リレーによる演算回路

和田 英一*

概要リレーといえば, 鉄道の信号機と転轍機を総括制御する継電連動装置や電話のステップバイステップ交換機で多用されていた他, 機械式から電子式へ進化の途中に計算機がリレーで作られた時代があった. かなりのリレー計算機がアメリカ, ドイツなど世界各地で製造されていた. 日本では電気試験所の ETL Mark II(1955 年) や富士通の FACOM 128A (1956 年) が大型リレー計算機の代表であった. 2017 年度の情報処理技術遺産に認定された自己相関係数計算機 (1954 年) のような単一機能のものも少なからずあり, Shannon の迷路探索機もリレーで作られていた. リレーは今も使われているが, 計算機用としてはカシオ計算機の AL-1 型 (1962 年) くらいを最後に殆ど姿を消した. いまではリレー計算機の構造を理解している人は皆無に近いと思われる. リレーで計算する代表的な回路は電子回路よりはるかに分りやすく面白いのでその辺りを説明したい.

はじめに

リレーは鉄道や電話など, 巨大なシステムを論理的に制御する基本装置として多用されていた一時代があった. 信号を赤にしてからでないと, 転轍器を操作できない. 列車が信号機の近くにいれば, 信号は変えられない. 電話の呼出し信号が来たら, 空いている交換器を探し始める. お話し中は, その通信回路を保持する. こういう因果関係をシステムに組み込むには, リレーを使うのが有効であった.

機械式から電子式へ進化の途中に計算機がリレーで作られた時代もあった. かなりのリレー計算機がアメリカ, ドイツなど世界各地で製造されていた.

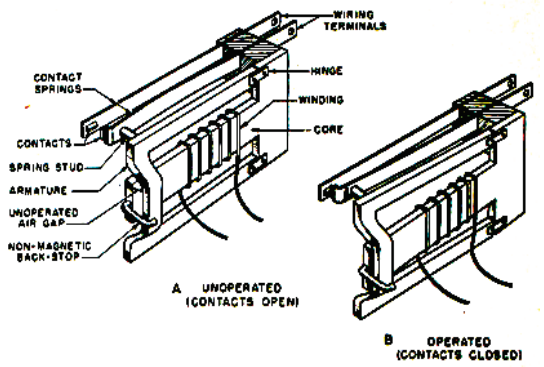
日本では電気試験所の ETL Mark I や Mark II や富士通の FACOM 128A や 138 が大型リレー計算機の代表であった. 富士通のリレー計算機は今でも動態保存されている.

自己相関係数計算機のような単一機能のものも, Shannon の迷路探索機もリレーで作られていた. リレーは今も使われているが, 計算機用として使われていない.

リレーでの計算は電子回路よりはるかに分りやすく面白いのでそれを説明する.

なお, この話はこの後にある富士通のリレー計算機の発表の前座である.

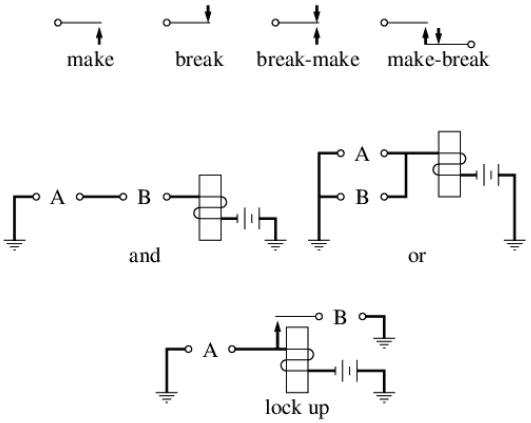
リレー回路の基本



左は後述の Keister の本から借用したリレーの原理図である. それぞれの図の中央の core(鉄芯) に winding(コイル, 巻線) があり, また hinge(蝶番) で可動な armature(可動鉄片, 元々は武器, 甲冑の意で, 素人の amateur ではない) が接点のバネで押し戻されて装着されている (左図). コイルに電流が流れると, 可動鉄片が鉄芯に引き寄せられ, 二つの接点が接触する (右図).

