

New Illinois Computer の演算および制御装置*

相 磯 秀 夫**

米国イリノイ大学電子計算機研究所では、1956年頃から超高速電子計算機 New Illinois Computer の設計、製作に力を注いできたが、このほどその設計と製作をほぼ完了し、現在最終調整を急いでいる。

New Illinois Computer の目的とするところは、超高速の非同期式電子計算機の可能性を世に示すことと、学内の計算機需要に答えることである。この計算機の運転速度の目標は、1952年に同大学で作られた Illiac (加算 $80\mu\text{s}$, 乗算 $700\mu\text{s}$, 呼出し時間 $18.5\mu\text{s}$) の約100倍である。設計を終了した現在 New Illinois Computer に期待できる速度は、加算 $1.6\sim 2.0\mu\text{s}$, 乗算 $6\sim 7\mu\text{s}$, 呼出し時間 $1.5\sim 2.0\mu\text{s}$ 程度であるが、先廻り制御方式を採用し、入出力装置の機能を改善することによって、目標とする運転速度が得られるものと思われる。

New Illinois Computer の特長は、

- (1) トランジスタによる高速直結形回路から構成されていること、
 - (2) 制御方式に非同期式を採用していること、
 - (3) 高速演算装置を備えていること、
 - (4) 大容量の高速記憶装置と高度の先廻り制御方式を備えていること、
- である。以下、これらの特長について簡単に述べよう。

1. 基本回路

計算機の論理回路は原則としてトランジスタから構成されている。使用されているトランジスタは Western Electric GF 45011 と Texas Instrument S 116 で共に Mesa 形 PNP トランジスタである。回路はすべて不飽和形であるため、電圧のクリップにダイオード Transitron S 577 G が用いられている。

基本回路の設計に際しては、

- (1) トランジスタのエミッタ電流に対するエミッタ・ベース間の順方向電圧降下を示す曲線。(これはコレクタ・ベース間の電圧をパラメータにとっており、曲線はその特性の最大値および最小値を示し

ている)

- (2) ダイオードの順方向電流に対する電圧降下を示す曲線。(これも最大値と最小値が与えられている)

- (3) トランジスタの電圧電流特性。(コレクタ消費電力 120 mW , $0.93 \leq \alpha \leq 1$, コレクタ・ベース間の逆耐電圧 25 V)

- (4) 回路の許容裕度。(飽和電圧 1 V , 雑音裕度 0.3 V , 電圧および抵抗値の変動 $\pm 3\%$ (実物は $\pm 1\%$))

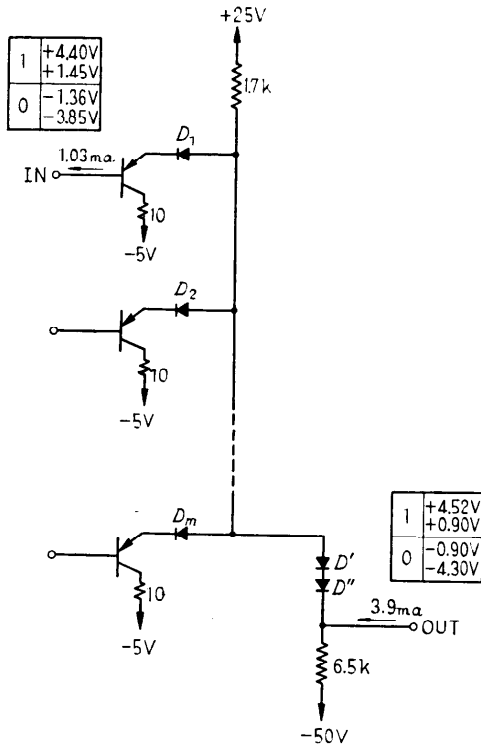
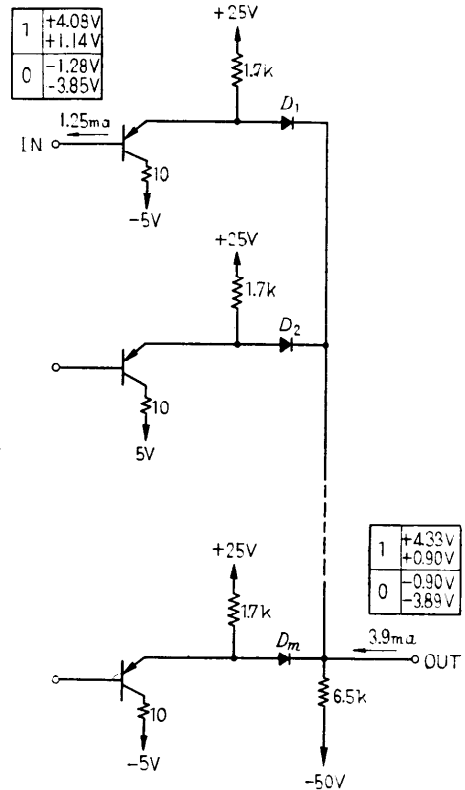
- (5) 動作方式。(非同期式、直結形回路)

などの基本的な資料と条件を考慮に入れ、信号 1 と 0 に対する両極端の場合について直流的な解析 (Worst Case Tolerance Analysis) を行なっている。過渡的な解析としては浮遊容量に流れ込む充電電流をさらに見込んで行なっている。

