnection of systems.

Rules have been developed for constructing self-timed systems for which the closure has been verified and the physical realizability has been demonstrated. These are discussed in Chapter 7 of Mead and Conway [48].

Session on Future Applications

Chairman: A. Davis

I. C. Seitz, Caltech

System Timing in VLSI

The scaling of the feature sizes of switches and wires will have a major effect on system timing in VLSI. Signal propagation will obey a diffusion process model of time delay due to the resistance and parasitic capacitance of wires. Such delays scale as the square of the wire length, which will lead to intolerable clock-skew problems in synchronous systems. These findings were reported at the Caltech VLSI Conference [49].

II. Lloyd I. Dickman, Digital Equipment Corp.

Perspectives on Design Styles for Computer Systems

This presentation illustrates the reasons for which particular design styles (self-timed, synchronous, etc.) are used in computer system design. This provides the basis for understanding which aspects of computer architecture and technologies must change such that self-timed design techniques will be exploited. Three principal influences on the choice of design style are: the architecture which will be supported, engineering constraints on cost and performance, and product design concerns.

Architectural Influence:

Architecture (i.e. programmer visible features such as the instruction set) can influence the choice of an implementation style. In particular, contemporary architectures are fundamentally synchronous. That is, there are numerous places where instruction execution is synchronized with respect to events such as interrupts, page faults, data exceptions, etc. Strict rules specify the perceived order in which opcode fetching, address mode decoding, operand accessing, and execution are performed. High speed machines which operate on several instructions, or portions of a single instruction in parallel, contain extensive control logic to coordinate these actions such that the results are the same as if they were performed sequentially.

Data-flow and non-procedural architectures provide a high degree of potential parallelism by eliminating traditional instruction sequencing. This enhances the appeal of self-timed rather than synchronous circuit design for these implementations since self-timed logic design mirrors the desired architecture.

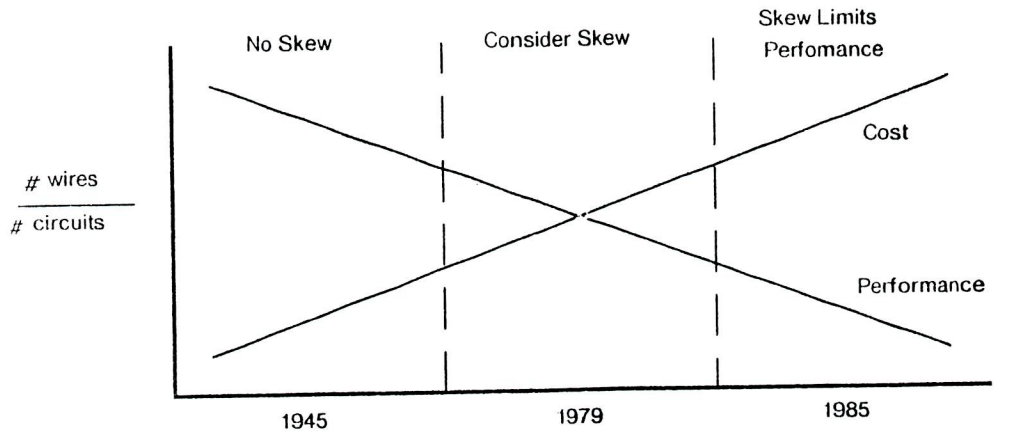
Engineering Influences:

Figure 1 depicts how the ratio of "wires" to "circuits" has influenced computer design over time. "Wires" are the paths which signals must travel between active circuit elements; these include actual wiring, printed circuit board etch and interconnects on integrated circuits. "Circuits" are active logic elements. Both wires and circuits can be characterized in terms of economic cost, and signal propagation performance.

The positively sloped line shows that the monetary cost of wires has been increasing over time relative to the cost of circuits. The negatively sloped line shows that the performance of circuits has

---

Fig. 1. Design Styles



increased relative to the performance of wires. Taken together, these two lines show that many of the original computers were designed under the constraints of expensive circuit elements interconnected by wires which propagated signals at essentially instantaneous speeds. However, contemporary computer design is undertaken in an environment where both circuits and wires are costly, and the signal propagation delay in wires is similar to the circuit delay. The expected divergence of the two lines in the future suggests that design must accommodate a world in which circuit elements are fast and cheap, while wires will be relatively slow and expensive.

Examination of Figure 1 is helpful in understanding why computer design styles have evolved over time. Early computer design used fast, cheap wires and slow, expensive circuits. Clock skew could be ignored and registers to store state information were minimized. Today's world achieves an engineering balance between costs and performance for both wires and circuits; synchronous design with consideration given to clock skew is a viable technique. The future is likely to find circuit elements which are much faster and cheaper than the wires which interconnect them. Since clock skew constrains minimum circuit times, it is unlikely that purely synchronous techniques will be used; self-timed systems are an intriguing possibility.

