ILLIAC II—A Short Description and Annotated Bibliography

H. C. BREARLEY, JR., MEMBER, IEEE

Abstract—**ILLIAC II is a general purpose computer built at the University of Illinois, Urbana. It contains about 55 000 transistors and has a floating multiply time of 6.3 μs. A number of features are provided to increase the speed of operation. There are three controls in largely concurrent operation. The control circuits are largely asynchronous and speed independent. The floating point arithmetic unit contains two adders and utilizes redundant number representation and separate carry storage. The memory hierarchy has members ranging from the 10-word, 0.2-μs. flow gating memory to magnetic tapes. Order fetches from the main 1.8 -μs. core memory are minimized by packing two to four orders per word, and by holding two words of orders in the flow gating memory for execution of short loops. The bibliography lists 40 papers related to the design of this computer.**

ILLIAC II is a large, high-speed, general-purpose computer built by the Digital Computer Laboratory,[1] University of Illinois, Urbana. Comprehensive plans for its construction were given in a widely quoted 1957 report [38]. No similarly comprehensive post-construction report exists, although a number of papers describing various aspects of the computer have been published. This bibliography lists these papers and provides a short description of the computer as a guide to the entries.

The papers cited fall into two classes. The first class is the open literature consisting of journal articles, symposia proceedings, and the like. The second class is Digital Computer Laboratory Reports. These are included because they are rather widely held in the libraries of computer organizations and they have been cited in the literature. The internal Digital Computer Laboratory documents related to construction are not cited here.

Manuscript received January 6, 1965.
The author is with the Department of Computer Science, University of Illinois, Urbana, Ill.
[1] Recently renamed Department of Computer Science.

HISTORY

Planning for ILLIAC II began June 1, 1956, and culminated in 1957 in a report describing the proposed design [38]. Design began in 1957 and final chassis construction began in 1960. In 1962, the two controls, arithmetic unit and core memory began operation with paper tape input and output. At the present time the machine is essentially complete and in use, and work continues on the addition of input-output devices and other peripheral equipment. The work has been supported jointly by the University of Illinois, the Atomic Energy Commission and the Office of Naval Research. The IBM Corporation donated a number of input-output devices.

ORGANIZATION

ILLIAC II is a highly parallel computer, with three simultaneously operating controls. Operations of the floating-point arithmetic unit are controlled by an arithmetic control. Transfer of data between the core memory and the slower memories is controlled by an

interplay control. Other control functions are performed by a supervisory control called Advanced Control. Among the functions of Advanced Control are fetching and storing of operands, address construction and indexing, and partial decoding of the orders for the other two controls. The Advanced Control order code is rather elaborate, and in conjunction with the 13-bit registers in the fast memory it provides for a large variety of 13-bit fixed-point arithmetic and logical operations, except multiplication and division.

The hierarchy of memories consists of the following:

1) Fast transistor memory, 10 words, 0.2 μs.
2) Core memory, 8192 words (soon to be 12 288 words), 1.8 μs.
3) Drum memory 65 536 words, 8.5-ms average access time, 7.8-μs word period.
4) Magnetic tapes and disk files.

The order code contains long and short instructions. A 13-bit short instruction, which occupies only a quarter word, contains four bits to specify an index register containing an address or a fast register containing an operand. A 26-bit long instruction contains in addition a 13-bit address. Long instructions may be packed two to a word. Two words of orders are held in the fast memory. This makes it possible to execute a loop of up to eight short instructions (two words) without any instruction fetches from the core memory. If at the same time the operands are held in the ten-word fast memory, a very fast loop can be written.

A detailed consideration of the size and speed requirements of the various parts of the machine for several classes of problems is given in Taub, et al. [38], which also contains an early version of the order code. Consideration of problem types is also contained in Taub [39]. More detailed descriptions of the organization and the order code are contained in Gillies [8], [9], [11]. Up to date details are given in the ILLIAC II programmer's manual [5].

Arithmetic Unit

The arithmetic unit is asynchronous, double-precision, floating-point. It is radix 4 in almost all respects. Single-precision operands are 52-bits long, with a 45-bit fraction and a 7-bit exponent (base 4) in radix complement representation. The range of normalized single-precision numbers in the memory is

$$4^{-64} \leq |X^y| \leq 4^{63}.$$

Results of most arithmetic operations are not normalized and the programmer is free to normalize or not as he stores them. To aid in fixed-point programming, orders are provided which force the exponent to one of three values, with corresponding shifts in the fraction part. The roundoff which occurs when storing a double-precision arithmetic result in the single-precision memory is obtained by adding 1 or 0 to the last retained fraction bit for discarded fractions greater or less than one half, respectively. The equality case is made dependent on the (presumably random) last retained bit to produce an unbiased roundoff.