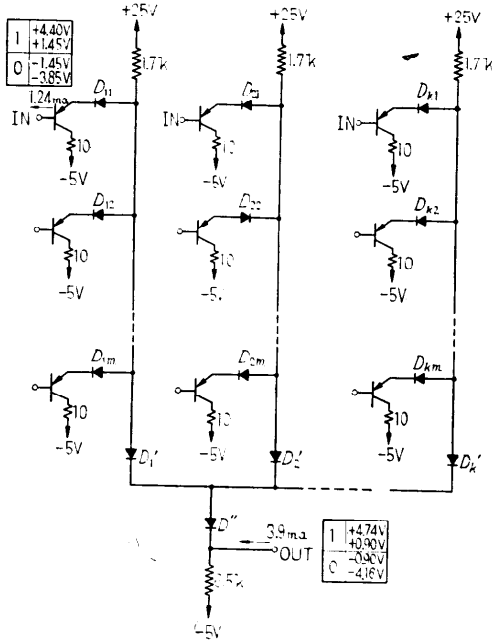
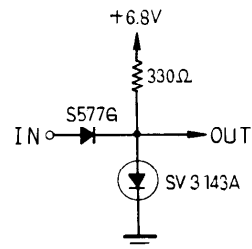
基本回路の種類は多種多様であるが、出力の性質から大別して、入力の数や出力の数によって回路の出力電圧が変わる Non-Restoring Circuit と常に一定の出力電圧が保償される Restoring Circuit がある。第1図～第3図は前者の例である。これらの回路で多段論理回路を構成することができるが、出力電圧が次段回路の入力制限電圧を越えるものは使用を許されない。出力をクリップ (Bump ともいう) して逃げられるものは第4図に示すような回路を附加して用いる。この回路は、パワー・ダイオードに電流を流した時にその順方向電圧降下が電流の大きさにあまり関係がないという特性を利用したものである。Non-Restoring Circuit の fan-out はいずれも3であるが、動作速度はかなり速く 4 ns 程度である。第5図の NOT 回路は Restoring Circuit の一例である。この種の回路は電流スイッチ形の増幅器、ブリーダ (抵抗、またはゼーナ・ダイオードと抵抗を組み合わせたもの)、およびエミッタ・ホローから構成されている。Restoring Circuit では $3.3\text{ k}\Omega$ のエミッタ・ホローから5本の fan-out が得られるが、動作速度は $10\sim 20\text{ ns}$ 程度である。NOT 回路のほかに、AND, AND-NOT, Exclusive OR, Equivalence, Level Restorer, Flip-Flop, Cable Drivers, Slow Circuits など多種

* Arithmetic and Control Units for the New Illinois Computer, by Hideo Aiso (Electro-technical Laboratory, Tokyo)

** 電気試験所

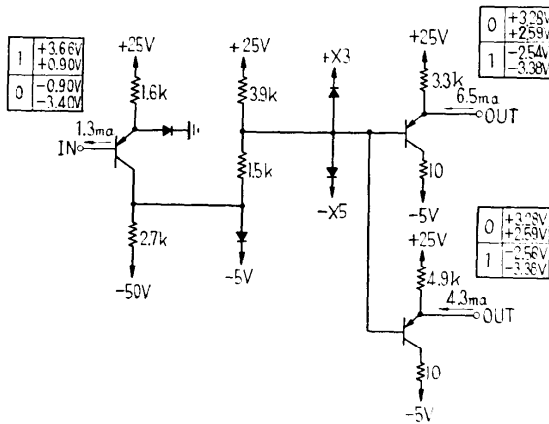
第1図 m 入力 AND 回路 ($m \leq 8$)第2図 m 入力 OR 回路 ($m \leq 8$)

類のものがあるが、いずれも上述の回路がその基本形となっており、できるだけ動作速度が速くなるように設計されている。

第3図 AND-OR 回路 ($m \leq 8, K \leq 5$)

第4図 Bump 回路 (+X3)

基本回路の Layout は $2' \times 1'$ 、 $1\frac{1}{2}' \times 1'$ および $1' \times \frac{3}{4}'$ の3種類のシャーシに、それぞれの論理設計に応じて行なわれる。したがって、各シャーシ間の互換性は全くない。配線の長さは奇生振動や雑音の発生を防ぐ意味で、いかなる配線も2フィート以内でなければならない。これを越す配線はケーブル駆動回路を介して、特性インピーダンス 140Ω の Twisted Pair



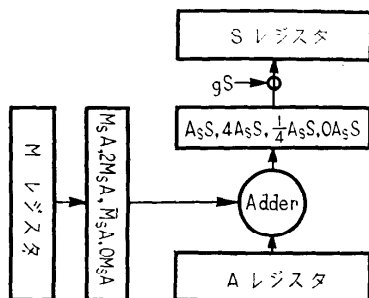
第5図 NOT 回路

に置き換える必要がある。実際の Layout は論理設計の重要性を考慮に入れた上で、ケーブル駆動回路をなるべく使わないように考慮して行なわれている。

2. 非同期式制御方式

New Illinois Computer に採用されている非同期方式は Dr. D.E. Muller が提唱する Speed Independent Logic と称する方式である。非同期方式において最も問題となるのは各回路間の動作の競争であるが、Speed Independent Logic はこれを解決する一方法である。一般に非同期式が同期式に比べて優れている点は、回路の動作速度を最大限に利用し得ることであるが、Speed Independent Logic ではさらに回路の誤動作を見つけ出し得るように容易に設計できる利点がある。

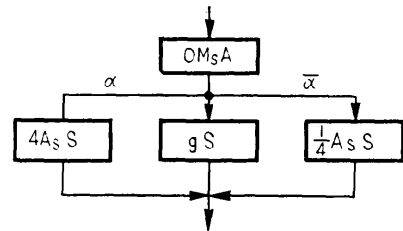
一方、前者が後者に比べて劣る点は、設計が非常に複雑になり、多くの素子を必要として不経済になることである。実際に New Illinois Computer でも演算



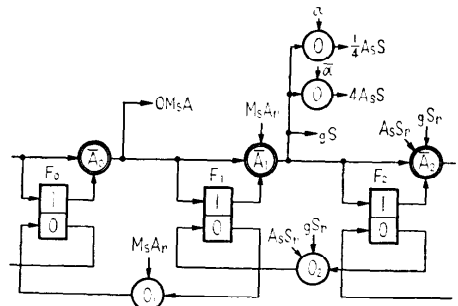
第6図 演算装置の一部

装置の大部分は Speed Independent Logic に倣していない程である。次に一例を挙げてこの方式の概念を示そう。

第6図は演算装置の一部であるが、第7図に示すように、ある条件に応じて A レジスタの内容 $[A]$ ($[X]$ は X の内容を表わす) を 4 倍または $1/4$ 倍して B レジスタに移す場合を考えてみる。まず制御装置は $[M]$ が加算器に加わらないように M レジスタと加算器の間にある選択回路の $OMsA$ を選ぶ。次に条件 α が 1 ならば $4AsS$ を選択回路で選び 4 倍の $[A]$ を S レジスタに移す準備をする。もし条件 α が 0 ならば $1/4AsS$ を選んで $1/4$ 倍の $[A]$ を S レジスタの入力として与える。ついでゲート信号 gS をオンにして S レジスタに加算器の出力を入れる。第8図はこの機能を行なうための制御回路を Speed Independent Logic で設計した例である。