* IIJ 技術研究所
ew@iij.ad.jp

こういう絵をいちいち描くわけにはいかないので, 通常は右のような回路図にする. 最上段の図で, 可動鉄片はコイルに電流が流れていない時, 磁化されていない時の位置に描く. 鉄道用語を借りれば, これを定位という. make 接点は電流が流れると接点が閉じる. break 接点は定位では閉じていて, 電流が流れると開く. break-make と make-break は切替え接点 (トランスファーともいう) で, break-make では左からの電流が, 定位では上の接点へ接続し, 電流が流れると下の接点へつながる. その間に, いずれにもつながらない瞬間がある. break してから make する.



make-break では右からの電流が, 上の接点に接続しているが, 電流が流れると左からの鉄片が下へ動き, 右からの鉄片を下へ押して, それと上の接点を離し, 左右の鉄片が接続する. こちらはまず make し, その後 break するので make-break という.

計算機屋かく戦えり

リレー計算機は過去のものだから, それに関った人の話は貴重である. 「計算機屋かく戦えり」には, 中嶋章と塩川新助の章がある.

- 榛澤正男, シャノン以前にコンピュータの基礎理論を発信した日本人
  中嶋章が 1936 年に「継電器回路に於ける単部品路の等価変換の理論 (其の一)」を発表
  Claude Shannon の A Symbolic Analysis of Relay and Switching Circuits は 1938 年
- 岸本行雄, 二進法によるコンピュータを考察した塩川新助
  1934 年に二進法リレー回路を開発, 35 年に特許
  1939 年から 40 年頃二進法 10 桁の四則演算, 十進法二進法変換機能を持つリレー計算機を完成

このうち, 私は塩川さんには会ったことがある. 永田町の電気試験所に Mark II を見学にいったとき, 塩川さんが来ておられた. もう 1 人の中嶋章という人は, 次項の Shannon との先陣争いで知られている.

Shannon の論文

C.E.Shannon は 1938 年に MIT の修士論文としてリレー回路の研究を発表した. この A Symbolic Analysis of Relay and Switching Circuits は有名になり, 今でもインターネットで探すとあちこちに置いてあるのが見付かる.
これについては数年前のツィッターで

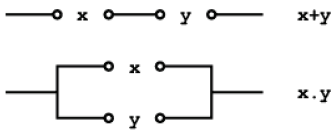
ryugo hayano @hayano 2012-12-08 07:33:09
(科学者列伝) 1864 年の今日 12/8 George Boole が亡くなった. 49 歳. ブール代数の人. 1937 年に当時 MIT の学生だったシャノンが, ブール代数を用い, あらゆる論理演算がスイッチ回路で実行できることを示す (史上最も有名な) 修士論文を書いた.

Eiiti Wada @eiitiwada 2012-12-10 20:22:11
早野君は中島章 http://t.co/9Kk0K4CD を知らぬらしい. @hayano 1937 年に当時 MIT の学生だったシャノンが, ブール代数を用い, あらゆる論理演算がスイッチ回路で実行できることを示す (史上最も有名な) 修

士論文を書いた.

ryugo hayano @hayano 2012-12-10 20:49:53
和田先生ごめんなさい！ @eiitiwada: 早野君は中島章 http://t.co/x6h6NBdA を知らぬらしい. @僕 1937 年に当時 MIT の学生だったシャノンが, ブール代数を用い, あらゆる論理演算がスイッチ回路で実行できることを示す（史上最も有名な）修士論文を書いた.

