THE UNIVERSITY OF MICHIGAN

COLLEGE OF LITERATURE, SCIENCE, AND THE ARTS
Department of Communication Sciences

Technical Report

ITERATIVE CIRCUIT COMPUTERS:
CHARACTERIZATION AND RESUME OF ADVANTAGES AND DISADVANTAGES

John H. Holland

ORA Project 03105

under contract with:

DEPARTMENT OF THE NAVY
OFFICE OF NAVAL RESEARCH
CONTRACT NO. Nonr-1224(21)
WASHINGTON, D.C.

administered through:

OFFICE OF RESEARCH ADMINISTRATION ANN ARBOR

February 1965

RESEARCH PROGRESS REPORT

Title: "Iterative Circuit Computers: Characterization and Résumé of Advantages and Disadvantages," J. H. Holland, University of Michigan Technical Report 03105-35-T, February 1965; Nonr-1224(21).

Background: The Logic of Computers Group of the Communication Sciences Department of The University of Michigan is investigating the application of logic and mathematics to the design of computing automata. Studies of adaptation in an automaton framework forms a part of this investigation.

Condensed Report Contents: This report begins with a revised characterization of the class of iterative circuit computers (i.c.c.), the various members of this class being determined by the substitution instances of a quintuple (A, A^o, X, f, P) where A is any finitely generated abelian group, A^o is a selected subset thereof, X is any finite set, f and P are finite functions over extensions of X. A brief description of the relation of i.c.c.'s to universal embedding spaces follows. The report concludes with a discussion of some of the advantages and disadvantages of i.c.c. organization for computers constructed of microelectric modules.

For Further Information: The complete report is available in the major Navy technical libraries and can be obtained from the Defense Documentation Center. A few copies are available for distribution by the author.

The discussion which follows is divided into two parts: The first part presents a characterization of the various possible organizations for iterative circuit computers — a streamlined version of the 1960 characterization [4]. The second part comments on some advantages and disadvantages of these organizations for practical machines, particularly micromodular arrays.

The class of iterative circuit computers is the set of all devices (automata) specified by the admissible substitution instances of the quintuple (A, A^O, X, f, P). Each particular quintuple designates a distinct iterative circuit computer organization. Intuitively the five parts of the quintuple determine the following features of the organization:

(i) selection of A determines the underlying geometry of the array, particularly the dimension — thus, among other things A determines whether the array is to be planar, 3-dimensional or higher dimensional;

(ii) selection of A^O determines the standard neighborhood or connection scheme of modules in the array — this A^O determines the number and arrangement of modules directly connected to a given module;

(iii) selection of X determines the storage register capacity of the module;

(iv) selection of f determines the instruction set and related operational characteristics of the module;

(v) selection of P determines the path-building (addressing) capabilities of the modules — see below.

More formally, the admissible substitution instances of each of the five quantities are:

(i) A must be some finitely generated abelian group having a designated set of generators, say $g_0, g_1, \ldots, g_n$, with the restriction

that no constraining relations involve g_o. That is, the group is free on g_o. The positions of modules in the array are indexed by elements of the subgroup A' generated by $g_1, \ldots, g_n$. The time-step is given by the exponent of g_o. Thus $\alpha = g_o^t g_1^{j_1} \ldots g_n^{j_n}$, an element of A, specifies time-step t at the module having coordinates $(j_1, \ldots, j_n)$. By choosing the subgroup A' appropriately the modules can be arranged in a plane, or a torus, or an n-dimensional array, etc. For example, if A' is free on two generators g_1, g_2, an infinite planar array is specified. If the constraining relations

$$g_1^{100} = e$$

$$g_2^{100} = e \quad \text{, where e is the group identity,}$$

are added, a 2-dimensional torus 100 modules in each diameter (10,000 modules total) is specified.