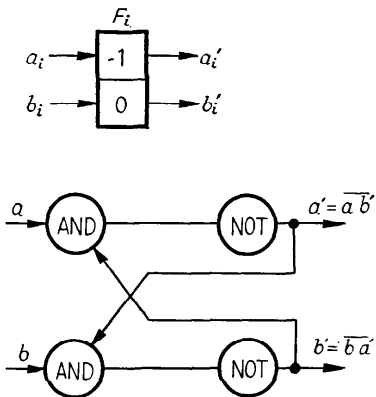
第7図 制御の流れ図



第8図 制御回路

New Illinois Computer では Negative Logic を用いているが第9図に示すフリップ・フロップは常時 $a=1, b=1, a'=0, b'=1$ になっている。いま $\bar{A}_0$ の出力が何らかの方法で 1 から 0 になった場合を考えよう。この時 F_0 は $a_0=1, b_0=1, a'_0=1, b'_0=0$ になっているが F_1 は $a_1=0, b_1=1, a'_1=1, b'_1=0$ にな

る。一方 $OMsA$ も 1 から 0 になり M からの入力を絶つが、この時 MsA の選択回路から“確かに MsA 選択回路が働いている”ということを知らせる応答信号 $MsAr$ も 1 から 0 になる。したがって、 0_1 の出力も 1 から 0 になり F_0 は $a_0=1, b_0=0, a'_0=0, b'_0=1$ となって $\bar{A}_0$ の出力が 0 から 1 に戻る。これに伴って、 $MsAr$ も 0 から 1 に変わり、 0_1 の出力が 1 に戻って F_0 は定常状態になる。この時 F_1 は最初に考えた F_0 の状態と同じであり、 $\bar{A}_1$ の出力を 1 から 0 に変えて次段に制御信号を送る。

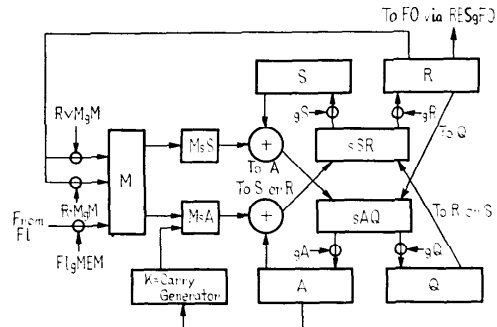


第9図 フリップ・フロップとその機能

第2段階では gS ，したがって gSr を 1 から 0 にするが、この時 α が 1 ならば $4AsS$ を、また α が 0 ならば $1/4AsS$ を 0 にし、それに伴って $AsSr$ も 1 から 0 に変わる。 gSr と $AsSr$ が共に 0 になれば 0_2 の出力が 0 になり、 $\bar{A}_1$ の出力は 1 に戻る。以下同様、次々に制御信号が送られて行く。第8図は最も簡単な制御回路の一例であるが、動作の終了を検出する OR 回路と AND-NOT 回路、および信号の流れを制御するフリップ・フロップの組み合わせは、全体の制御装置を通じて基本的なものである。制御装置の設計に際しては、まず制御信号の流れ図が書かれ、それに従って何種類か準備されている実際の回路に置き換えて行く。より複雑な回路ならびに応答信号の発生回路についての説明は他の機会にゆずる。

3. 演算装置

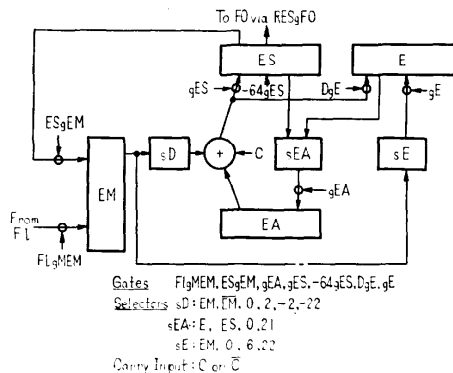
第10図は主演算装置 (MAU) のブロック図である。MAU は浮動小数点演算における仮数部を処理する部分で、第10図はその主なレジスタ、ゲート、選択回



Gates: $F1gMEM, R-MgM, RVMgM, gA, gQ, gR, gS, RESgFO$

Selectors: $sSR: \frac{1}{2}AQ, 4AQ, QA, OA, 0 \rightarrow SR$
 $sAQ: \frac{1}{2}SR, SR, ASR, SO \rightarrow AQ$
 $MsA: 0, M, M, M, M, M, M, M \rightarrow A\text{-adder}$
 $MsS: (1) CM, M, 2M, M, 2M, \sim M \rightarrow S\text{-abtn}$
 (2) $-2^{-44}, 2^{-44}$
 (3) $KMsS$

第10図 主演算装置 (MAU)



Gates: $F1gMEM, ESgEM, gEA, gES, -64gES, DgE, gE$

Selectors: $sD: EM, EM, 0, 2, -2, -22$

$sEA: E, ES, 0, 21$

$sE: EM, 0, 6, 22$

Carry Input: C or $\bar{C}$

第11図 指数演算装置 (EAU)