A number of features are provided to increase the speed of operation. Redundant number representations and separate carry storage are used within part of the arithmetic unit to eliminate carry propagation during repeated additions such as occur in multiplication. In general a carry bit is provided for each two fraction bits. Multiplier digits, originally having values 0, 1, 2, 3 are recoded to the range $-1, 0, 1, 2$ and two-at-a-time shifts are provided. Two adders are provided so that addition may be performed both while gating from the accumulator (A, Q) to the temporary accumulator (S, R) and vice versa. Radix 4 division was considered by Robertson [30], but rejected in favor of redundant binary nonrestoring division, wherein the quotient digits are generated as $-1, 0, +1$ and then recoded as base 4 digits with values between -3 and $+3$. Carries are assimilated before a store, since the other parts of the computer do not use redundant number representation.

The floating-point arithmetic unit as constructed is described theoretically in Robertson [31] and in detail in Penhollow [19]. Earlier plans were described in Taub, et al. [38] and Wheeler [40]. In addition there were a number of earlier studies. These included redundant number representations by Avizienis [1], [2], [3] and Metze [14], use of redundant number representation in the whole computer instead of just the arithmetic unit by Metze and Robertson [15], separate carry storage adders by Takahashi [37], efficient multiplier and division recodings by Penhollow [18], and efficient division by Robertson [30], Metze [16] and Shively [34].

Speed Independence and Control Design

Theories of asynchronous circuits and speed independence were studied extensively prior to construction. The speed independence problem is stated physically and theoretically in Taub, et al. [38]. Detailed theoretical studies are in Muller and Bartky [17], Shelly [33], and Bartky [4]. A circuit is speed-independent if its function does not depend on the speeds at which its constituent parts operate. Advantages of speed independence are increased reliability and ease of maintenance.

The realization of speed independence used in the controls of ILLIAC II involves the collection of reply signals to insure that all the operations which must be performed at each step are complete before going on to the next step. Some of the problems involved in designing the arithmetic control in this way are described in Swartwout [35], Robertson [32], and Gillies [10]. Advanced Control was designed in a similar but not identical way. The arithmetic unit was not made speed-independent in order to avoid increasing its complexity and cost and decreasing its speed. The electromechani-

cal peripheral devices are inherently synchronous, but the philosophy of speed independence was partly extended to them by the provision of replies and alarms for many of the control signals.

A theoretical study of methods of designing a speed-independent control, including the method actually used for the arithmetic control, is contained in Swartwout [36].

Speeds

Some approximate operation times are as follows.

Floating add or subtract	2.5 to 3.5 μs
Floating multiply	6.3 μs
Floating divide	16.0 μs
Indexing	1.0 μs
13-bit integer orders	2.0 μs
Fast memory	0.2 μs
Core memory	1.8 μs

The times shown for arithmetic do not include instruction or operand accessing times because Advanced Control performs memory accesses concurrently with arithmetic, usually with zero-net time charges. Instruction decoding, address construction and indexing are similarly overlapped with arithmetic, and most absorb no effective time at all.

Fast Memory

Ten words of very fast storage are provided, called the fast memory or flow-gating memory. These ten registers are composed of transistor flip-flops with common input and output buses and special gating arrangements to keep the number of transistors small. The design achieves high speed and high sensitivity along with the usually contradictory high stability by using variable feedback. During the write-in operation, a gate signal lowers the average potential of the flip-flop. This produces the following two effects: 1) information is allowed to flow into the circuit through a diode from the input bus and 2) the feedback in the flip-flop is disabled. This reduces the circuit to a difference amplifier, and the information is stored in the base-emitter capacitances. At the end of the write-in operation the average potentials are raised back to normal, thus cutting off the input diode and allowing the feedback to permanently store the information. The operation time is 0.2 μs. The transistor counts per bit are basic flip-flop 2, output driver 1, write and read drivers and terminations about 2.3.

The fast memory sits at the "crossroads" of the computer, and some of its registers are also intimately identified with other parts of the machine, e.g., the core memory, Advanced Control, the arithmetic control, and the arithmetic unit. Four of the fast registers are also addressable as quarter words, thus providing 16 registers of 13 bits each for use as index registers and for other purposes.

