New Illinois Computer*

D. E. Muller**

I. History and Introduction

In 1957 a report (1) describing the general plan for the design of a very high speed computer was completed by members of the Digital Computer Laboratory at the University of Illinois. This report was the culmination of a study program which had been begun in 1956 and which had as its purpose the investigation of the feasibility of constructing a computer which was about one hundred times as fast as the computers which were then in use.

Already as early as 1954, two years after the completion of the presently operating computer Illiac at the University of Illinois, the desirability of such an investigation was apparent because the possibility of using high speed transistor circuitry enabled one to construct much faster basic logical elements—flipflops, AND, OR, and NOT elements—than had heretofore been used. Furthermore, many important scientific problems are of such a nature that they require excessive amounts of numerical calculation, and for this reason could not easily be solved, if at all, with existing computers. Therefore, it was proposed (1) that a computer be built for use in scientific research which would embody circuit and logical design techniques to give it as high a calculation speed as possible.

Naturally, there were certain difficulties inherent in such a plan. Assuming that an extremely fast arithmetic unit could be constructed by combining the best circuit and logical design methods, it was not possible to make nearly as great an improvement in the speed of the memory. Hence, for certain types of problems the computer would be "memory limited" and the increased arithmetic speed would fail to yield a corresponding improvement in the overall speed of calculation.

Two methods were to be used to overcome the memory limitation. First, it was proposed that a heirarchy of memories be used rather than a single memory. The smallest memory would consist of flipflops or of some similarly fast, but possibly expensive storage units. This memory could be used for the temporary storage of instructions, addressi nformation, and arithmetic results. The progressively larger but slower memories would hold information which was increasingly further removed from the portion of the calculation being currently performed. Thus, it was hoped that the actual time required for memory use would be nearer to that required by the high speed memory than to the time which would be required if all of the computer's storage were of the slower type.

The second method for counteracting the effect of the memory limitation was to be the overlapping of time required for memory use with the time required by the arithmetic unit for numerical calculation. Two interlocked controls would be necessary so that the arithmetic unit and other portions of the computer could function concurrently. Thus, while the arithmetic unit was performing some calculation with a given set of numbers, the computer could proceed with the construction of addresses for operands which would be required in future calculations, and these operands could then be obtained from the store. The control for supervising the nonarithmetic portions of instructions

---

* An invited talk presented at the second annual meeting held in Tokyo November 16, 1961.

** Visiting Professor at the University of Tokyo on leave from the Digital Computer Laboratory. University of Illinois, U.S.A.

was called "advanced control" because it was to be capable of advancing several instructions ahead of the control which supervised the functioning of the arithmetic unit. This latter control was called "delayed control" and the two controls are commonly abbreviated A.C. and D.C. In the case of those problems which were dominated by arithmetic calculations, it was thought that the bulk of memory use could thus be performed during the time that lengthy arithmetic operations were taking place and therefore the effect of having a relatively slower memory would be correspondingly reduced.

In 1958 the preliminary work on the design and construction of the new Illinois computer was begun. Additional work was done on the organization of the computer and on the basic circuits which were to be used. The number of people involved in this effort was so large that it is impossible for me to mention here the names of all the people who made important contributions in the several areas of work. However, I should point out that the computer project as a whole has been under the direction of J. E. Rob ertson. Work on the organization and logical design of the control has been directly super vised by D. B. Gillies.

II. The Main Arithmetic Unit

Figure 1 shows a block diagram of the arit hmetic unit of the new Illinois computer. Blocks labelled A, S, Q, R and M represent registers for numbers. Associated with each register is a gate which allows passage of information into the register. When the gate is closed information remains stored in the register and may be used by other sections of the arithmetic unit, but when the gate is opened the contents of the register are, in general, changed and the information present on the lines going into the register replaces whatever was previously in the register.