路および加算器の関係を示したものである。第11図は指数部を処理するための指数演算装置 (EAU) である。

M および EM は磁心記憶装置から読み出された被演算数を貯えるためのレジスタで、 M は 46 ビット、 EM は 8 ビットから構成されている。被演算数の仮数部を x ，指数部を y とすると、記憶装置の内部で数値 w は $-1 \leq x \leq 1-2^{-44}$ ， $-64 \leq y \leq 63$ の値をとる。 M ， EM レジスタ内部では

$$x = -2m_{-1} + \sum_{i=0}^{44} 2^{-i} \cdot m_i$$

$$y = -128em_7 + \sum_{j=0}^6 2^j \cdot em_j$$

の値をとる。すなわち、演算装置内部においては、指

数は -128 から +127 までの値をとり得る。

New Illinois Computer では2進数2桁を一つの単位 (base 4 という) として取扱う方式を採用し、さらに A, S レジスタの1桁おきに桁上げを貯えるビットを設けて演算の高速化を図っている。base 4 の演算により高速桁移動が可能になり、桁上げを貯えるビットを備けることによって、桁上げ伝播による演算の遅れを少なくすることができる。分離された桁上げビットは必要に応じて桁上げ清算回路Kを介して清算される。A, Q およびEレジスタが累算器を構成するが、その内部において数値 n は

$$n = f \cdot 4^e$$

$$f = -2a_{-1} + a_0 + a_0^* + \sum_{i=1}^{44} 2^{-i} a_i + \sum_{j=1}^{22} 2^{-2j} a_{2j}^* + 2^{-46} \sum_{k=1}^{42} 2^{-k} q_k, \quad -1 \leq f < 1,$$

$$e = -128e_7 + \sum_{i=6}^0 2^i e_i, \quad -128 \leq e < 128$$

なる値をもつ。SおよびQレジスタはそれぞれAおよびQレジスタに対する予備的なレジスタで、[A, Q]をS, Rに移す際に[M]をMsA選択回路を介して加えることができる。また[S, R]をA, Qに移す際にも[M]をMsS選択回路を介して加えることができる。A, Q, S, R レジスタに関する選択回路はブロック図の下に示す機能をもち、base 4 の1桁の桁移動、レジスタ間の内容の交換などを行うことができる。Mレジスタに関する選択回路は $\pm[M]$, $\pm 2[M]$, $0M$ を加算器に与えることができ、この機能によって加減算、2進1桁の桁移動などが可能になる。

EAU内の選択回路も指数演算に必要な機能をもち、カウンタとして利用できるように定数を選ぶことができる。ゲート信号はそれぞれのレジスタの入力側に付いており、新しい情報を入れる際にオンにする。RvMgM, R·MgMは[R]と[M]との間の論理作用を行なわせるためのゲート信号である。ESレジスタの -64_e ESは浮動小数点の数値を0にする時にオンにする。このとき、ESレジスタは -64 がセットされる。数値0は、 $f=0$, $e=-64$ で表現される。

演算装置にはこれらの他に、演算を高速に行なうために零検出回路、標準化検出回路、最下位桁上げ清算回路、乗除算制御装置が設けられている。

演算装置に関する命令は第1表に示すように37種類あるが、これらは12のSub-sequenceの組み合わせで行なわれる。演算制御装置の内部にある読取器には命令の状態を示すフリップ・フロップが九つあり、

第1表 演算装置に関する命令

00	CBS	Clear subtract. Replace n by $-x \cdot 4^v$ if $x \neq -1$, or $+1/4 \cdot 4^{v+1}$ if $x = -1$.
01	CST	Clear subtract twice. Replace $f \cdot 4^e$ by $-2x \cdot 4^v$ if $-2x$ is in range, or $-1/2x \cdot 4^{v+1}$ otherwise.
02	CAD	Clear add. Replace n by $x \cdot 4^v$.
03	CAT	Clear add twice.
04	NOT	Clear add digit-wise complement. Replace n by $(-x - 2^{-44}) \cdot 4^v$.
05	AND	Logical multiply. For every i , replace a_i by $x_i \cdot a_i$ (e is unchanged).
06	OR	Replace a_i by $x_i \vee a_i$, for every i .
07	DBA	Replace a by $2a \bmod 2$. Do not set OV.
10	SUB	Subtract w from n .
11	SBE	Subtract from exponent. An 8 bit operand is subtracted from e . This operand is the rightmost 8 bits of an address.
12	ADD	Add w to n .
13	ADE	Add to exponent an 8 bit operand.
14	SSC	Subtract and store clear. Has the effect of executing SUB followed by STC (Code 26).
15	CSE	Clear subtract exponent. (8 bit operand).
16	ASC	Add and store clear. ADD followed by STC. (Code 26).
17	CAE	Clear add exponent (8 bit operand).
20	MPY	Multiply.
21	DIV	Divide by w .
22	NDV	Negative divide. Divide by $-w$.
23	VID	Inverse divide. The normalized rounded value of f is used as divisor, and w is the dividend.
24	STR	Store. The accumulator is normalized and $a_r \cdot 4^e$ is normalized and sent to memory. The accumulator is normalized and unrounded. If $e \geq 64$, OV is set and e is stored modulo 64.
25	XCH	Exchange. The new value of the memory register is the same as if STR were executed. The new value of n is the same as if CAD (Code 0.2) were executed.
26	STC	Store clear. First STR is executed. Then $(f - a_r) \cdot 4^{e+22}$ is placed in A, 0 is placed in Q and $e-22$ is placed in E.
27	STN	Store negatively. The accumulator is normalized and $-a_r \cdot 4^e$ is transferred to the accumulator. Then STR is executed.
30	STF	Store fixed point. A fixed point number is either 0.4^{-64} or has exponent zero. If Z is not on, n is converted to fixed point by shifting right or left and counting up or down respectively on e until e becomes zero. Then $a_r \cdot 4^0$ or 0.4^{-64} is sent to memory. If overflow occurs during the shift or as a result of round-off, OV is set and a_r is held modulo 2.
31	SIF	Store integer part as a floating point number. The accumulator is shifted until its exponent becomes $+22$ and $a \cdot 4^{22}$ is transferred to memory. Zero is represented in memory as 0.4^{-64} .
32	SEQ	Store with exponent equal. The accumulator is shifted until its exponent becomes equal to a given 7 bit number, $a_r \cdot 4^e$ or 0.4^{-64} is transferred to memory. If a_r overflows it is held modulo 2 and OV is set.