The early plans for the fast memory were given in Taub, et al. [38], and Poppelbaum [20]–[22]. A brief mention is also made in Poppelbaum [24]. Detailed experimental data on the fast memory, including tolerance analyses, waveforms and other details, is given in Guckel, Kunihiro and Crow [12]. A patent covering the flow-gating principle was issued in 1962 [25].

Core Memory

The core memory was originally planned to contain 8192 words of 52 bits plus parity each. There were to be two 4096-word modules, with odd addresses in one module and even addresses in the other to halve the average access time for sequential addresses. The first 4096-word module was completed in 1962. It was word oriented, with one switch core per word and two data cores per bit. Two data cores per bit gave bipolar output and a loading on the switch cores that was virtually independent of the digit pattern. Partial switching was used to increase speed and reduce core heating. Readout was destructive and a restoration cycle was provided.

Early plans for the core memory were described in Taub, et al. [38]. Some earlier experiments were reported in McKay, et al. [13]. Detailed plans for the construction of the first 4096-word module were described in Ray [27]. Theoretical studies of partial switching are contained in Ray [28], [29].

The first 4096-word module was finished in 1962, and has been in operation since then (without the interleaved addresses feature) at a cycle time of 1.8 μs. In 1964, a commercial 8192-word core memory was purchased. The original 4096-word module and 4096 words of the commercial core memory are now in operation with interleaved addresses. This exhausts the addressing capabilities of the original 13-bit address field. The addressing scheme is presently being modified to allow the additional 4096 words also to be used.

Circuits

The basic circuits used in the high-speed portions of the machine are nonsaturating current switching circuits using *pnp* germanium mesa transistors. Switching times are 10 to 40 ns. Early reports on these circuits are Taub, et al. [38] and Poppelbaum and Wiseman [22]. The actual construction was based on a revised design completed in the summer of 1960. A patent covering the asymmetrical flip-flop was issued in 1960 [23]. A tutorial description of some of the memory elements is in Rao [26].

The slower parts of the computer (Interplay, Drum Memory, Input-output Channels, etc.) contain a variety of slower circuits. These include saturating, nonsaturating, current switching, and NOR topologies using germanium transistors.

The computer contains about 55 000 transistors and 133 000 diodes, exclusive of the commercially built input-output devices.

Input-Output and Interrupt

Two input-output systems are provided, a high capacity full word system and a slower quarter word system.

Full word data transfers in the memory hierarchy are between the core memory and one of the other memories or devices. Transfers between the core memory and the ten-word fast memory are supervised by Advanced Control. All other full word transfers are performed by Interplay, which contains the necessary controls and data buffers. Interplay is a wired program computer of a limited sort. It begins a data transfer between the core memory and one of the other memories or devices in response to a command from Advanced Control. After the initial setup, Advanced Control and Interplay operate independently without interaction except that they compete for core memory accesses. Each of the Interplay Channels can be performing a transfer at the same time. Currently there are nine channels in use out of a possible 32. The capacity of Interplay is one word every 3.5 μs.

The slower input-output system, called the special register system, allows Advanced Control to exchange 13-bit characters with up to 64 input-output registers. Each 13-bit transfer requires Advanced Control to execute one order as distinguished from Interplay which operates in parallel with Advanced Control and requires execution of only two Advanced Control orders to transfer a block of data, generally 256 words. The special register system is used for low-speed input-output and to transmit control and status information for peripheral devices.

An interrupt system is connected to certain bits of the special registers. For example, when an Interplay channel completes the transfer of a block of data, a completion signal is provided via one of the special registers. This may, if desired, interrupt the program then running and call a supervisory program to initiate another transfer or take other action. The interrupt system may also be actuated by errors, power failures, requests from consoles, expiration of a time interval, etc.

Magnetic Drum Memory

The Magnetic Drum Memory stores 65 536 words on two 3400-r/min drums. Each word is stored as four 13-bit characters plus parity. The character period is 1.95 μs; the word period is 7.8 μs. Nonreturn-to-zero recording is used at a packing density of 288 bits per inch. Full 52-bit parallel recording with a 1.95-μs word period was considered but not used because it would have required four times as many read and write amplifiers and it would have almost completely occupied the core memory while a drum transfer was in progress. Drum data is written and read in 256-word blocks, with eight blocks per band, and 16 bands per drum. Gaps between the blocks allow for head switching so that following any block transfer, random access to one of the 16 blocks in the next sector may be obtained without waiting.