This system contrasts with that used in many computers in which a given register may have several gates, the choice of gate determining the

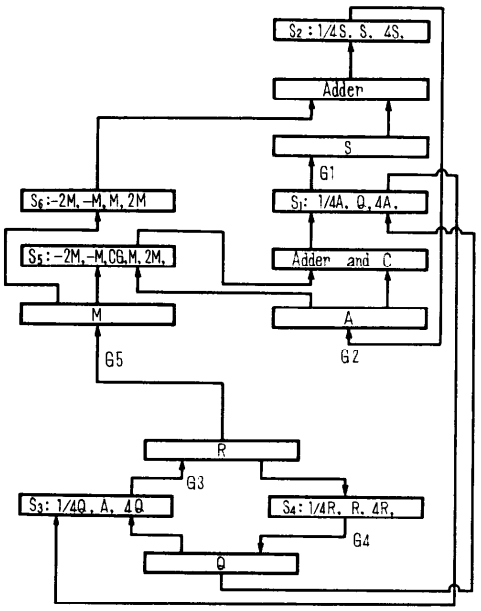


Fig. 1. Arithmetic unit

source of the information passing into the register. In the present system, on the other hand, the source of information is determined by another unit, called a selector. A selector unit is usually associated with each gate, but it also may be used separately, and always requires separate control signals for its operation. Of the many inputs to a selector only one is allowed to pass to the output, the choice depending upon which control signal is applied to the selector.

Use of gates and selectors, rather than simply gates, is occasioned by certain practical considerations relating to transistor circuitry used in the flipflops and other elements which form the registers. Each flipflop requires the use of a gate signal g and an input signal x. If $g=1$ then x is gated into the flipflop, while if $g=0$ the flipflop retains its previous digit. This circuitry will not be described in detail here, but it is interesting to note that the system requires the use of a source (selector) and destination (gate) signal for each transfer of information. It differs slightly from the usual source-destination scheme,

however, in that the selector signal must be present during the entire period of gating, since if no selector signal is supplied the effect will be the same as if the signal zero were applied to all flipflops in the register. Thus, whenever possible, the selector signals are set, and allowed to remain fixed, while a number of gating operations is performed. For example, during a left or right shift, which may take place during addition, multiplication, division, or normali zation, it is possible, to set the selectors so that the shifting occurs in the proper direction. The only signals that then need be changed while the shift is taking place are those which are applied to the gates.

The principal type of number representation used within the computer is a floating point representation which may be thought of as binary, but which is perhaps most accurately described as base 4. In this scheme, 45 of the 52 binary digits in a word are allotted to the fractional part of the number, while the remaining 7 form the exponent. A twos complement system is used, with the first binary digit of the fractional part giving the sign and the remaining 44 binary digits (or 22 base 4 digits) completing the fraction. The exponent determines the power of 4 by which the fraction should be multiplied.

A double length accumulator, consisting of 89 binary digits for holding the fractional parts of numbers, is used in the arithmetic unit. The registers shown in figure 1 forming the double length accumulator are A, S, Q and R. A and S together act as a shifting register for the most significant 45 bits and Q and R as a shifting register for the 44 least significant bits. During addition, the fractional part of the augend is held in double precision in this register, while its 7 bit exponent is held in a register in the exponent arithmetic unit. A single precision floating point number acts as the addend. It may be brought into the M register either from the core store or from one of the temporary storage registers. After addition has taken place the sum appears as a double precision number in the accumulator.

In order to carry out addition with the rather small number of registers in the arithmetic unit, it is necessary to consider several separate processes depending upon the relative magnitudes of the exponents of the two numbers being added. In all, five cases occur, each with a somewhat different addition sequence. Hence, the addition operation requires more control complexity than would be needed if only single precision addition were provided. However, by using double precision accumulation, the roundoff error may be greatly reduced when a series of numbers, possibly of varying sign, are added together. Also, double precision accumulation of products is greatly simplified as a result of this feature, and complete double precision floating arithmetic may even be programmed with relative ease.

Prior to multiplication the multiplier is held in the accumulator. In this case, just the rounded most significant 45 bits are used as the multiplier. During multiplication this multiplier is held in Q, R and is shifted to the right as the product displaces it. A decoding of the digits of the multiplier takes place at the least significant end of Q and R so that they may be reinterpreted as having weights -1, 0, $+1$ and $+2$. Each repetitive step of multiplication then consists of shifting the contents of A, Q to S, R or of shifting the contents of S, R to A, Q and a total of two binary places to the right. During this shift the most significant 45 digits are passed through one of the two adders (the other being used on alternate steps) and depending on the reinterpreted multiplier digit either: (i) the multiplicand is subtracted from the partial product or (ii) nothing is added to the partial product as it passes through the adder or (iii) the multiplicand is added to the partial product or (iv) twice the multiplicand is added to the partial product. The input to the adder at the output of the M selector is made equal to the appropriate multiple of the multiplicand by the proper setting of the M selector.