Product Design Influences:

The product design strongly influences the choice of circuit design style. The following considerations have caused synchronous techniques to be favored. First, systems are constrained; the need for open-ended extensibility which often motivates asynchronous design is not present. Second, buses to which the processor must attach are often synchronous. High speed data transfer requirements are best met where there is no "handshaking." Finally, the marketplace for mass-produced goods wants to see that each unit produced is identical to every other unit. One implication for the manufacture of computer systems is that each "identical" machine should run programs at approximately the same speed. Additionally, testing and reliability are areas which might benefit from circuits with deterministic speed characteristics.

Synchronous design as currently practiced by industry results in machines whose data paths are extremely well matched to the system clocks. The engineering effort to perform this tuning is believed well spent since a critical understanding of both wire and circuit delays is important where the signal transit time is evenly divided between the two. Furthermore, engineers have found synchronous systems simpler to design and debug. The ability to deterministically "halt" an implementation reduces the portion of a design an engineer needs to understand at any one time. The complexity of contemporary designs is approaching the intellectual limits of people; powerful design aids coupled with tractable design styles are necessary.

III. J. B. Dennis, MIT Laboratory for Computer Science

Methodologies for Design

We are designing a two-by-two router module which can be used to construct routing networks with N input ports and N output ports using $(N/2)\log_2 N$ modules. A two-by-two router accepts packets in byte-serial format at the two input ports and transmits each packet at the output port specified by one bit of the first byte of the packet. Each port is implemented in hardware as nine data wires (8 that represent one byte and one wire to indicate "last byte"). We are interested in two

designs: one that we can fabricate from standard commercial devices; and one that we could realize in custom LSI (e.g., in NMOS).

Methodologies for constructing self-timed realizations divide into two classes: those in which the data transfer structure and the control structure are separate subsystems; and those in which the data and control are merged. Separation of data and control leads to systems where timing relationships between control and data signals must be satisfied by careful design. Combining data and control allows realization of true speed independent systems except for shared connections to C-elements.

Methodologies for separate data and control structures include: synthesizing the control as an interconnection of standard asynchronous control modules (fork, join, sequence, union, cond, iter, arbiter) [14]; drawing a Petri net for the control and synthesizing hardware by direct translation using Patil's rules or his asynchronous logic array [41]. These methodologies seem useful for realizing asynchronous modules from commercial components, but seem to require many more gates than a direct, hand-designed implementation. Perhaps optimizing techniques may be formulated to reduce the complexity of these designs.

Methodologies that yield true speed independent designs generally employ dual rail coding of data. They make exorbitant use of gates, but connections are very local. This approach seems very attractive for custom LSI (or VLSI) realizations where it is doubtful that practical schemes for controlling the relative timing of signal can be developed.

Session on Hardware Realization Issues

Chairman: C. Molnar

1. **Stephen H. Unger**, Columbia University

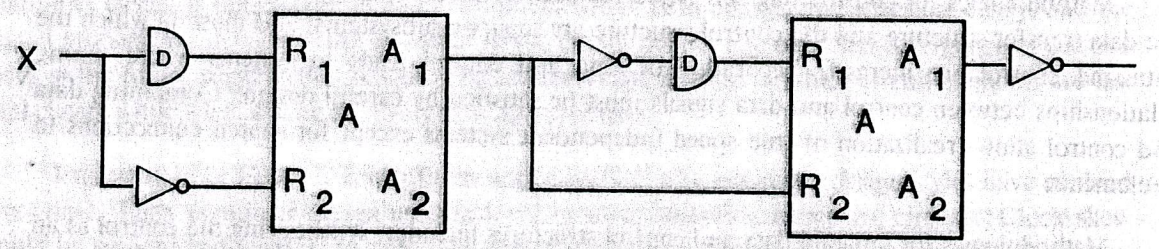
Equivalence of Synthesis Problems for Arbiters, Synchronizers and Inertial Delays

It is widely believed that bounded arbiters, bounded synchronizers and ideal inertial delay elements are not physically realizable. However, there are as yet no generally accepted proofs that this is the case. In this presentation I shall complete a series of arguments showing that all of these elements are physically realizable if and only if any one of them is realizable, that is, either all 3 are realizable or all 3 are unrealizable. (I am inclined to agree with the general consensus that none are realizable.) I will also make a suggestion as to one possible way to configure several non-ideal inertial delays so as to reduce the likelihood of runt pulses occurring.

It follows from previous work [58, 59] that, given ideal inertial delays, both synchronizers and arbiters can be realized. More recently Strom has shown [50] that the circuit below, using pure delays D and ideal arbiters A realizes an ideal inertial delay of magnitude D cascaded with a pure delay of the same magnitude.