System Programs

The ILLIAC II software includes an assembler called NICAP, a FORTRAN II translator, and an operating system program. Among other things, NICAP handles the multiple-orders-per-word problem and translates complex address field expressions, including nested parentheses to any depth. Parts of address field expressions which can be evaluated at translation time are so evaluated. The remaining additions and subtractions are prepared for execution at run time by the 13-bit fixed-point arithmetic unit in Advanced Control; multiplications and divisions are prepared for execution by the floating-point arithmetic unit. The address field compilation algorithm is described in Gear [6].

The FORTRAN II translator produces assembly language in a single pass. Effective use of the Drum Memory enables the translator to proceed without the use of magnetic tapes, thus gaining an order of magnitude in speed. The operating system program provides for batch processing. The various system and library programs are described in a user's manual [5] and in a compiler writer's manual [7].

Acknowledgment

In addition to the authors cited, a large number of other people participated in the design and construction of ILLIAC II. Particular appreciation is expressed to Profs. J. E. Robertson and D. B. Gillies for their assistance in the preparation of this bibliography.

Bibliography

[1] Avizienis, A., A study of redundant number representations for parallel digital computers, Ph.D. thesis, University of Illinois, Urbana, 1960. (Also DCL Rept 101, Digital Computer Lab., University of Illinois, May 20, 1960.)
[2] ——, Signed-digit number representations for fast parallel arithmetic, *IRE Trans. on Electronic Computers*, vol EC-10, Sep 1961, pp 389–400.
[3] ——, On a flexible implementation of digital computer arithmetic, in *Information Processing 1962*. Amsterdam, The Netherlands: North Holland Publishing Co., 1963, pp 664–670.
[4] Bartky, W. S., A Theory of asynchronous circuits III, DCL Rept 96, Digital Computer Lab., University of Illinois, Urbana, Jan 6, 1960.
[5] Gear, C. W., Ed., ILLIAC II manual, Digital Computer Lab., University of Illinois, Urbana, Mar 1963. This loose-leaf programmer's manual is kept up to date with revision pages.
[6] ——, Optimization of the address field compilation in the ILLIAC II, *The Computer J.*, vol 6, Jan 1964, pp 332–335.
[7] ——, Ed., New Illinois compiler and assembler programming system systems manual, Digital Computer Lab., University of Illinois, Urbana, Jul 1964. This loose-leaf manual describes the system programs, including the NICAP and FORTRAN translators.
[8] Gillies, D. B., Organization of a very-high-speed computer, DCL Rept 93, Digital Computer Lab., University of Illinois, Urbana, Aug 24, 1959.
[9] ——, The design of a very-high-speed scientific computer, notes for course Theory of Computing Machine Design, University of Michigan, Ann Arbor, Jun 26–29, 1961.
[10] ——, A flow chart notation for the description of a speed independent control in *Switching Circuit Theory and Logical Design*, AIEE Publication S-134, 1961, pp 109–110.
[11] ——, Order code for the new Illinois computer, DCL Internal Rept, Digital Computer Lab., University of Illinois, Urbana, Jun 15, 1962, revised Aug 14, 1962. Current version is available as ch 3 of the ILLIAC II Manual [5].
[12] Guckel, H., T. Kunihiro, and R. K. Crow, Final report—flow gating, DCL Rept 106, Digital Computer Lab., University of Illinois, Urbana, Mar 24, 1961.