(ii) A^o must be a finite set of elements, $\{a_1, \ldots a_k\}$, belonging to the subgroup A' of A. For a module at arbitrary location, A^o specifies the arrangement of directly connected modules. Thus the modules directly connected to the module indexed by $\alpha = g_o^t g_1^{j_1} \ldots g_n^{j_n}$ will be the modules at $a_i \; \alpha = g_o^t g_1^{j_1+k_{il}} \ldots g_n^{j_n+k_{in}}$, where $a_i = g_1^{k_{il}} \ldots g_n^{k_{in}}$. For example, if there is a module at (j_1, j_2) relative to generators g_1, g_2, and the directly connected modules are to be at coordinates (j_1+1, j_2), (j_1, j_2+1), (j_1-1, j_2), and (j_1, j_2-1), then A^o should be the set $\{g_1, g_2, g_1^{-1}, g_2^{-1}\}$, where g^{-1} is the group inverse of g.

(iii) X can be an arbitrary finite set. The set of internal states of the module is the set $S = X \times Y$ where Y is the cartesian product $\Pi_{i=1}^{k} Y_i$, $Y_i = \Pi_{j=1}^{k} \{a_j \cup \phi\}$ and $a_j \in A^o$. That is, Y is the set of k x k matrices with entry Y_{ij} being a_j or ϕ. The set X corresponds roughly to the possible states of the module's storage register; the set Y consists of the possible gate configurations for the paths — see the transition equations below.

(iv) f can be an arbitrary finite function of the form

$$f: \{S \cup \phi\}^{k} \to S \quad .$$

f deterimines the instruction set, that is, the permissible transitions of the storage register states — see the transition equations.

(v) P can be an arbitrary finite function of the form

$$P: S \to Y \quad .$$

P determines changes in path gating — see the transition equations.

Having chosen (A, A^o, X, f, P), the behavior of the corresponding iterative circuit computer is completely determined by the following state transition schema:

[$S(\alpha)$ will designate the element of S associated with α under the mapping defined recursively by the transition schema. Under interpretation $S(\alpha)$ designates the internal state of the module with space-time coordinates $(t, j_1, \ldots, j_n)$ corresponding to $\alpha = g_o^t \, g_1^{j_1} \ldots g_n^{j_n}$. This convention will also be used for the components of S and, in particular, $Y_{ij}(\alpha)$ will designate the value of element Y_{ij} in the matrix Y associated with α. Note also, that $g_o\alpha = g_o^{t+1} \, g_1^{j_1} \ldots g_n^{j_n}$ designates the module at the same space coordinates as given by α, but at time t+1 rather than t.]

The transition schema for $Y(g_o\alpha)$ determines the path-gating at time t+1 in the corresponding module in terms of the internal state of the module at time t, $S(\alpha)$. Under interpretation, if $Y_{ij}(\alpha) = a_j$ the gate is open so that information can be passed without a time-step delay from the module at $a_j\alpha$ through the module at α to the module at $a_i^{-1}\alpha$; if $Y_{ij}(\alpha) = \phi$ the gate is closed. In other words $Y(\alpha)$ tells how information is to be channeled through the module to its immediate neighbors; the matrices for these neighbors tell how the information is to be sent on from there, etc. (Details of the information transfer are given by the transition equations for $S(g_o\alpha)$).

$$Y_{ij}(g_o\alpha) = Y_{ij}(\alpha) \text{ if } Q_{ij}(\alpha) = 0 \text{ and } P_{ij}(\alpha) = a_j$$

$$= P_{ij}(\alpha) \text{ otherwise}$$

where $P_{ij}(\alpha)$ is the matrix element (i,j) of $P(S(\alpha))$

and $Q_{ij}(\alpha) = \Lambda_{h=1}^{k} q_{jh}(a_j\alpha)$

$$q_{jh}(\beta) = 0 \text{ if } Y_{jh}(\beta) = \phi \text{ and } P_{jh}(\beta) = a_h$$

$$= 1 \text{ if } P_{jh}(\beta) = \phi$$

$$= \Lambda_{\ell=1}^{k} q_{h\ell}(a_h\alpha) \text{ otherwise}$$

where $\Lambda_{x=1}^{k} q_{jx}$ is the conjunction of the q_{jx}.