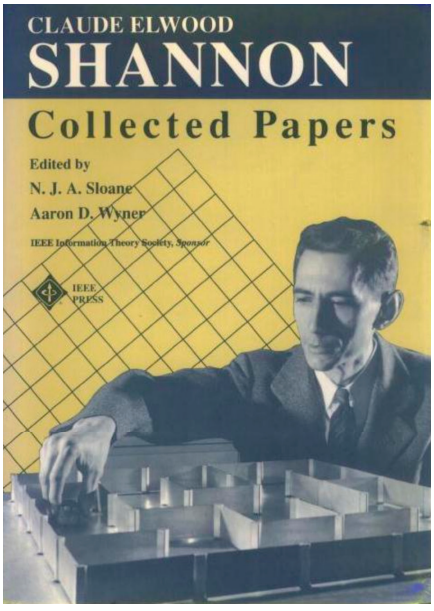
ryugo hayano @hayano 2012-12-10 20:51:49
(和田英一先生に出られちゃ〜しかたがね〜な〜)
というやりとりがあった.
中島論文は Shannon のに比べて簡単には見付からないが, 山田による解説がある [0].
一方, Shannon の論文は, 今の時代に読むと, 当り前のように思えることだが, 当時は最先端の仕事であったであろう.
Shannon は通信理論を発表したころは, ベル研にいたが, その前は MIT にて, Vannevar Bush が開発した微分解析機のグループにいたらしい. ベル研から MIT に戻ってきていたので, 私が 1973 年から 74 年に MIT にいたとき, 電気工学科の建物には Shannon の教授室があり, その前を通るときは足が震えた.
Shannon の論文では, 接点 x をヒンドランスと考え, 閉回路を 0, 開回路を 1 とする. 直列では x, y ともに 0 の時 0 になり, 並列では x, y のいずれか 0 の時 0 になるから, 直列は x+y, 並列は x.y になる. これは Keister の本でも同じである.

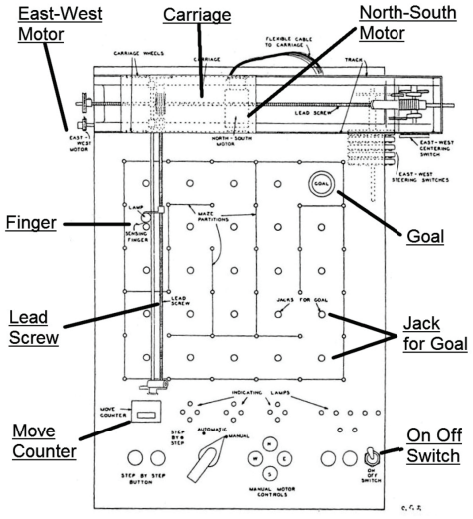


最近は閉回路を真, 開回路を偽とするのが多く, 直列は x∧y, 並列を x∨y とする.

Shannon の迷路探索機

Shannon は論文の後, リレーを使った迷路探索機を作ったので有名になった.



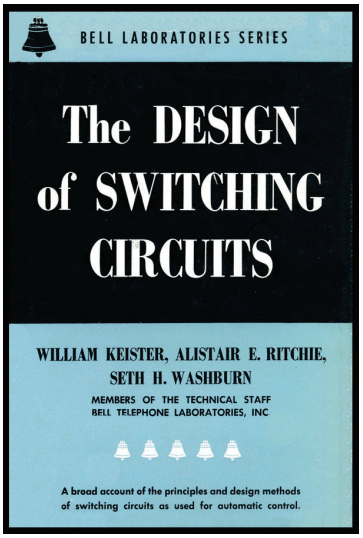


上の左は 2 号機で, 盤面の下から制御されるネズミが迷路の中を動く. 右が 1 号機で, リレーで制御される finger が Goal へ向って進む.

Shannon のネズミについては, 「A Computer Perspective」[1] に記述がある.

> シャノンは, 複雑に入り組んだ問題—電話交換機—を研究するために, 迷路を解くネズミを作った. このネズミのように, 電話呼出しは最短の可能な経路を通って相手先に到達しなければならない.
> 彼は, 彼の作ったネズミについて, 「試行錯誤の方法で迷路を解くことができ, 答えを記憶し, また状況が変ってその答えがもはや適当でなくなるとそれを忘れることができる」と説明している

Keister の本



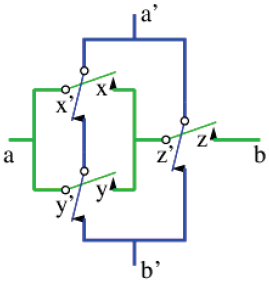
リレー回路の詳細な説明は, W. Keister 他による The Design of Switching Circuits にある.
この本は元々戦後まもなくにベル研で社内教育用に書かれた. それを基礎として 1950 年に MIT の院生のための講義に使われたものが本になった.
私が東大理学部の院生の頃, 工学部の磯部教授の自動制御の講義を聞いたら, リレーの話が多く, 後にその元が Keister のこの本であることが分った.
後述の十進法加算器は, この本にある話の紹介である.