- 33 SIA Store integer address. The accumulator is shifted until its exponent is +6, and its first 13 bits are stored into a modifier register.
- 34 STU Store unnormalized. $a \cdot 4^e$ or $(a \cdot 1/4) \cdot 4^{e+1}$ or $0 \cdot 4^{-64}$ is transferred to memory.
- 35 STL Store logical $a \cdot 4^e$ is transferred to memory. The last 7 bits of e are used and Z does not affect the operation and OV is not set.
- 36 STQ Store Q. $q \cdot 4^e$ is transferred to memory in a manner similar to STL.
- 37 SEX Store exponent. The 8 bit exponent is extended to 13 bits by sign digit duplication, and is stored in a modifier register.
- 40 Unassigned.
- 41 LDQ Load Q. Then on-sign bits of x ($x_1, x_2, \dots, x_{14}$) are transferred to Q.
- 42 DAV Difference absolute value. n is partially normalized and $p = a \cdot 4^e$ is formed. Then, in effect, $-|w|$ is formed by CAD or CSB, and $|p| - |w|$ is formed by ADD or SUB.
- 43 SRM Store remainder from immediately preceding divide order.
- 44 Unassigned.
- 45 LRS Long right shift. A 7 bit operand tells the number of base 4 logical right shifts to be performed on f . The sign of f is not duplicated, and e is not changed. If the 7 bit operand is negative, left shifts are called for. These do not set OV.
- 46 Unassigned.
- 47 ARS A right shift. Similar to LRS except a instead of f is affected.
- LIN Load IN. This is not actually an arithmetic control order, but normally it should precede SEQ. The address generated by Advanced Control is copied into all 4 quarters of IN.

これによって12の Sub-sequence を使い分けている。その Sub-sequence は、

- (1) Add sequence (SUB, ADD, SSC, ASC)
- (2) Clear-add type sequence (CSB, CST, CAD, CAT, NOT, AND, OR, DBA)
- (3) Store sequence (STU, STL, STQ, STX, SRM)
- (4) Shift sequence (LRS, ARS)
- (5) Store preliminaries sequence (STF, SIF, SEQ, SIA)
- (6) Exponent arithmetic sequence (SBE, ADE, CSE, CAE)
- (7) Load Q sequence (LDQ)
- (8) Normalize sequence (MPY, DIV, NDV, VID, STR, XCH, STC, STN, DAV)
- (9) Difference absolute value sequence
- (10) Divide sequence
- (11) Multiply sequence

(12) Correct overflow and detect zero sequence

である。演算制御装置は各 Sequence の制御信号の流れに従って、第8図に示したような、Speed Independent Logic で設計されている。

ここで桁上げビットをもった2進加算器について簡単に述べておこう。

いま a_i, m_i, s_i をそれぞれ i 桁目の加数, 被加数, その和とし, z_i を i 桁目への桁上げとすると, 次の関係式が成り立つ。

$$s_i = (a_i \oplus m_i) \oplus z_i \quad (1)$$

$$z_{i-1} = (a_i \oplus m_i) z_i \vee a_i m_i \quad (2)$$

ただし, i が小さいほど桁の重みは大きい。

$$(a_i \oplus m_i) z_i = b_{i-1} \quad (3)$$

とすれば (2) 式は

$$\begin{aligned} z_{i-1} &= b_{i-1} \vee a_i m_i \\ z_i &= b_i \vee a_{i+1} m_{i+1} \end{aligned} \quad (4)$$

となる。(4) 式の関係 (1), (3) 式に代入すれば, 両式は

$$s_i = (a_i \oplus m_i) \oplus (b_i \vee a_{i+1} m_{i+1}) \quad (5)$$

$$b_{i-1} = (a_i \oplus m_i) \cdot (b_i \vee a_{i+1} m_{i+1}) \quad (6)$$

となる。この二つの式から

$$s_i \cdot b_{i-1} = 0 \quad (7)$$

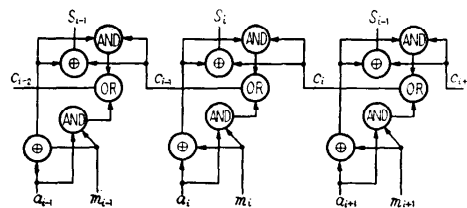
$$a_i \cdot m_i \cdot b_{i-1} = 0 \quad (8)$$

なる関係式が得られる。すなわち $(a_i \cdot m_i)$ と (b_{i-1}) は共に1になり得ない。 s_i は次の加算では a_i になるのであるから b_{i-1} を伝播させずに貯えておいても上の関係式は保たれる。したがって、下位からの桁上げは起らない。貯えられた b_{i-1} が c_{i-1} として次の加算に表われるとすれば

$$s_i = (a_i \oplus m_i) \oplus (c_i \vee a_{i+1} m_{i+1}) \quad (9)$$

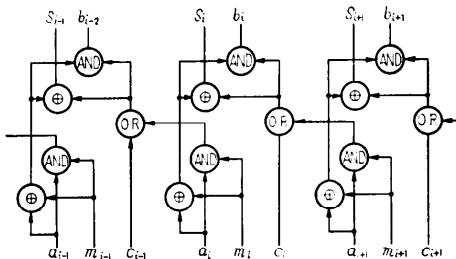
$$b_{i-1} = (a_i \oplus m_i) \cdot (c_i \vee a_{i+1} m_{i+1}) \quad (10)$$

となり、累算器における数 x は擬和 a_i と桁上げ c_{i-1} によって表わされる。第12図は (1), (2) 式を満足する普通の加算器であるが、最下位の桁から最上位の桁への桁上げの伝播で加算速度がきまる。第



第12図 普通の加算器

13 図は (9), (10) 式を満足する回路であるが, この回路では桁上げの伝播がないから加算は極めて高速に行なわれる。ただし, 真の値を得るには貯えられた桁上げを清算しなければならない。New Illinois Computer では 1 桁おきに桁上げを貯える方式を採用している。



第 13 図 桁上げビットを別にもつ加算器

New Illinois Computer の演算に関する機能は浮動小数点に限られるが, 固定小数点もその特殊な場合とみなして取扱うことができる。演算の一例として除算の方式について簡単に述べよう。