Consider now the realization of an ideal bounded arbiter. A flow table for such a device and logic expressions are:

Fig. 2. An Inertial Delay Realized with Ideal Arbiters

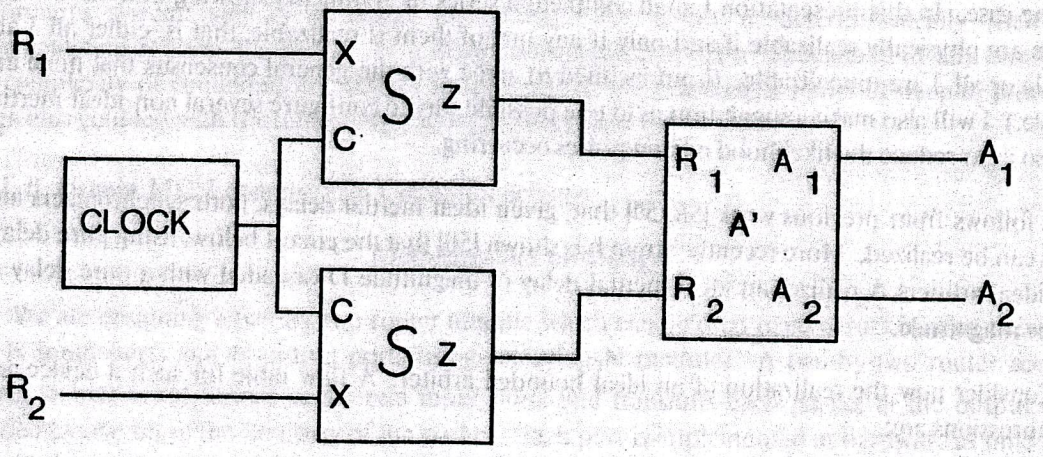


	R_1R_2 = 00	01	11	10	y
1	①, 00	①, 01	①, 01	2, 00	0
2	1, 00	1, 00	②, 10	②, 10	1

$$Y = R_1\overline{R_2} + R_1y$$
$$A_1 = R_1y$$
$$A_2 = R_2\overline{y}$$

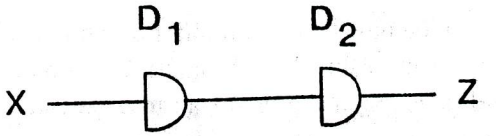
This would work properly *if* we had an ideal inertial delay in the y branch. Without such a device, but with ideal synchronizers S, we could embed a circuit A' described by the above equations in the circuit below, using a *non*-ideal inertial delay (to be described below) in the y-branch. The clock parameters are chosen to conform with the desired resolution time of the arbiter and with the stray delays in the circuit for A'. The latter determines the maximum time that, during a multiple input change, the system could appear to be in state 1-10 during a change from 1-00 to 1-11. This time ε should be small compared to the clock pulse width W_c. There is no problem in physically realizing an inertial delay that will filter out changes of duration $< \varepsilon$ and pass (with delay) changes of duration

Fig. 3. Ideal Arbiter Realized with Synchronizers



$\geq W_c$ [4]. If no changes of duration between these values are presented to it, then no runt pulses will be produced. This constraint is satisfied by the given circuit. Hence if we had ideal synchronizers, we could build ideal arbiters. This completes the argument about the equivalence of the realizability problem for the 3 devices.

As a final point, I would like to present an idea for designing better (although not ideal) inertial delays. If, in the circuit below, $D_2 > D_1$ (both are nonideal inertial delays), then an input that could produce a runt from D_1 would pass through D_2 unchanged.



Thus there is no advantage in the cascade connection. But if $D_1 > D_2$, an x pulse of width D_2 (that might lead D_2 to produce a runt) would *not* pass through D_1. The only runts that could appear at Z would have to be generated in D_1 *and* passed by D_2. But it seems reasonable that, for an inertial delay of magnitude D, any runts would be of duration $\leq D$. If this is so, then *many* (unfortunately not all) runts generated in D_1 would be filtered out by D_2. Thus the cascade circuit could produce fewer runts than a single inertial delay. (We assume D_2 is just a *little* less than D_1.) The matter should be studied further.

II. Alan Hayes, The University of Utah

Self Timed Realizations of Asynchronous State Machines

I have developed a method for realizing asynchronous state machines with inputs and outputs which obey a four-phase request-acknowledge protocol. The method is similar to the stored-state method for realizing synchronous state machines and produces systems which are guaranteed free hazards and races. In addition to simplifying the process of designing asynchronous state machines, the method also allows utilization of available MSI parts.