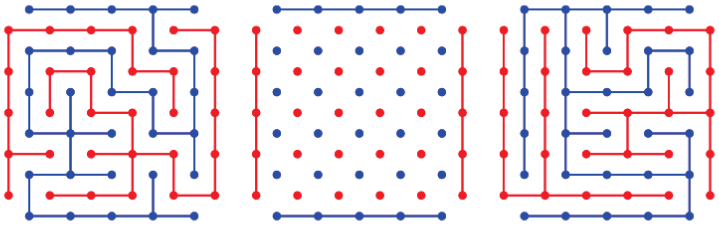
Boolean Algebra

Shakespeare の Hamlet, 1 幕 3 場で, Polonius が英国への留学に出発する息子の Laertes に与える訓戒の一節には, 真 (true) と偽 (false) が含まれていて, 計算機屋には興味深い.

ポローニアス	Polonius:
なにより肝心なのは, 自己に忠実であれということだ,	This above all; to thine ownself be *true*:
そうすれば, 夜が昼につづくように間違いなく	Ant it must follow, as the night the day,
他人にたいしても忠実にならざるをえまい.	Thou canst not then be *false* to any man.
ハムレット 1 幕 3 場 (小田島訳)	Hamlet I,iii

リレー屋は Switching Algebra というが, リレー回路の設計の基礎は Boole 代数である.
$x \wedge y = y \wedge x$, $x \vee y = y \vee x$
$x \wedge x = x$, $x \vee x = x$
$x \wedge \overline{x} = 0$, $x \vee \overline{x} = 1$
$(x \wedge y) \vee z = (x \vee z) \wedge (y \vee z)$, $(x \vee y) \wedge z = (x \wedge z) \vee (y \wedge z)$ 分配則
...
のような式が並ぶ. 特に重要なのが De Morgan's laws $\overline{x \wedge y} = \overline{x} \vee \overline{y}$ である. 下の左の図で, 緑の線は左右に a, b を結んでいる. その間に, x と y が並列に入ったものと, z が直列になっている. 一方青い線は, 上下の a', b' を結んでいる. その間に x' と y' が直列になったものと, z' が並列になっている. x', y', z' は $\overline{x}, \overline{y}, \overline{z}$ なので, De Morgan's law を示している.
右の 3 つの図は, Shannon のスイッチングゲームというもので, 中央のようなボードに, 赤のプレヤーは赤点同士, 青のプレヤーは青点同士をつなげていき, 両端の赤線同士, 青線同士を結べば勝つというものである.





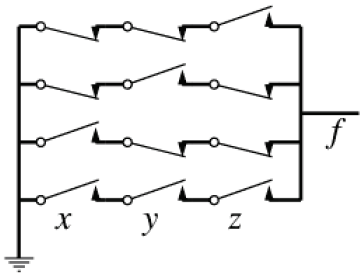
Shannon Switching Game

$ab = (x \vee y) \wedge z$; $a'b' = (x' \wedge y') \vee z'$

リレー回路の設計

論理式が与えられたとき, それを実現するリレー回路を設計する基本方針は, 主加法標準形である.

x	y	z	f	$\wedge xyz$	$\wedge x\overline{y}\overline{z}$	$\wedge \overline{x}y\overline{z}$	$\wedge \overline{x}\overline{y}z$
0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	1
0	1	0	1	0	0	1	0
0	1	1	0	0	0	0	0
1	0	0	1	0	1	0	0
1	0	1	0	0	0	0	0
1	1	0	0	0	0	0	0
1	1	1	1	1	0	0	0