採用している方式は Non-Restoring による 2 進法の除算を多少変形したものである。一般に除算の過程においては

$$2(w_i - y_i \cdot m) = w_{i+1} \quad (11)$$

$$2(w_{i+1} - y_{i+1} \cdot m) = w_{i+2} \quad (12)$$

なる関係がある。ここに w_i , y_i はそれぞれ加減算の i 回目の繰返しにおける部分剰余, および商である。また m は除数である。 y_i は 1, 0, -1 の値を取り得る。(12) 式を 2 倍すれば

$$2w_i - (y_i) \cdot 2m = w_{i+1} \quad (13)$$

$$\{w_{i+1} - (y_{i+1}) \cdot m\} \cdot 4 = w_{i+2} \quad (14)$$

となる。除算はこの関係式にしたがって行なわれる。すなわち, m が M レジスタに, $2w_i$ が S, R レジスタに最初貯えられているものとすれば (13) 式は [S, R] を A, Q に移す際に y_i の値に応じて $\pm 2m$, または 0 を加算器に加えることによって次の剰余を得ることを示している。また次の繰返しでは [A, Q] を S, R レジスタに移す際に y_{i+1} の値に応じて $\pm m$, または 0 を加算器に加え, 4 倍にすることによって (14) 式が得られる。商 y_i , y_{i+1} は加算器の出力と除数を調べることによって決定される。すなわち, 剰余と除数の符号が一致していなければ次の商は +1, もし一致していれば次の商は -1 である。もし剰余が除数の大きさの 1/2 またはそれよりも小さければ次の商は 0

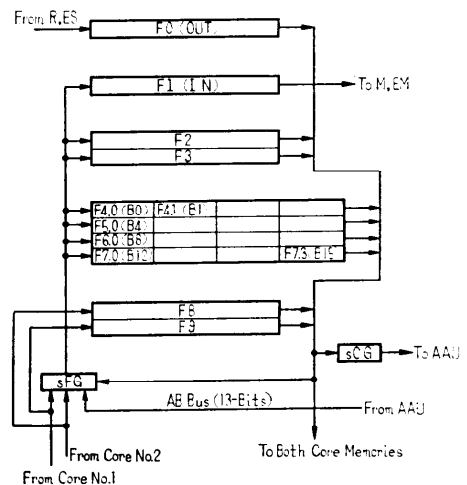
である。商を立てる論理回路は Speed Independent Logic になっており, 加算器の一部と除数の符号を比較して MsA または MsS 選択回路を制御している。

乗算の方式も極めて高速な方法と採用しているが, ここでは省略する。

4. 先廻り制御装置

New Illinois Computer の先廻り制御装置の主な役割は,

- (1) 必要な命令語を磁心記憶装置からトランジスタによる高速記憶回路 (Flow-Gating と呼ばれる) にとり出してくる。
- (2) 命令が実行されるにしたがって, 命令カウンタの内容を変える。
- (3) アドレス計算を行ない, それにしたがって被演算数を磁心記憶装置から読出してくる。
- (4) 演算装置を用いないですむ命令を実行する。
- (5) 割込み信号に対して適切な処置をすることである。先廻り制御装置は磁心記憶装置, 演算装置, および入出力装置をも制御し, 計算機全体の実効的な運転速度の向上を図るためのものである。

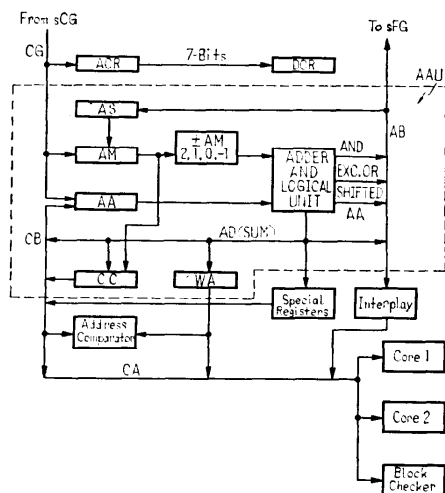


第 14 図 高速記憶回路 (Flow-Gating) によるバッファ

第 14 図は先廻り制御装置の高速記憶回路によるバッファ・レジスタである。高速記憶回路は Flow-Gating と呼ばれ, トランジスタで構成されている。サイクル時間 0.15~0.2 μ s の動作速度を持っている。各レジスタは 52 ビットからなっているが, これらは選択回路 sFG を介して磁心記憶装置から情報を受取

り sCG を介してアドレス演算装置に情報を与えている。

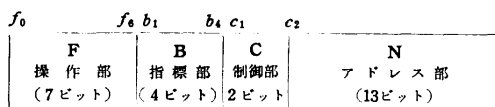
FO は OUT レジスタと呼ばれ、演算装置のためのバッファで“Store”命令によって R および ES レジスタの内容を一時的に貯える。F1 は IN レジスタと呼ばれ、演算命令の被演算数を貯えるバッファで、ここから M および EM レジスタに移される。F2, F3 は一時的なバッファである。F4 から F7 はそれぞれが4分割されており、Bレジスタとして用いられるが、F2 または F3 と同じように用いることもできる。F8, F9 は命令語のためのバッファで磁心記憶装置に直結している。



第15図 アドレス演算装置 (AAU)

第15図はアドレス演算装置 (AAU) と入出力装置、磁気ドラム記憶装置および奇数番地と偶数番地ごとにまとまっている磁心記憶装置 #1, #2 との関係を示している。磁心記憶装置はそれぞれ 4,096 語の容量をもち、1.5~2.0 μ s のサイクル時間で、独立的に動作する。アドレス計算と B レジスタによる変更計算は 13 ビットの AA, AM, AS レジスタと加算器によって行なわれる。WA レジスタは“Store”命令のアドレスを一時的に記憶しておくためのものであり、アドレス比較回路は [WA] と同一番地のものを読み出すことを防ぐために設けられている。CC は 12 ビットからなるレジスタで命令語の所在を示している。なお命令語は二つの磁心記憶装置から同時にとり出される。

アドレス部を必要としない命令語は第16図に示すように 13 ビットからなるから、F8, F9 レジスタには最大八つの命令語を貯えておくことができる。DD



第16図 命令語の構成

および EE レジスタと呼ばれる 3 ビットの二つのレジスタから構成されるカウンタが CC に附属しており、次に実行する命令語が F8 または F9 のどこにあるかを指示している。ACR および DCR はそれぞれ先廻り制御装置ならびに演算装置のための命令レジスタである。この他、割込み信号のためのレジスタ、入出力装置を制御したり、磁気ドラム記憶装置と磁心記憶装置との間でブロック転送を行なうための制御装置などが附加している。