At the conclusion of the multiplication process the double precision product appears in the

accumulator. However, during both addition and multiplication, actual addition is only partially performed, in the sense that the carrys in the addition process between base 4 digits, instead of being assimilated into the sum, are preserved in special 22 bit registers forming special parts of A and S. These digits must ofcourse, be supplied as further inputs to the adders. In order to speed addition the carrys are allowed to remain unassimilated as long as possible, so assimilation is only performed during division and when a number is brought out of the accumulator.

Normalization of the contents of the accumulator is another process which need not accompany every arithmetic operation. Thus, after addition and multiplication no normalization occurs, but if desired, the contents of the accumulator may be normalized at the time they are stored. In this instruction, roundoff is also performed, and the single precision representation is obtained which is closest to the original contents of the double precision accumulator. After normalization, the fractional part of the number in the accumulator lies in the range; $-1 \leqslant f < -1/4$, $f=0$, or $+1/4 \leqslant f < +1$. The roundoff process may produce both exceptional cases $f=-1/4$ and $f=+1$ lying outside of the given range. When $f=-1/4$ nothing further is done and the number is still regarded as properly normalized, but in the latter case, since $f=+1$ cannot be represented as a fraction in the memory, the fraction $+1/4$ is stored instead, and the exponent which is stored is made one greater than the exponent which resulted from the normalization process.

Various special arithmetic instructions are provided in the instruction code. These include:

1. Inverse divide (as well as conventional division) in which the divisor is in the accumulator and the dividend is brought from the memory.
2. Exchange accumulator and memory.
3. Double accumulator (most significant part only). This is regarded as a logical instruction and does not involve changing the exponent or setting an overflow flipflop even if the contents of the accumulator overflow during the doubling process.
4. Various types of store instructions. Both normalized and unnormalized store instructions are possible. Also provided is a "fixed point" store instruction in which the number, instead of being normalized, is first converted to one in which the exponent has the value zero (or possibly some other chosen value) before storing takes place. One of the most useful instructions for double precision work and for programs in which the roundoff error must be estimated, is the "store clear" instruction. The effect of this instruction upon the memory is the same as that of the normalized, rounded store instruction, but the accumulator is left with a residue which is equal to the difference between its original double precision contents and the quantity stored.
5. A set of logical instructions, all acting upon the contents of the accumulator.
6. A variety of instructions affecting just the exponent. These include add, subtract, clear add, clear subtract, and store. A complete list of arithmetic instructions is given in a recent report (4). Most of the specialized features of this part of the instruction code result from the properties of the high speed storage which is used in conjunction with the arithmetic unit, and from the form of address construction which is used.

III. Order Structure

In table 1 there is described a ten word "flow gating" memory which is used to reduce the number of accesses to the main core store. The first eight words of this small, but very fast, memory are directly addressable, and the last two words are used as instruction registers. In favorable cases, it is possible to program a

loop of instructions so that all the instructions required by the loop will fit into the two instruction registers. Then, during the execution of the loop it is not necessary to bring instructions from the core memory. For this reason and in order to save instruction accesses in general, the order structure of the Illinois computer is made as compact as possible. Each order consists of a 13 bit (1/4 word) instruction part, which may or may not be followed by a 13 bit address. Thus, there are two types of instructions, "short" or 13 bit instructions, and "long" or 26 bit instructions, consisting of two 13 bit control groups, the second control group being used during the address construction process. The 13 bit control groups follow each other from left or right in a word and the words of instruction in successive locations in the memory unless a jump instruction occurs. Therefore, a long instruction may be split between two words, but its operation is in no way changed when this happens.

A 13 bit instruction consists of three fields, called F, B, and C which consist of 7, 4, and 2 bits respectively. The type of operation to be performed is determined by the 7 bit field F, while the other fields determine the way in which the address is constructed. The 4 bit field B usually refers to one of 16 index registers of 13 bits each which are formed from the four flow gating registers F4 through F7. If C=0 or 1, the contents of this index register are taken as the address of the instruction and hence the instruction is of the short, or 13 bit type. By making C=1 the corresponding index register is automatically advanced by 1 after the completion of the instruction. The advantage of this type of address construction is twofold. First, some storage space in the high speed flow gating memory is saved because the instructions used are all "short" instructions not requiring separate addresses, and although more index registers are required, there is a net saving in high speed storage, and hence a greater likelihood that a loop of instructions can be stored completely within the high speed memory and executed without repeated references to the core memory for instructions. Second, time is also saved during the construction of addresses, since within a loop the only addition required is that occurring when the the address is advanced.