上の表が変数 x, y, z の関数 f で, f は 3 つの二進数の和である.
この表を眺めて, f が 1 である行の, x, y, z 条件を見ると, 2 行目は 0,0,1 だから x が偽, y も偽, z が真のとき f が真なので, $\overline{x} \wedge \overline{y} \wedge z$ と書く. 表の欄では, $\wedge$ を前置にしてある.
こうして 4 つの 1 に対応する, $\wedge$ の式, 最小項 (minterm) を論理和 $\vee$ で結ぶと, 主加法標準形 (full disjunctive normal form) が出来る. 「主」というのは, いずれの最小項にも全ての変数が現れるからだ.

二進化十進法

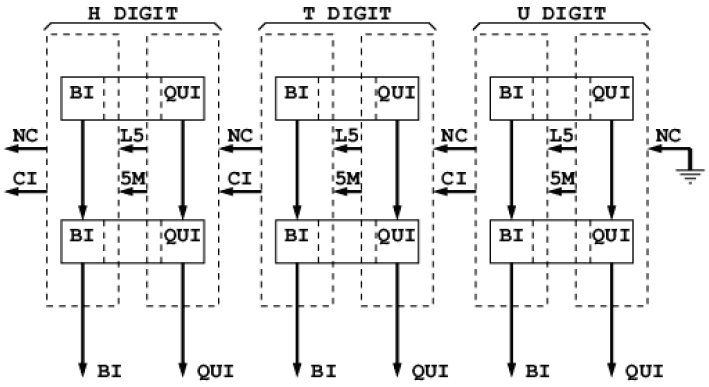
最近の計算機内部は殆どが二進法になっているが, 計算機の揺籃期には十進法の計算機も少なからず存在した. 計算のデータや答は十進法でないと不便だから, 計算を二進で行うと, 出入り口で十進と二進の変換をすることになる. 計算が大量だと, 変換の手間も無視できるが, ちょっとした計算をするなら, 十進のまま計算するのが得だからという理由である.
しかし計算の素子はオンオフ状態を持つ二進素子が一般的なので, 十進法で計算するといっても, 十進数を二進素子で表現する必要がある. それを二進化十進法という. 時計で六十進法を十進数で表現しているのは, 十進化六十進法といえる.
下の表で, 左の二進法は, 通常の二進法の 0 から 9 までを使う一番普通のものである. しかしこれは加算の結果が十を超えたかどうか, 毎回 1010 と比較しなければならない.
そこで考えられたのが, 右隣の三余り符号である. 二進法の数に 3 ずつ足したものを使う. メリットは繰上げ検出が容易なことだ. 十進法の加算で繰上げが出るときには, この加算でも繰上げが出る. これは便利だが, その代り加算の結果の補正がいる. 繰上げの出ないときは, 3 を引き, 出たときは 3 を足す.
次は 01247 コードというもので, 5 ビットを使う. 5 ビットのうち, 1 がちょうど 2 個ある組合せは 10 個あって, それを 0 から 9 に対応させる. 5 個のビットに 0,1,2,4,7 と名前をつけると, 0 以外は 1 である 2 つのビット名の和がその数になるという仕掛けである. しかしこれは計算には向いていない.

	二進法				三余り符号				01247 コード					二五進法						
									0	1	2	4	7	00	5	0	1	2	3	4
0	0	0	0	0	0	0	1	1	0	0	0	1	1	1	0	1	0	0	0	0
1	0	0	0	1	0	1	0	0	1	1	0	0	0	1	0	0	1	0	0	0
2	0	0	1	0	0	1	0	1	1	0	1	0	0	1	0	0	0	1	0	0
3	0	0	1	1	0	1	1	0	0	1	1	0	0	1	0	0	0	0	1	0
4	0	1	0	0	0	1	1	1	1	0	0	1	0	1	0	0	0	0	0	1
5	0	1	0	1	1	0	0	0	0	1	0	1	0	0	1	1	0	0	0	0
6	0	1	1	0	1	0	0	1	0	0	1	1	0	0	1	0	1	0	0	0
7	0	1	1	1	1	0	1	0	1	0	0	0	1	0	1	0	0	1	0	0
8	1	0	0	0	1	0	1	1	0	1	0	0	1	0	1	0	0	0	1	0
9	1	0	0	1	1	1	0	0	0	0	1	0	1	0	1	0	0	0	0	1