命令語は第16図に示したように 13 ビットから構成されているが、その内の F および C のビットで、その命令語がアドレス部 N をもつか否かを決定している。B ビットは指標レジスタ B_i を指示する。AM レジスタの内容が被演算数 OP の計算に用いられることがあるが、[AM] を使用するか否かは前に実行された命令によって決まる。+[AM] を用いる場合は Σ^+ , -[AM] の場合は Σ^- , また用いられない場合は Σ というフリップ・フロップがセットされる。命令語がアドレス部をもっている場合 (長い命令語) は次の 13 ビットでアドレス部を示すが、 Σ フリップ・フロップによって

$$OP = N + \Sigma AM$$

となる。命令語がアドレス部をもっていない場合 (短い命令語) は特別な命令を除いて

$$OP = \Sigma AM$$

と解釈される。

第2表に先廻り制御装置が実行する命令を挙げたが Σ フリップ・フロップは“— to next”形の命令でセットされる。B レジスタによる変更命令では OP は

$$C=1, 3 \text{ の場合} \quad OP \rightarrow AM, \Sigma=1$$

$$C=0, 2 \text{ の場合} \quad OP \rightarrow B_i, \Sigma=0$$

$$C=0, 1 \text{ の場合} \quad \text{短い命令} (OP = \Sigma AM)$$

$$C=2, 3 \text{ の場合} \quad \text{長い命令} (OP = N + \Sigma AM)$$

となる。

磁心記憶装置に飛越す命令語は常にアドレス部 N をもち、C ビットとによって次に実行する命令語の所在を指示する。この場合のアドレス計算には B レジスタは用いられない。F8, F9 の内部に飛越す命令は $B_i' = [B_i] + 1$ とし、その結果 $[B_i] \neq 0$ なら飛越す。

第2表 先廻り制御装置に関する命令

1. CLEAR SUBTRACT FROM MODIFIER (CSM) $-OP$.
2. SUBTRACT FROM MODIFIER (SFM) $B_i - OP$
3. CLEAR ADD TO MODIFIER (CAM) OP
4. ADD TO MODIFIER (ADM) $B_i + OP$
5. EXCLUSIVE OR MODIFIER (EOM) $b_{ij} \oplus OP$
6. AND MODIFIER (ANM) $b_{ij} \cdot OP_j$
7. OR MODIFIER (ORM) $b_{ij} \vee OP_j$
8. CIRCULAR RIGHT SHIFT MODIFIER (CRM)
Circular shift of B_i by $((OP) \bmod 16) \bmod 13$ places.
9. NEGATE AND "AND" WITH MODIFIER (NAM)
 $\overline{OP} \cdot B_i$
10. NEGATE AND "OR" WITH MODIFIER (NOM)
 $\overline{OP} \vee B_i$
11. EQUIVALENT WITH MODIFIER (EQM) $OP = B_i$
12. CLEAR NEGATE MODIFIER (CNM) $\overline{OP}$
13. SUBTRACT FROM NEXT (SFN): Address to AM , Σ^- is set.
14. ADD TO NEXT (ATN): Address to AM , Σ^+ is set.
15. COUNT AND JUMP (CAJ): $B_i' = B_i + 1$; jump if $B_i' \neq 0$.
16. JUMP IF POSITIVE MODIFIER (JPM).
17. JUMP IF NEGATIVE MODIFIER (JNM).
18. JUMP IF ZERO MODIFIER (JZM).
19. JUMP IF NON-ZERO MODIFIER (JUM): In 2 to 5, the modifier tested is B_i , which is negative if the leftmost bit is one.
20. JUMP ON DC CONDITION (JDC): B is decoded to indicate the condition accumulator positive, negative, zero, non-zero, overflowed, or jump unconditionally.
21. ENTER SUBROUTINE (JBS): $B_i' = (CC + 1)$; then jump. Since the order is long, and may therefore span word boundaries, it should be noted that CC is the address of the memory word containing N .
22. JUMP TO LEFT HAND ORDER (JLH): Jump to left control group of address formed by the normal rules, that is, the rules for the address in an Arithmetic Control order.
23. Count AND JUMP TO LAST (CJL).
24. COUNT AND JUMP TO FIRST (CJF).
25. COUNT AND JUMP TO SECOND (CJS).
26. OR TO B (ORB): Normal address construction is done. The 4 rightmost bits are combined by the logical OR function with the B field of the next order.
27. LOAD IN (LIN): Normal address construction is done. The address is placed in the 13 rightmost bits of IN , add $INFL$ set. This order should precede the DC order store with exponent equal (SEQ).
28. LOAD FAST REGISTER (LFR): The core memory word addressed is placed in F_8 .
29. STORE FAST REGISTER (SFR): The word in F_8 is placed in core memory location addressed. For LFR and SFR, $address = \Sigma AM$ if $c = 3$; $= N + \Sigma AM$ if $c \neq 3$. $2 \leq B \leq 7$ is the allowed range for LFR: SFR may have $B = 0$ also.
30. LOAD MODIFIER (LDM): This order is always long. Core word $(N + \Sigma AM)$ is fetched, and the quarter of it aligned with B_i is gated into that modifier.

31. STORE MODIFIER (STM): See LDM. Execution of this order is straight-forward due to provision in core memory to write selectively into quarter words.
32. PREPARE INPUT/OUTPUT DEVICE (PID or POD): Normal address construction. The rightmost 8 bits of the address, and a bit distinguishing PID from POD, are sent to the I/O channel addressed by the leftmost 5 bits of the address.
33. INITIATE BLOCK TRANSFER (IBT): Normal address construction. The address is entered in the interplay address list as the first core memory location to be used in a block transfer by the channel last named by a PID or POD order. The transfer is then started.
34. ADD SPECIAL REGISTER TO NEXT (ASN): This order is always short. $AM' = \Sigma AM$ plus the contents of one of up to sixty-four 13 bit special registers, the register being addressed by the 6 bit number BC . Σ^+ is set.
35. SUBTRACT SPECIAL REGISTER TO NEXT (SSR).
36. STORE IN SPECIAL REGISTER (SSR): ΣAM is placed in special register BC . Σ is reset.
37. BUSY BLOCK FLIPFLOP (BBF): Normal address construction is done. The core memory block lockout flipflop addressed by the leftmost 5 bits of the address is turned on.
38. FREE BLOCK FLIPFLOP (FBF): See BBF.