Under interpretation $P_{ij}(\alpha)$ specifies a proposed gate-setting for time t+1 at the given module. $Q_{ij}(\alpha)$ prohibits any change in the gate-setting, if there are any changes elsewhere in the path leading through that particular gate. This prohibition prevents the following unstable situations:

(i) a cycle of connections without delay (operation of modules belonging to such a cycle would in general be indeterminant).

(ii) an indefinitely long chain of connections without delay (otherwise a possibility in certain interesting infinite arrays).

The transition equations for $X(g_o\alpha)$ are given in terms of a function I: $B \to S$, defined for a subset B of A. Under interpretation I represents input to the computer:

$$X(g_o^N\alpha) = f_X(S'(a_1\alpha, 1), \ldots, S'(a_k\alpha, k))$$

where f_X is the restriction of f to X

and $S'(\beta, i) = S(\beta)$ if $Y_i(g_o\beta) = (\phi, \ldots, \phi)$

and $I(\beta)$ is not defined

$= I(\beta)$ if $Y_i(g_o\beta) = (\phi, \ldots, \phi)$ and $I(\beta) \in S$

$= f(S'(Y_{i1}(\beta) \cdot \beta, 1), \ldots, S'(Y_{ik}(\beta) \cdot \beta, k))$

otherwise

where $\phi\beta = \phi$ and $S'(\phi, j) = \phi$.

Before going on to the practical advantages and disadvantages of such organizations I would like to relate iterative circuit computers to the infinite automaton set down by von Neumann [9]. Von Neumann's cellular automaton is a natural generalization of the McCulloch-Pitts type of automaton (all primitive elements involve a unit delay): it is an infinite 2-dimensional iterative array of an automaton of that type. In an analogous way iterative circuit computers are natural generalizations of the Burks-Wright type of finite automaton (all switches have negligible delay). Kleene proved that any behavior realizable by a finite automaton of the Burks-Wright type (i.e. any regular event) can be realized by a McCulloch-Pitts automaton at the cost of a constant delay of two time-steps [6]. Thus there is a negligible difference behaviorally between these two types of finite automata. This is not true of the corresponding infinite generalizations. There is no way of simulating finite automata in a von Neumann

array in such a way that the corresponding behaviors all occur with some constant change of time scale, $t' = kr + c$; the more complex the finite automaton simulated, the slower the simulation. On the other hand, an arbitrary finite automaton can be simulated in an iterative circuit computer, preserving not only behavioral timing ($k = 1$) but also details of local structural and behavioral relations. (E.g., the simulation can reflect differences corresponding to realization of a switching function in terms of $\{\wedge, \vee, \sim\}$ vs. realization in terms of the stroke function, $\{|\}$.) In fact, it can be shown such simulation is possible in iterated cellular arrays with locally finite information transfer characteristics only if there is provision for the "making" and "breaking" of non-delay paths. For development of these points the reader is referred to [5].

As a potential organization for computers constructed from micro-electronic components, iterative circuit computers (i.c.c.) exhibit two principal characteristics:

(1) the entire processing part of the computer can be made up of identical modules uniformly interconnected,

(2) within limits of overall storage capacity, the resulting computer can execute an arbitrary number of different programs simultaneously.

The first characteristic offers the following advantages:

(i) set-up costs can be spread over a large number of identical units,

(ii) production, inventory and repair can all be centered on a single integrated component,

(iii) short leads and simple inter-unit connection procedures can be used,

(iv) interface between units can easily be kept standard so that improved units (even those involving new instructions) can be added, thus increasing capacity or capability, without shutdown — programs already written will run correctly on the modified device.

The second characteristic offers the following advantages:

(i) high computation rates for calculations which can be executed in parallel fashion (many outstanding problems owe their computational difficulty to the fact that the underlying process is essentially parallel, forcing single sequence machines into a scanning procedure; in such cases parallel computations offer computation rates unobtainable by any feasible decrease in the cycle time of a single sequence machine).

(ii) space-sharing becomes possible, i.e., the computer can be divided into arbitrary independently-operating sub-computers as required (with store protect features, the sub-computers can be prevented from interaction under central control) and reorganization can be effected, upon demand, under program control.