III. Clement K. C. Leung, MIT Laboratory for Computer Science

Fault Tolerance in Self-Timed Hardware Systems

For this presentation, a self-timed hardware module is one which communicates with its environment via dual-rail protocols. For design and analysis we model hardware failures as random wave trains appearing at the outputs of failed modules. A random wave train may wander arbitrarily in the region bounded by the signal levels representing 0 and 1. Hardware failures in a module can lead to pathological input conditions at its neighbors. These pathological conditions can be detected and masked using redundancy. The implementation of familiar modular redundancy schemes and coding techniques for the adopted hardware and fault model requires solutions to two specific problems:

1. Some pathological input conditions will violate the conventions for packet communication using dual-rail.

2. A failed module may cease to acknowledge inputs and deliver outputs. These failures are said to kill a module. Killing faults can only be dealt with using time-outs. It is thus necessary to introduce a time metric into an otherwise completely self-timed system.

When hardware failures can be modeled as random pulse trains, which oscillate between 0 and 1 but do not reside in the intermediate region for any significant amount of time (relative to logic gate delay), pathological input conditions under dual-rail transmission can be filtered out using a register constructed from ideal C-elements and asynchronous arbiters.

A time-metric is incorporated by introducing the concepts of *performance compatible* modules and *in-phase* operation. Two module copies are performance compatible if, whenever every input packet and acknowledgment is delivered to both of them within a fixed time interval, they will also deliver corresponding acknowledgments and output packets within a fixed time interval. A system supports in-phase operation if every input packet and acknowledgment is delivered to module copies within a fixed time interval. These two concepts apply to modules and systems, respectively. In a system supporting in-phase operation of performance compatible module copies, time-out mechanisms for fault detection and masking can be based on performance incompatibility factors and propagation delays.

IV. T. Chaney and F. Rosenberger, Washington University

Synchronization Flip Flops

These slides of oscilloscope photographs show the responses of several different types of flip-flops to marginal input timing conditions and illustrate some of the various modes of flip-flop metastable behavior that have been found experimentally. I have developed a test method for determining a set of parameters that can be used to define the resolving performance of a flip-flop in a synchronizer design.

Synchronizer failures are currently rare, but as circuits increase in density and speed, the rate at which fixed time synchronizer failures occur could scale as the product of these two factors. Thus, their occurrence may become more serious. This is discussed more fully in [9]

As a further problem in synchronizer design, we have found some TTL and CMOS flip-flops that have undesired modes of synchronizer behavior, such as more than two stable states. Hence they may become permanently hung between logic thresholds.

V. **Mishell J. Stucki**, Washington University, Dept. of Computer Science

I have developed a procedure for synthesizing the circuit equations for self-timed circuits that accept and generate transition signals [52, 53]. The procedure is based on behavioral specifications given in a modified Petri Net format. The circuit equations are derived in a simple, fast, mechanical manner, and special logic elements are not needed.

I have also developed a new scheme for synchronization in clocked systems which decouples the sampling rate from resolution time considerations in systems with non-pausable clocks. The same scheme can be applied in systems with pausable clocks to keep clock-jitter considerations from constraining the basic clock rate. This is discussed more fully in [55].

Finally, in [54] I reported on a data-flow type of implementation for macromodules . In this implementation, the invoking of a data operation is decoupled from the beginning of operation execution. Control signals move through a system invoking operations without waiting for the operations to execute. Once invoked, an operation waits until the data it needs becomes available and then executes.

Session on Design Aids

Chairman: M. Ercegovac

I. **Clement K. C. Leung**, MIT Laboratory for Computer Science

An Approach to the Design and Implementation of Self-Timed Hardware Systems

The design and implementation of hardware systems in VLSI share many problems with the engineering of large software systems. This talk presents a multi-level approach to constructing self-timed hardware systems organized as a packet communication architecture. An architecture description language ADL is proposed as a tool for specifying and refining system design. Hardware implementation is viewed as a translation process from ADL descriptions into interconnections of hardware elements.

ADL has constructs for supporting structural refinement and for specifying the stream processing behavior of modules. In structural refinement a module is decomposed into an interconnection of submodules. In these steps the data structures for passing information among submodules are also designed and specified for each interconnection. The stream processing behavior of modules can be given in several forms: as a recursive definition of a stream processing function; or as the interaction between input, output and state variables (if any). I consider a recursive definition a more "abstract" specification to be further refined into a description of the latter form. The ADL constructs for describing stream processing using state variables are given a rigorous definition through translation into data flow graphs whose operational semantics is in turn given by a firing rule. Non-determinism is incorporated through a monitor construct.

This approach is illustrated with the design of a 2 x 2 router, which is first specified recursively. It is then decomposed into two input modules and two output modules. The behavior of each submodule is described using state variables.