In order to perform multiple modification of addresses, one may make use of an "add to next" or "subtract from next" instruction which affects the way in which the address of the next order is formed. In fact, a large set of arithmetic and logical operations may be carried out by the address arithmetic unit upon 13 bit operands. Furthermore, all such operations take place without interfering with the concurrent functioning of the arithmetic unit.

IV. Control System

Operation of delayed control, that portion of the control which supplies the gate and selector signals to the main arithmetic unit, may be thought of as consisting of a sequence of control steps. During each control step several gate or selector settings may take place concurrently, and the following control step may depend upon the presence of various control signals. But in general more than one control sequence will not be going on within delayed control at a given time. This feature of delayed control has permitted the design of delayed control by means of flow charts indicating (i) which gates and selectors are set at any given control step and (ii) which control steps may follow the given step as well as (iii) what conditions determine the choice of the next control step.

Such flow charts make more precise the usual verbal description or outline used to express control sequencing. However, they have another use as well, for rules have been developed for carrying out the replacement of the symbols of a flow chart with suitable logical blocks which combine to form a circuit to carry out the actions indicated by the flow chart.

Flow charts describing control sequences must be written in accordance with certain rules if they are to be used directly for logical design purposes (5b, c). A distinction is made between four types of operations which may be performed in any given control step. These are: (i) operation of gates, (ii) setting of selectors which feed gates directly, such as those appearing at the adder outputs, (iii) setting of selectors which feed logical networks such as those connected to the output of the M register which in turn feed the adders, and (iv) setting of flipflops which are used for control purposes at a later point in the sequence.

In the case of gates, two end signals are provided, one to indicate that the gate was turned on, and a second to indicate that the gate was turned off. These two signals appear on a single line and are opposite in direction, that is, one is represented by the signal transition 0 to 1 and the other by the signal transition 1 to 0. In the case of selectors and control flipflops, two end signals also occur. The first indicates that the selector was set, or that the information was gated into the control flipflop; the second indicates that the logic from which the information flowing into the selector or control flipflop was derived, has been cleared.

A control flipflop is designed so as to be capable of giving a definite indication that its setting is complete, and also a definite indication that its inputs have been cleared. There is, however, no similar check made upon the logic which occurs at the output of such a control flipflop. Hence, it is necessary to insert sufficient delay in the logic of the control so that all combinational circuitry which follows such a control flipflop has sufficient time to reach equilibrium before the outputs from such circuits are used for later control purposes.

Similarly, end signals which indicate the conditions of gates and flipflops are not taken from all the register positions to which gate and selector signals are supplied. Instead, a suitable delay is allowed for the transmission of such signals to the registers and the signal, thus delayed if necessary, is used as the end signal to initiate the next operation.

Except for the introduction of such bypasses in which the operation of some portion of the arithmetic unit is not directly checked, there is speed independent operation within the control in the sense that its correct functioning does not depend upon the relative speeds of the logical elements from which it is formed. Thus, a central section of the control forms a speed independent core. This central section supplies signals to other sections of the computer which operate with delays lying within prescribed limits.

Speed independent design of the central section of the control is achieved, in the case of delayed control, by following rigid rules for the replacement of sections of the flow chart by logical circuitry. These rules were developed heuristically, but were tested rigorously, using the present computer Illiac, to make certain the resulting circuits carried out the sequencing described by the flow charts, and that this sequencing will occur regardless of what relative speeds are assumed for the various logical elements.

In the case of advanced contorol, it is no longer possible to describe the control operation in terms of a sequence of control steps. Instead, one must think of the operation of advanced control as being made up of a number of concurrently acting sequences, which are carried out by semi-autonomous subcontrols to perform its various functions. Thus, separately acting sections of advanced control regulate such things as access to the core memories, acceptance of information from the OUT register F0, index arithmetic and address construction, and various other operations having to do with the decoding and sequencing of instructions.