(iii) because all units are identical and because programs can be executed simultaneously, diagnostic procedures can continually sample modules, with a negligible loss in efficiency — if faulty or failing modules are located, it is possible to "program around" them until replacement.

There are several important disadvantages to i.c.c. organization. Perhaps the most serious disadvantage, to the present time, has been the number of active elements required in each module: several hundred for reasonable flexibility and ease of operation. Closely related, is the low average use factor for some elements in the module. For example, each module usually will have arithmetic capability; yet at any given time only a small fraction of the modules will be using this capability. (It is worth noting that reduced use factors are tolerated even in contemporary single sequence machines as a matter of convenience: compiled programs for simulation of highly parallel systems can be 3 or 4 times slower than "hand-crafted" programs, resulting in a lowered effective use factor

for the arithmetic unit.) The importance of these disadvantages decreases as the average cost of a module drops. If the cost of production is largely set-up cost, it may be possible to produce complicated modules for what it presently costs to produce and assemble a few transistors. Should this happen, average use factor for individual elements is no longer a reasonable measure of overall machine efficiency.

Other potential disadvantages center on problems of internal access and broader problems of programming languages and programming convenience. Some aspects of these problems have been investigated (see [2], [7], and [8]), but they remain largely unexplored. The internal accessing problem stems from the procedure used to locate operands — an operand is accessed by "building a path" (opening a sequence of gates) to it. Two difficulties arise. One depends on the number of operands accessible to any module via a reasonable number of path-building operations. The other concerns path interference (crossover, multiple access, etc.). The first of these is less a problem than it seems at first sight. Assume that no more than 10 gates can be opened, for a given path extension, in one machine cycle (i.e. each path can be extended through at most 10 modules in one machine cycle). Still, in two machine cycles, any of 400 modules can be accessed from any given module in a 2-dimensional machine where each module has four nearest neighbors. The problem of path interference is more serious; however there are at least two ways of alleviating this problem: higher dimensionality [8] or path-building via trunk lines [3]. Both can be used together and both also further reduce the first problem.

Programming convenience and the design of programming languages for an i.c.c. are extensive problems not likely to be solved in a fell swoop. One or two comments may show the scope of the problem and where hope lies. For some kinds

of problems, programming is actually simpler than for conventional single sequence machines. The solution of partial differential equations in 3-space and time (the weather problem, etc.), by conventional methods on a single-sequence computer, requires that about 90% of the program be given over to scanning logic, boundary manipulation, etc. These considerations are eliminated in an appropriate i.c.c., since the basic grid-point sub-routine can simply be copied throughout — all grid-points are then updated simultaneously. In other contexts, because programs can be executed simultaneously, we face the usual difficulties of parallelism: asynchrony and priority. Certain natural operations, such as association along paths, complicate the problem. Techniques used in the design of asynchronous circuits are useful here. For example, sub-programs can be written so that, when their part of the calculation is ready the corresponding output registers are assigned ready status; sub-programs go to execution status when all operands required are in ready status. An example of an i.c.c. compiler, along somewhat different lines, appears in [1].

There is a great class of highly-parallel problems intrinsically beyond the capability of any presently feasible single sequence machine: the unrestricted weather problem, magnetohydrodynamic problems, command and control problems, simulations of biological and ecological systems, and so on. Iterative circuit computers offer the possibility of treating these problems in parallel fashion — thus making them accessible to computation. On balance I would say that the feasibility of an i.c.c. rests upon resolution of the first-mentioned disadvantage: design and production of a module with several hundred active elements at a relatively low cost. The other problems will almost certainly be resolved sufficiently to permit useful computation if this can be accomplished.

ACKNOWLEDGMENT

The work described here was in part supported by the Office of Naval Research under contract Nonr 1224(21).

REFERENCES

1. Comfort, W. T., "Highly Parallel Machines," Workshop on Computer Organization, ed. Barnum, A. A., and Knapp, Spartan, 1963.

2. Comfort, W. T., "A Modified Holland Machine," Proc. 1963 Fall Joint Computer Conference IEEE, 1963.

3. Gonzalez, R., "A Multi-Layer Iterative Circuit Computer," Transactions on Electronic Computers EC-12, 5, 781-790, 1963.