For hardware implementation I propose a set of self-timed hardware elements. Each element interacts with its environment via dual-rail protocols. This set includes dual-rail AND, OR and NOT elements which perform the respective logic functions in dual-rail; a register element; switch and multiplexor elements; and a non-deterministic merge element. Combinational and sequential functions can be implemented with these elements. If an ADL description is refined to the level where every actor in the corresponding data flow graph can be implemented by an available hardware module, its translation into a self-timed hardware system is straightforward.

II. Trevor Mudge, University of Michigan

I have developed a design language with sufficient scope to describe multiprocessing systems and for which any syntactically correct program describes a system with a deadlock-free control structure. The control problem associated with multiprocessing is quite complex, and the opportunities for creating a control structure which can hang-up are great. My language syntactically prevents such structures and hence gives the user a true design tool, rather than just an aid for documenting the principles of operation of a system.

The design language can be used to design digital systems which conform to the following model: the system partitions into a data structure and a hierarchically-organized, asynchronous control structure. Actions in the data structure are assumed to be representable as register transfers. The coordination of these actions is accomplished by the control structure.

In [30, 31] I discuss an implementation procedure based on Dennis' set of asynchronous modules [13].

III. Robert M. Keller, University of Utah

FGL as a Hardware Design Language

FGL is a Lisp-based data-flow-like language which has been used for programming the AMPS multiprocessing system [22]. I have also looked into using FGL for the design of asynchronous modular hardware. Motivations for this attempt include the desire to facilitate human comprehension of logic designs, correctness proofs, refinement, and smooth transitions between design levels.

In FGL, there is no inherent separation of data and control. Instead we have an applicative style of programming, with composition and production as the main combining constructs. The translation of productions, even recursive ones, into an appropriate controlling state machine, as well as the translation of the basic FGL operators into hardware, are steps which are automatable and largely need not concern the high-level designer. Detailed exploration and automation of these translation techniques is a topic which will be pursued in the future.

IV. J. R. Jump, Rice University

A Digital Design Language Suitable for Describing Asynchronous Systems

This project has been concerned with the development of a language that could be used to specify the behavior of a digital system at various levels of detail. The goal was to produce a language that satisfied the following properties:

1. It could represent both synchronous and asynchronous systems in a natural and consistent manner.

2. It could be used to represent the algorithm realized by a system at a level that is independent of its implementation but that would allow the addition of implementation details in a natural way.

3. It should provide syntactic features that encourage the development of structured programs in a top-down manner.

The language that has been developed uses a concept called a *guard* to achieve most of these goals. Each statement can be protected on both sides by guards. These are expressions which regulate the activation and termination of the statements they guard. When control arrives at a statement, that statement is not initiated until its front guard is asserted. Once a statement is initiated, control is not passed to the next statement until the rear guard is asserted. Thus, the sequencing control of a system, either synchronous or asynchronous, can be specified by guard expressions.

Final Words

I. Jack Dennis

I am very pleased with the events of these few days. There has been a fruitful exchange of ideas, and I sense a general feeling that self-timed principles could have an important, if not essential, role in the design of very large-scale integrated circuit devices. What is needed is a good methodology for going from a system concept to LSI masks with certainty that the resulting device will implement exactly the intended function. There are many problems to be solved. Best wishes!

3. Bibliography

The following list contains both the references cited in the preceding summaries as well as many of the important works in the area of self-timed systems.

[1] Altman, S. M., and P. J. Denning, "Decompositions of Control Networks", *Record of the Project MAC Conference on Concurrent Systems and Parallel Computation*, ACM, New York 1970, 81-92.

[2] Armstrong, D. B., A. D. Friedman, and P. R. Menon, "Design of Asynchronous Circuits Assuming Unbounded Gate Delays", *IEEE Trans. on Computers*, Vol. C-18, No. 12 (December 1969), 1110-1120.

[3] Bell, C. G., and J. Grason, "The Register Transfer Module Concept", *Computer Design*, May 1971, 87-94.

[4] Bell, C. G., J. L. Eggert, J. Grason, and P. Williams, "The Description and Use of Register Transfer Modules (RTMs)", *IEEE Trans. on Computers*, Vol. C-21, No. 5 (May 1972).

[5] Bell, G. C., J. Grason, and A. Newell, *Designing Computers and Digital Systems*, Digital Press, Maynard, Ma., 1972.

[6] Bruno, J., and S. M. Altman, "Asynchronous Control Networks", *IEEE Conference Record. Tenth Annual Symposium on Switching and Automata Theory*, 1969, 61-73.

[7] Chaney, T. J., S. M. Ornstein, and W. M. Littlefield, "Beware the Synchronizer", *COMPCON-72, IEEE Computer Society Conference*, San Francisco, Ca., September 1972.