最後は二五進法という. 0 から 9 を 5 で割った商 (0 か 1) と, 5 で割った剰余 (0,1,2,3,4) とであわらす. 7 ビットを用いるが, これも 2 out of 7 なので (1 out of 2 と 1 out of 5), エラー検出の機能を持つ.

二五進法のリレー加算回路

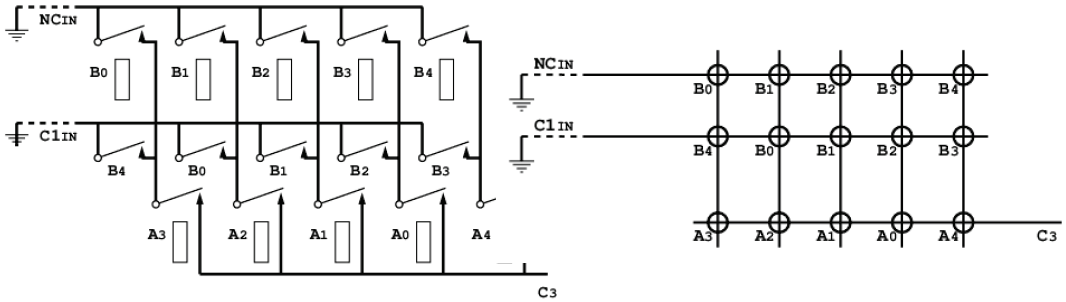
ここからが Keister の本にある, 二五進法のリレー加算回路である.



二五進法のリレー加算回路では十進法の各桁が, QUI という五進法の加算部分と BI という二進法の加算部分に別れる. 十進法の桁の間に繰上げがある (C1 が接地), 繰上げがない (NC が接地) の情報があるように, 五進法から二進法へも繰上げの情報 (5 以上であった (5M が接地); 5 未満であった (L5 が接地)) が伝えられる. 3 個ある十進の桁は, 右から一の桁 (U), 十の桁 (T), 百の桁 (H) であり, 一の桁へ右から入る繰上げ情報は NC の方が接地されていて, 繰上げがないことを示している.

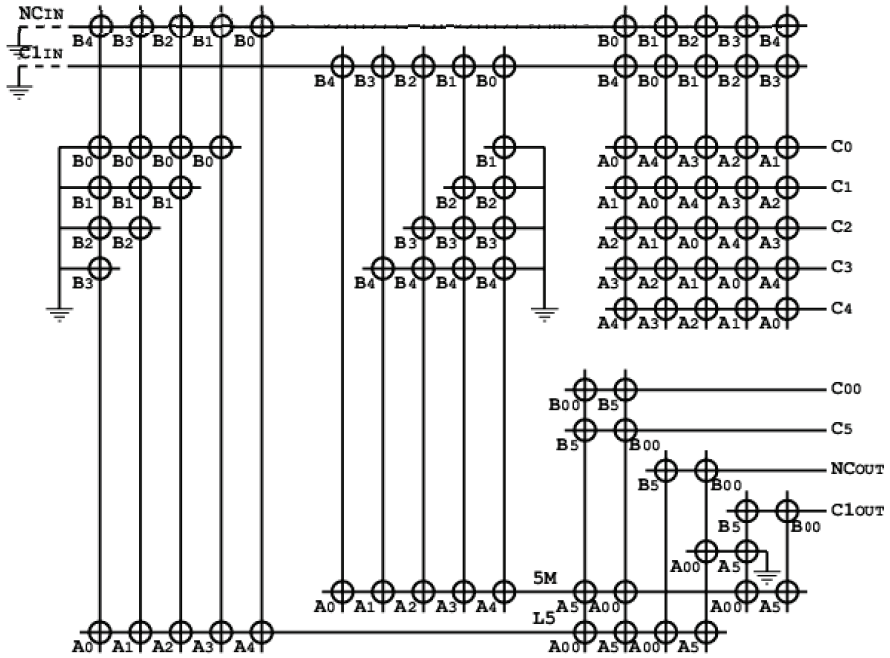
各桁からの出力は, BI が 2 本 (0 か 5) と QUI が 5 本 (0 から 4 で, そのうち 1 本が接地) ある.