4. Holland, J. H., "Iterative Circuit Computers," Proc. Western Joint Computer Conference, 259-265, IEEE, 1960.

5. Holland, J. H., "Universal Spaces: A Basis for Studies of Adaptation," to appear in Automata Theory, ed. Caianiello, E. R., Academic Press, 1965.

6. Kleene, S. C., "Representation of Events in Nerve Nets and Finite Automata," Automata Studies, 3-41, Princeton, 1956.

7. Newell, A., "On Programming a Highly Parallel Machine to be an Intelligent Technician," Proc. Western Joint Computer Conference, 267-282, IEEE, 1960.

8. Squire, J. S., and Palais, S. M., "Programming and Design Considerations of a Highly Parallel Computer," Proc. Spring Joint Computer Conference, IEEE, 1964.

9. von Neumann, J., "The Theory of Automata: Construction, Reproduction, Homogeneity, unpub. manuscript.

DISTRIBUTION LIST

(One copy unless otherwise noted)

Technical Library
Director Defense Res. & Eng.
Room 3C-128, The Pentagon
Washington, D.C. 20301

Defense Documentation Center 20
Cameron Station
Alexandria, Virginia 22314

Chief of Naval Research 2
Department of the Navy
Washington 25, D.C.
Attn: Code 437, Information
Systems Branch

Director, Naval Research Laboratory 6
Technical Information Officer
Washington 25, D.C.
Attention: Code 2000

Commanding Officer 10
Office of Naval Research
Navy 100, Fleet Post Office Box 39
New York, New York 09599

Commanding Officer
ONR Branch Office
207 West 24th Street
New York 11, New York

Office of Naval Research Branch
Office
495 Summer Street
Boston, Massachusetts 02110

Naval Ordnance Laboratory
White Oaks, Silver Spring 19
Maryland
Attn: Technical Library

David Taylor Model Basin
Washington,D.C. 20007
Attn: Code 042, Technical Library

Naval Electronics Laboratory
San Diego 52, California
Attn: Technical Library

Dr. Daniel Alpert, Director
Coordinated Science Laboratory
University of Illinois
Urbana, Illinois

Air Force Cambridge Research Labs
Laurence C. Hanscom Field
Bedford, Massachusetts
Attn: Research Library, CRMXL R

U. S. Naval Weapons Laboratory
Dahlgren, Virginia 22448
Attn: G. H. Gleissner, Code K4
Asst. Dir. for Computation

National Bureau of Standards
Data Processing Systems Division
Room 239, Building 10
Washington 25, D.C.
Attn: A. K. Smilow

George C. Francis
Computing Laboratory, BRL
Aberdeen Proving Ground, Maryland

Office of Naval Research
Branch Office Chicago
230 N. Michigan Avenue
Chicago, Illinois 60601

Commanding Officer
ONR Branch Office
1030 E. Green Street
Pasadena, California

Commanding Officer
ONR Branch Office
1000 Geary Street
San Francisco 9, California

DISTRIBUTION LIST (Concluded)

The University of Michigan
Department of Philosophy
Attn: Professor A. W. Burks

National Physical Laboratory
Teddington, Middlesex, England
Attn: Dr. A. M. Uttley, Supt.
Autonomics Division

Commanding Officer
Harry Diamond Laboratories
Washington, D.C. 20438
Attn: Library

Commanding Officer and Director
U. S. Naval Training Device Center
Port Washington
Long Island, New York
Attn: Technical Library

Department of the Army
Office of the Chief of Research
and Development
Pentagon, Room 3D442
Washington 25, D.C.
Attn: Mr. L. H. Geiger

National Security Agency
Fort George G. Meade, Maryland
Attn: Librarian, C-332

Lincoln Laboratory
Massachusetts Institute of Technology
Lexington 73, Massachusetts
Attn: Library

Office of Naval Research
Washington 25, D.C.
Attn: Code 432

Kenneth Krohn
6001 Dunham Springs Road
Nashville, Tennessee

Mr. Laurence J. Fogel
General Dynamics/Astronautics
Division of General Dynamics Corp.
San Diego, California