[13] McKay, R. W., N. N. Yu, and C. Pottle, A one-word model of a word-arrangement memory, DCL Rept 79, Digital Computer Lab., University of Illinois, Urbana, May 1957.
[14] Metze, G., A study of parallel one's complement arithmetic units with separate carry or borrow storage, Ph.D. thesis, University of Illinois, Urbana, 1958. (Also DCL Rept 81, Digital Computer Lab., University of Illinois, Nov 11, 1957.)
[15] Metze, G., and J. E. Robertson, Elimination of carry propagation in digital computers, in *Information Processing*. Paris, France: UNESCO, 1960, pp 389–396.
[16] Metze, G., A class of binary divisions yielding minimally represented quotients, *IRE Trans. on Electronic Computers*, vol EC-11, Dec 1962, pp 761–764.
[17] Muller, D. E., and W. S. Bartky, A theory of asynchronous circuits, in *Annals of the Harvard Computation Laboratory*, pt I, vol 29. Cambridge, Mass.: Harvard University Press, 1959, pp 204–243.
[18] Penhollow, J. O., A study of arithmetic recoding with applications in multiplication and division, Ph.D. thesis, University of Illinois, Urbana, Sep 1962. (Also DCL Rept 128, Digital Computer Lab., University of Illinois, Sep 10, 1962.)
[19] ——, The arithmetic subsystem of the new Illinois computer, DCL Rept 160, Digital Computer Lab., University of Illinois, Urbana, Jan 24, 1964.
[20] Poppelbaum, W. J., Flow gating, *Proc. WJCC*, May 1958, pp 138–141.
[21] ——, Flow gating, DCL Rept 83, Digital Computer Lab., University of Illinois, Urbana, July 10, 1958.
[22] Poppelbaum, W. J., and N. E. Wiseman, Circuit design for the new Illinois computer, DCL Rept 90, Digital Computer Lab., University of Illinois, Urbana, Aug 20, 1959.
[23] Poppelbaum, W. J., Transistor flipflop, U. S. Patent 2933621, Apr 19, 1960.
[24] ——, Millimicrosecond computer circuits, *NEREM Record*, 1960, pp 22–23.
[25] ——, Flow gating, U. S. Patent 3067339, Dec 4, 1962.
[26] Rao, P. V. S., Some memory elements used in ILLIAC II, DCL Rept 119, Digital Computer Lab., University of Illinois, Urbana, Jun 21, 1962.
[27] Ray, S. R., Design of the core storage unit, DCL Rept 91, Digital Computer Lab., University of Illinois, Urbana, Aug 21, 1959.
[28] ——, Engineering model of a partial switching effect in ferrite cores, Ph.D. thesis, University of Illinois, Urbana, 1961. (Also DCL Rept 111, Digital Computer Lab., University of Illinois, Sep 5, 1961.)
[29] ——, Model of partial switching in polycrystalline ferrites, IEEE Publication T-149, 1963, pp 10-7-1 to 10-7-3.
[30] Robertson, J. E., A new class of digital division methods, *IRE Trans. on Electronic Computers*, vol EC-7, Sep 1958, pp 218–222.
[31] ——, Theory of computer arithmetic employed in the design of the new computer at the University of Illinois, notes for course Theory of Computing Machine Design, University of Michigan, Ann Arbor, Jun 13–17, 1960.
[32] ——, Problems in the physical realization of speed independent circuits, in *Switching Circuit Theory and Logical Design*, AIEE Publication S-134, 1961, pp 106–108.
[33] Shelly, J. H., The decision and synthesis problems in semimodular switching theory, Ph.D. thesis, University of Illinois, Urbana, 1959. (Also DCL Rept 88, Digital Computer Lab., University of Illinois, May 29, 1959.)
[34] Shively, R. R., Stationary distributions of partial remainders in S-R-T digital division, Ph.D. thesis, University of Illinois, Urbana, 1963. (Also DCL Rept. 136, Digital Computer Lab., University of Illinois, May 15, 1963.)
[35] Swartwout, R. E., One method of designing speed-independent logic for a control, in *Switching Circuit Theory and Logical Design*, AIEE Publication S-134, 1961, pp 94–105.
[36] ——, Further studies in speed-independent logic for a control, Ph.D. thesis, University of Illinois, Urbana, 1962. (Also DCL Rept 130, Digital Computer Lab., University of Illinois, Dec 13, 1962.)
[37] Takahashi, S., Separate carry storage adders, DCL Rept 97, Digital Computer Lab., University of Illinois, Urbana, Mar 7, 1960.
[38] Taub, A. H., D. B. Gillies, R. E. Meagher, D. E. Muller, R. W. McKay, J. P. Nash, W. J. Poppelbaum, and J. E. Robertson, On the design of a very high-speed computer, DCL Rept 80, Digital Computer Lab., University of Illinois, Urbana, 1st ed., Oct 1957, 2nd ed., Apr 1958.
[39] Taub, A. H., Machine organization with respect to problem types with particular emphasis on the scientific computer, notes for course Theory of Computing Machine Design, University of Michigan, Ann Arbor, Jun 20–24, 1960.
[40] Wheeler, D. J., The arithmetic unit, DCL Rept 92, Digital Computer Lab., University of Illinois, Urbana, Aug 21, 1959.

Reprinted from IEEE TRANSACTIONS
ON *ELECTRONIC COMPUTERS*
Volume EC-14, Number 3, June, 1965
Pp. 399-403