As advanced control proceeds, it processes one instruction after another. Instructions not

requiring the use of the arithmetic unit may be completely executed by advanced control, and when this is possible advanced control passes on to the next instruction. In other cases advanced control starts the execution of the instruction and then passes the residue on to its subcontrols and to arithmetic control. Advanced control, however, never proceeds with any operations which are conditional upon calculations which have yet to be made and which might have to be undone at a later time depending upon the outcome of such calculations. Thus, it stops upon encountering a jump which is conditional upon the outcome of an arithmetic calculation rather than proceeding with a possibly unneccessary calculation or use of the memory. Advanced control also stops if it requires information which has not been supplied, or if some register or other unit which it must use is occupied with another calculation. It proceeds therefore, as far as possible within the limitations of the equipment which it controls and equipment adjacent to that which it controls.

Yet a third control unit in the Illinois computer is called "interplay". This control deals with the transfer of information between the core memories and drums, magnetic tape units, and input-output devices. Such transfers take place in blocks of 256 words each and proceed while other instructions are being executed. Also, because of the synchronous nature of these devices, their use of the core memory, though infrequent, cannot be postponed. For this reason the system of priorities is such that interplay is granted core memory access whenever this access is necessary for the correct functioning of the auxiliary memories and the input-output equipment. During the transfer by interplay to the core memory of a block of words, these words must not be used by advanced control, since one cannot be certain whether or not the transfer has taken place. Similarly, when a block is transferred from the core memory, these words must not be modified by information coming from other units such as the flow gating memory, since such modification could take place either before or after the transfer. Thus, to preserve speed independence, the section of the core memory involved in the block transfer is "locked out" so that access, other than that concerned with the block transfer, cannot take place within the section.

V. Circuitry and Physical Construction

Although it is not the purpose of this description to consider details of circuit design, it is perhaps worth pointing out that certain special problems are encountered when circuits are to be designed so that they may be used in a speed independent control.

As is the case in synchronous computers as well, the principal logical delay in this computer occurs in signal paths requiring a large amount of amplification. Signals from arithmetic control must be amplified and transmitted to all the digits in the arithmetic registers. Large amplification is required because these signals must be sent through relatively long distances and must fan out to a large number of points. Effort in circuit design was therefore concentrated on the design of drivers for cables and drivers for gates, so as to make these critical elements as fast as possible. Since the distances over which the control signals are transmitted are comparable to the wavelengths of these signals, there is difficulty in making certain that the signals remain faithful and that spurious oscillations and noise are not introduced.

The physical layout of the main section of the computer is shown in figure 2. When looked at from above it has the general shape of the letter T. In the lower six feet of the T is found the arithmetic unit and its associated control. The arithmetic unit which occupies the body of the T is folded so that the first and last digits lie next to the bar of the T which contains delayed control. Those sections of delayed con-

trol which are considered to be the most used during computation are placed nearest to the arithmetic unit, while those having lower duty cycle are placed farther away in the bar of the T. Directly above delayed control and the arithmetic unit, is placed advanced control and the flow gating memory, and in a separate rack next to the main section is placed the core store and the interplay control.

Present plans call for the completion of an operational computer sometime early next year. Hlowever, the completion of advanced control will require additional time and at present the computer will be tested with a simplified instruction sequencing control. Work on advanced control will be carried out during 1962.