[8] Chaney, T. J., and C. E. Molnar, "Anomalous Behavior of Synchronizer and Arbiter Circuits", *IEEE Trans. on Computers*, Vol. C-22, No. 4 (April 1973), 421-422.

[9] Chaney, T. J., and F. U. Rosenberger, "Characterization and Scaling of MOS Flip Flop Performance in Synchronizer Applications", *Conference on Very Large Scale Integration Architecture, Design, Fabrication*, California Institute of Technology, January 1979.

[10] Clark, W. A., "Macromodular Computer Systems", *AFIPS Conference Proceedings*, Vol. 30 (Spring 1967), 335-336.

[11] Cohen, D., "A Parallel Process Definition and Control System", *AFIPS Conference Proceedings*, Vol 33 (Fall 1968), 1043-1049.

[12] Couranz, G. R., and D. F. Wann, "Theoretical and Experimental Behavior of Synchronizers Operating in the Metastable Region", *IEEE Trans. on Computers*, Vol. C-24 (June 1975), 604-616.

[13] Dennis, J. B., *Computation Structures*, Notes for Subject 6.232, Dept. of Electrical Engineering, M.I.T., Cambridge, Mass., 1970.

[14] Dennis, J. B., "Modular, Asynchronous Control Structures for a High Performance Processor", *Record of the Project MAC Conference on Concurrent Systems and Parallel Computation*, ACM, New York 1970, 55-80.

[15] Dennis, J. B., and S. S. Patil, "Speed Independent Asynchronous Circuits", *Proc. of the Fourth Hawaii International Conference on System Sciences*, 1971, 55-58.

[16] Goldberg, J., and H. S. Stone, "Asynchronous Propagation-Limited Logic", *IEEE Conference Record. Seventh Annual Symposium on Switching and Automata Theory*, 1966, 215-226.

[17] Grasselli, A., "Control Units for Sequencing Complex Asynchronous Operations", *IRE Trans. on Electronic Computers*, Vol. EC-11, No. 4 (August 1962), 483-493.

[18] Jump, J. R., "Asynchronous Control Arrays", *IEEE Trans. on Computers*, Vol. C-23, No. 10 (October 1974), 1020-1029.

[19] Kautz, W. H., "Totally Sequential Switching Circuits", *Switching Theory in Space Technology*, Stanford University Press, Stanford, Ca., 1963, 20-45.

[20] Keller, R. M., and D. F. Wann, *Analysis of Implementation Errors in Digital Computing Systems*, Technical Report 6, Computer Systems Laboratory, Washington University, St. Louis, Mo., March 1968.

[21] Keller, R. M., "Towards a Theory of Universal Speed-Independent Modules", *IEEE Trans. on Computers*, Vol. C-23, No. 1 (January 1974), 21-33.

[22] Keller, R. M., Lindstrom, G., and Patil, S., "A Loosely-Coupled Applicative Multi-Processing System", *AFIPS Proceedings*, NCC, June 1979,

[23] Kimura, I., "Automatic Computer as an Automaton; Some Topics Related to the Logical Circuits of the New Illinois Computer", *Electronics and Communications in Japan*, Vol. 46, No. 11 (November 1963), IEEE, New York 1964, 92-99.

[24] Kimura, I., "Extensions of Asynchronous Circuits and the Delay Problem", *J. of Computer and System Sciences*, Vol. 2, No. 3 (October 1968), 251-287.

[25] Littlefield, W. M., and T. J. Chaney, *The Glitch Phenomenon*, Technical Memorandum 10, Computer Systems Laboratory, Washington University, St. Louis, Mo., December 1966.

[26] Marino, L. R., "The Effect of Asynchronous Inputs on Sequential Network Reliability", *IEEE Trans. on Computers*, Vol. C-26, No. 11 (November 1977), 1082-1090.

[27] McNaughton, R., *Badly Timed Elements and Well Timed Nets*, Technical Report, Moore School of Electrical Engineering, University of Pennsylvania, June 1964.

[28] Miller, R. E, "Speed Independent Switching Circuit Theory", Chapter 10 of *Switching Theory, Vol. II: Sequential Circuits and Machines*, J. Wiley, New York 1965, 192-244.

[29] Misunas, D. P., "Petri Nets and Speed Independent Design", *Comm. of the ACM*, Vol. 16, No. 8 (August 1973), 474-481.

[30] Mudge, T. N., "Specifying a Design Language for Digital Systems", *Proc. of the 13th Annual Allerton Conference on Circuit and System Theory*, October 1975, 905-915.

[31] Mudge, T. N., *A Computer Hardware Design Language for Multiprocessor Systems*, Coordinated Science Laboratory Report R-787, University of Illinois, Sept. 1977.

[32] Muller, D. E., "Asynchronous Logics and Application to Information Processing", *Switching Theory in Space Technology*, Stanford University Press, Stanford, Ca., 1963.