アドレス計算に際しては、 C ビットは次のような作用をする。

$C=0$ の場合 $N' = [B_i] + \Sigma AM$ ($0 \leq B \leq 15$)

$C=1$ の場合 (i) と同じ、さらに $B_i' = [B_i] + 1$

$C=2$ の場合 $N' = [B_i] + N + \Sigma AM$

$C=3$ の場合

$B < 8$ なら $[F(B)] \rightarrow F1$

$B = 8$ なら $N' = N + \Sigma AM$

$B = 9$ なら $N' = N + \Sigma AM$ が浮動小数点の整数として $F1$ に移される。

$B = 10$ なら $N' = N + \Sigma AM$ が固定小数点の数値として $F1$ に移される。

$11 \leq B \leq 14$ なら 無操作

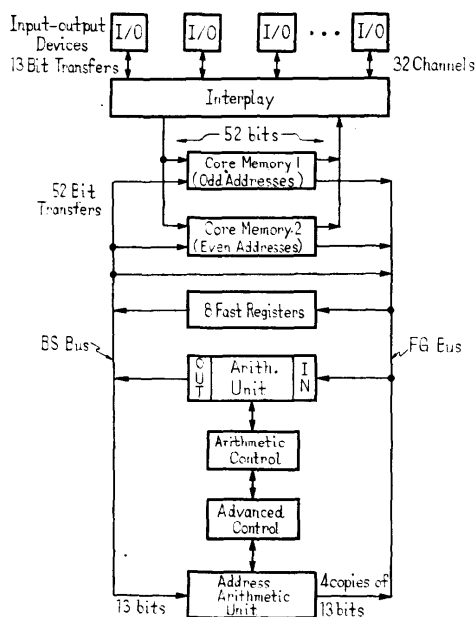
$B = 15$ なら 浮動小数点の数値零が $F1$ に入る。

先廻り制御装置は CC と DD レジスタに新しい命令語のアドレスがセットされると、直ちに磁心記憶装置のその記憶場所から命令語を讀出して $F8, F9$ に貯える。ついで実行すべき命令を ACR に移し、実行を開始する。命令が演算装置に関するものであれば、その命令語は ACR から DCR に移され、その実行を演算制御装置に依頼して次の命令の実行に移る。この場合、被演算数が必要であれば、先廻り制御装置が上に述べたアドレス計算をした後 $F1$ にそれを取り出し、

M, EM レジスタに移し終るまで, DCR はその実行を留めておく。ACR が演算の結果を条件とする飛越し命令を解読した場合は, 演算が終了するのを待って実行に移る。

サブルーチンへの飛越しの場合は $B_i' = [CC] + 1$ として飛越す。命令が“Store”であれば WA にアドレスを入れて次の命令に移る。“Store”に関する制御装置は, 演算が終了のを待ってその結果を FO に移す。その後磁心記憶装置が使える時に FO を“Store”する。この場合, もし WA が前の命令のアドレスを貯えている場合には, 先廻り制御は前の命令の実行が終了のを待つ。“Store”命令が実行されていない場合には [WA] と同一番地からの読み出しは許さない。

磁心記憶装置に関するブロック転送が行なわれる場合には, そのブロック内のアドレスをもつ命令の実行は中止されるが, その他の命令はブロック転送と並行して実行される。割込みが起った場合にはまずその原因を示すフリップ・フロップがオンになる。この時先廻り制御が実行しているプログラムは中断され, 必要な情報は割込み用の特別なレジスタに移され保存される。ついで割込みの原因に応じた特定なプログラムへの飛越しが実行され, 割込みの処理を行なう。



第 17 図 New Illinois Computer の
ブロック図

以上簡単に述べたように, 先廻り制御装置は効果的に演算装置および記憶装置を働かせ, 計算機の運転能力をよくする重要な役割を果たすものであるから, その動作はかなり高速のものが要求される。第 17 図は計算機全体の関係をブロック図で示したものである。

本年 3 月現在 New Illinois Computer は主演算装置と磁心記憶装置の設計, 製作が終り, 最終調整も順調に進行している。演算および先廻り制御装置の全ても今夏までには終了する予定である。調整の段階で実験的に得られた結果も良好で, 計算機全体の最終調整も本年末までには完了するものと思われる。

最後に, イリノイ大学でお世話になった同大学電子計算機研究所長 Dr. A.H. Taub, 直接御指導をいただいた Dr. J.E. Robertson, Dr. D.E. Muller, Dr. D.B. Gillies, Dr. G. Metze ならびに大学院学生の同僚に謝意を表し, 留学の機会を与えられた電気試験所長, 和田部長, 高橋元課長 (日立製作所に転勤), 西野課長および電子計算機研究室の諸氏にお礼申上げる。

参考文献

主なものを挙げる。

- 1) On the Design of a Very High-Speed Computer, Digital Computer Laboratory Report No. 80, (April 1958)
- 2) J.E. Robertson; Theory of Computer Arithmetic Employed in the Design of the New Illinois Computer at the University of Illinois, Digital Computer Laboratory File No. 319, (June 1960)
- 3) D.B. Gillies; Design of a Very High-Speed Scientific Computer, Digital Computer Laboratory File No. 376, (June 1961)
- 4) Advanced Control, R.R. Shively; Digital Computer Laboratory File No. 395 (Aug. 1961)
- 5) D.B. Gillies; A Description of the Operation of Delayed Control in Terms of Its Flow-Charts, Digital Computer Laboratory File No. 397, (Aug. 1961)
- 6) R.E. Miller, R.E. Swartwout, D.B. Gillies, J.E. Robertson; Introduction to Speed Independent Circuits, Proc. of the A.I.E.E., Second Annual Symposium on Switching Circuit Theory and, Logical Design pp. 87~110, (Sept. 1961)
- 7) D.E. Muller; New Illinois Computer, 情報処理, Vol. 2, No. 6, pp. 305~313 (11月, 1961)

(昭和 37 年 7 月 4 日受付)