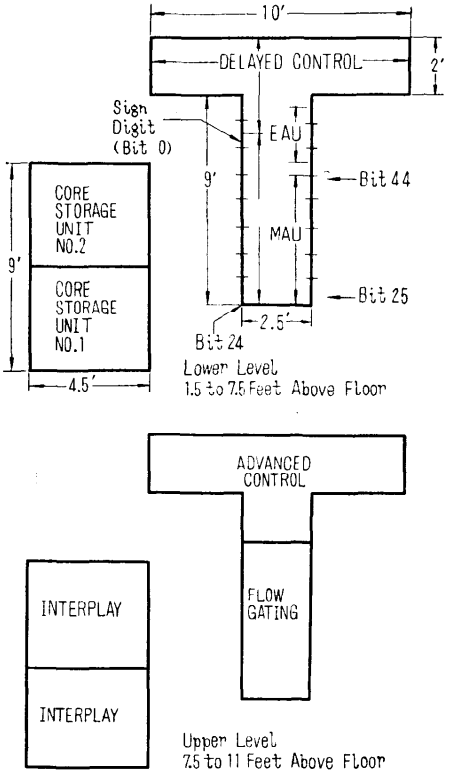
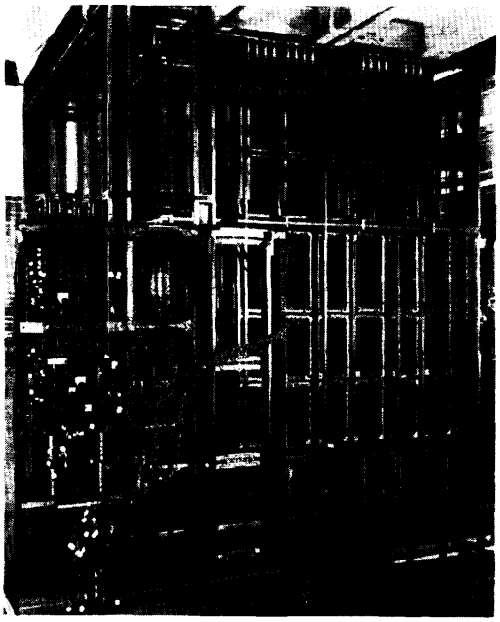


Fig. 2. Layout of major units



Main frame of the new Illinois computer as it appeared during construction in the summer of 1961.

Secondary frame of the new Illinois computer as it appeared during construction in the summer of 1961.

Table 1. Characteristics of the Illinois Computer

1. General Properties

a. Transitorized.
b. Directly coupled.
c. Asynchronous.
d. Parallel.
e. Binary (52 bit word).
f. Single and zero address instruction code.
g. Floating point arithmetic.

2. Speeds (estimated)

a. Multiplication: 7 μsec.
b. Addition: 2 μsec.
c. Division: 16 μsec.
d. Core memory interval: 1 to 2 μsec.

3. Outstanding Features

a. Concurrent arithmetic, memory use, and address processing.
b. Base four "carry save" arithmetic.
c. Double precision accumulation of sums and products.
d. Speed independent central control.
e. Use of rapid access registers for the temporary storage of a loop of instructions.

4. Storage

a. Ten Word "flow gating" memory consisting of flipflops with about 0.2 μsec. access time.

These ten words are allocated as follows:

1) Two buffer registers between the main core memory and the arithmetic unit. These registers, called OUT (F0) and IN (F1), are respectively used for results passing from the arithmetic unit to core storage, and for operands passing from core storage to the arithmetic unit.
2) Two registers (F 2 and F 3) for temporary storage of arithmetic results.
3) Four registers (F 4 through F 7), each of which may be used either as temporary storage or as four index registers.
4) Two registers (F 8 and F 9) for holding instructions which are currently being obeyed.

b. Two word arrangement core memories of 2^{12} words each. One memory is referred to by even addresses and the other by odd addresses. Access time per memory is about 0.5 μsec., and cycle time per memory is about 2.0 μsec.
c. Two drums of 2^{15} word each with a word time of 7.8 μsec., and a period of 17 msec. Block transfers of 256 words are used between drums and cores.
d. Four magnetic tape units with provision for adding more.

Bibliography

1. "On the Design of a Very High-Speed Computer", D.B. Gillies, R.E. Meagher, D.E. Muller, R.W. McKay, J.P. Nash, J.E. Robertson, and A.H. Taub. Digital Computer Laboratory Report No. 80, October 1957, revised in April 1958.
2. "Design of the Core Storage Unit", S.R. Ray. Digital Computer Laboratory Report No. 91, August 1959.
3. "Final Report-Flow Gating", Henry Guckel, Toshiro Kunihiro, Ronald K. Crow. Digital Computer Laboratory Report No. 106, March 1961.
4. "The Design of a Very High Speed Scientific Computer", D.B. Gillies. Digital Computer Laboratory File No. 376, June 1961.
5. Proceedings of the A.I.E.E., Second Annual Symposium on Switching Circuit Theory and Logical Design, Detroit, Michigan, October 15-20, 1961.
   a. "Introduction to Speed Independent Circuit Theory", R.E. Miller.
   b. "A Flow Chart Notation for The Description of a Speed Independent Control," D.B. Gillies.
   c. "One Method for Designing Speed Independent Logic for a Control", R.E. Swartwout.
   d. "Problems in the Physical Realization of Speed Independent Circuits", J.E. Robertson.