[33] Muller, D. E., "The General Synthesis Problem for Asynchronous Digital Networks", *IEEE Conference Record. Eighth Annual Symposium on Switching and Automata Theory*, 1967, 70-82.

[34] Muller, D. E., and W. S. Bartky, "A Theory of Asynchronous Circuits", *Proc. of an International Symposium on the Theory of Switching, The Annals of the Computation Laboratory of Harvard University*, Vol. 29, Part I. Harvard University Press, Cambridge, Ma., 1959, 204-243.

[35] Ornstein, S. M., J. J. Stucki, and W. A. Clark, "A Functional Description of Macromodules", *AFIPS Conference Proceedings*, Vol. 30 (Spring 1967), 337-355.

[36] Patil, S. S., and J. B. Dennis, "Description and Realization of Digital Systems", *Proc. of the Sixth Annual IEEE Computer Society International Conference*, San Francisco, Ca., September 1972.

[37] Patil, S. S., *Circuit Implementation of Petri Nets*, Computation Structures Group Memo 73, Project MAC, M.I.T., Cambridge, Ma., October 1972.

[38] Patil, S. S., *Synchronizers and Arbiters*, Computation Structures Group Memo 91, Project MAC, M.I.T., Cambridge, Ma., October 1973.

[39] Patil, S. S., *Bounded and Unbounded Delay Synchronizers and Arbiters*, Computation Structures Group Memo 103, Project MAC, M.I.T., Cambridge, Ma., June 1974.

[40] Patil, S. S., "Cellular Arrays for Asynchronous Control", *Proc. of the ACM 7th Annual Workshop on Microprogramming*, 1974.

[41] Patil, S. S., *An Asynchronous Logic Array*, Technical Memorandum 62, Project MAC, M.I.T., Cambridge, Ma., May 1975.

[42] Patil, S. S., "Micro-Control for Parallel Asynchronous Computers", *Proc. of the Euromicro Workshop*, Nice, France, North-Holland Publishing Co., 1975.

[43] Pechoucek, M., "Anomalous Response Times of Input Synchronizers", *IEEE Trans. on Computers*, Vol. C-25, No. 2 (February 1976), 133-139.

[44] Plummer, W. W., "Asynchronous Arbiters", *IEEE Trans. on Computers*, Vol. C-21, No. 1 (January 1972), 37-42.

[45] Robertson, J. E., "Problems in the Physical Realization of Speed Independent Circuits", *Proc. of the Second Symposium on Switching Circuit Theory and Logical Design*, AIEE, October 1961, 106-108.

[46] Seitz, C. L., "Asynchronous Machines Exhibiting Concurrency", *Record of the Project MAC Conference on Concurrent Systems and Parallel Computation*, ACM, New York 1970, 93-106.

[47] Seitz, C. L., *Graph Representations of Logical Machines*, Ph.D Thesis, Dept. of Electrical Engineering, M.I.T., Cambridge, Ma., January 1971.

[48] Seitz, C. L., "System Timing", Chapter 7 in *Introduction to VLSI Systems*, by C. A. Mead and L. A. Conway, Addison-Wesley Press, October 1979.

[49] Seitz, C. L., "Self-Timed VLSI Systems", *Proc. of the Caltech Conference on VLSI*, January 1979,

[50] Strom, B. I., "Proof of the Equivalent Realizability of a Time-Bounded Arbiter and a Runt-Free Inertial Delay," *Proc. of the Sixth Annual Symposium on Computer Architecture*, IEEE, 1979, 178-181.

[51] Stucki, M. J., S. M. Ornstein, and W. A. Clark, "Logical Design of Macromodules", *AFIPS Conference Proceedings*, Vol. 30 (Spring 1967), 357-364.

[52] Stucki, M. J., "An Approach for Synthesizing Transition Logic Circuits", *Proc. of the Eleventh Annual Allerton Conference on Circuit and System Theory*, Univ. of Illinois at Urbana-Champaign, Oct. 1973, 418-427.

[53] Stucki, M. J., "Synthesis of Level Sequential Circuits: Further Development of a Procedure Based on a Petri Net Type of Behavioral Description", *Proc. of the Thirteenth Annual Allerton Conference on Circuit and System Theory*, Univ. of Illinois, Oct. 1975, 896-904.

[54] Stucki, M. J., and M. L. Pepper, "Asynchronous Coordination of Data Operations: A Scheme Exploiting Overlap Potential", *Proceedings of the Thirteenth Annual Allerton Conference on Circuit and System Theory*, Univ. of Illinois, Oct. 1975, 887-895.

[55] Stucki, M. J., and J. R. Cox, Jr. "Synchronization Strategies", *Proc. of Conference on Very Large Scale Integration Architecture, Design and Fabrication*, Pasadena, Ca., January 1979.

[56] Swartout, R. E., "One Method for Designing Speed Independent Logic for a Control", *Proc. of the Second Symposium on Switching Circuit Theory and Logical Design*, AIEE, October 1961, 94-105.

[57] Unger, Stephen H., *Asynchronous Sequential Switching Circuits*, John Wiley and Sons, 1969.

[58] Unger, S. H., "Asynchronous Sequential Switch "Circuits with Unrestricted Input Changes", *IEEE Trans. on Computers*, Dec. 1971, 1437-1444.

[59] Unger, S. H., "Self-Synchronizing Circuits and Non-Fundamental Mode Operation", *IEEE Trans. on Computers*, March 1977, 278-281.

[60] Wann, D. F., C. E. Molnar, T. J. Chaney, and M. Hurtado, "A Fundamental Problem Associated with the Physical Realization of Certain Classes of Petri-Nets", *Proc. of the Conference on Petri Nets and Related Methods*, S. Patil, Ed., to be published. Also Technical Memorandum 215, Computer Systems Laboratory, St. Louis, Mo.

[61] Watson, I., *An Asynchronous Communication Protocol Between Units of the Manchester Data Flow Machine*, Internal Memo, Manchester University, 1979.

[62] Wormald, E. G., "A Note on Synchronizer or Interlock Maloperation", *IEEE Trans. on Computers*, March 1977, pp 317-318.

4. Workshop Participants

William B. Ackerman
MIT Laboratory for Computer Science
545 Main St.
Cambridge, Mass. 02139

Tilak K. M. Agerwala
IBM Watson Research Center
Yorktown Heights, NY 10598

Arvind
MIT Laboratory for Computer Science
545 Main St.
Cambridge, Mass. 02139

A. George Boughton
MIT Laboratory for Computer Science
545 Main St.
Cambridge, Mass. 02139

J. Dean Brock
MIT Laboratory for Computer Science
545 Main St.
Cambridge, Mass. 02139

Randal E. Bryant
MIT Laboratory for Computer Science
545 Main St.
Cambridge, Mass. 02139

Thomas. J. Chaney
Computer Systems Laboratory
Washington University
St. Louis, Mo. 63130

Wesley A. Clark
1572 Massachusetts Ave.
Cambridge, Mass. 02138

Alan Davis
University of Utah
Dept. of Computer Science
Salt Lake City, Utah 84112

Jack B. Dennis
MIT Laboratory for Computer Science
545 Main St.
Cambridge, Mass. 02139

Lloyd I. Dickman
Digital Equipment Corp.
146 Main St.
Maynard, Mass. 01754

Milos Ercegovac
University of California
Dept. of Computer Science
Los Angeles, Calif. 90024

Arif Feridun
MIT Laboratory for Computer Science
545 Main St.
Cambridge, Mass 02139

Alan B. Hayes
University of Utah
Dept. of Computer Science
Salt Lake City, Utah 84112

Anatol Holt
Boston University
Academic Computing Center
Boston, Mass.

J. Robert Jump
Rice University
Dept. of Electrical Engineering
Houston, Texas 77001

Vinod Kathail
MIT Laboratory for Computer Science
545 Main St.
Cambridge, Mass. 02139

Robert M. Keller
University of Utah
Dept. of Computer Science
Salt Lake City, Utah 84112

Clement K. Leung
MIT Laboratory for Computer Science
545 Main St.
Cambridge, Mass. 02139

Leonard R. Marino
San Diego State University
Dept. of Electrical Engineering
San Diego, Calif. 92182

David P. Misunas
Industrial Computer Controls
196 Broadway
Cambridge, Mass. 02139

Charles E. Molnar
Washington University
Computer Systems Laboratory
St. Louis, Mo. 63130

Trevor N. Mudge
University of Michigan
Dept. of Electrical and Computer Engineering
Ann Arbor, Mich. 48109

David Muller
University of Illinois
Dept. of Mathematics
Urbana, Ill. 61801

Keshav Pingali
MIT Laboratory for Computer Science
545 Main St.
Cambridge, Mass. 02139

Martin Rem
Busweg 70
5632 PN Eindhoven
The Netherlands

Frederick Rosenberger
Washington University
Computer Systems Laboratory
St. Louis, Mo. 63130

Charles Seitz
California Institute of Technology
Dept. of Computer Science
Pasadena, Calif. 91125

Mishell J. Stucki
Washington University
Dept. of Computer Science
St. Louis, Mo. 63130

Thye-Lai Tung
MIT Laboratory for Computer Science
545 Main St.
Cambridge, Mass. 02139

Stephen Unger
Columbia University
Dept. of Computer Science
New York, N. Y.