QUI の加算回路のうち C3(3 の場合に接地する) を出力するのは次の図左のようになっている.

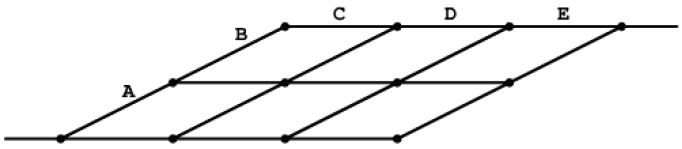


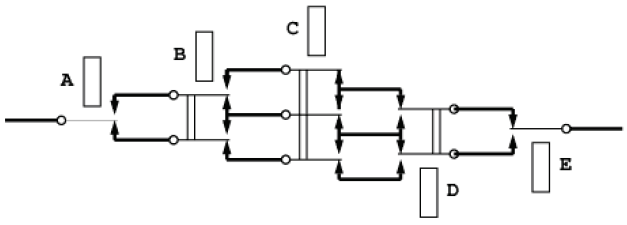
下からの繰上げがないと, NC_{IN} が接地し, B0 から B4 のいずれかが接地して, 対応する上段のスイッチが閉じる. 同時に A のスイッチのいずれかも閉じるが, B の番号と A の番号の和が 3(mod 5) のときに C3 が接地する. 下からの繰上げがあると, $C1_{IN}$ の方が接地して, 入力が 1 増えることになるので, B の下段のスイッチと

の和が 2(mod 5) のときに接地する.
いちいちこういう回路を描くのは面倒なので, スイッチがすべて make であることを了解し, 右のように描く.



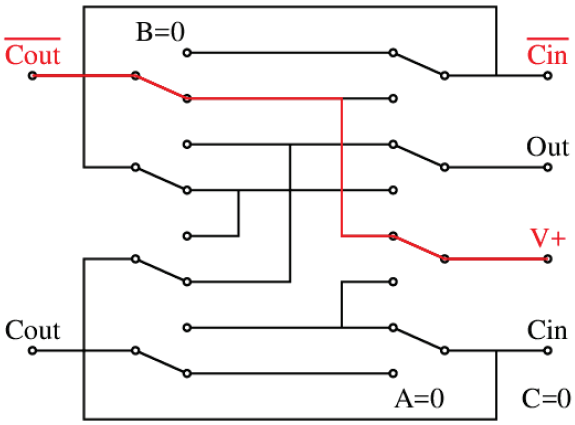
右の中程の接点が 5×5 に並ぶのが五進の加算回路で, C_3 が直前の図に対応する. その下が二進の加算部分になる. 中央の三角に並ぶ部分が五進から二進への繰上げ信号, 5M と L5 を作る部分である. この桁からの出力は図の右端からで, 五進が C_0 から C_4, 二進が C_{00} と C_5, 上の桁への繰上げが NC_{OUT} と $C1_{OUT}$ である.
make 接点の個数を見ると, A_0 から A_4 が各 7, B_0 から B_4 が各 8, A_{00} と A_5 が夫々 5, B_{00} と B_5 が夫々 4 で合計 93 個である. 8 接点ものリレーは動かすのも大変そうだ.
リレーの加算回路が凄いのは, 入力が揃った途端にキャリーが伝わり, 結果が得られることである.
二五進法の加算回路は, エラー検出機能を持つが, 2 out of 5 の検出回路をリレーで組んだのが次の図である. この話は後述の Shannon の回路と関係がある.





Zuse のリレー加算器

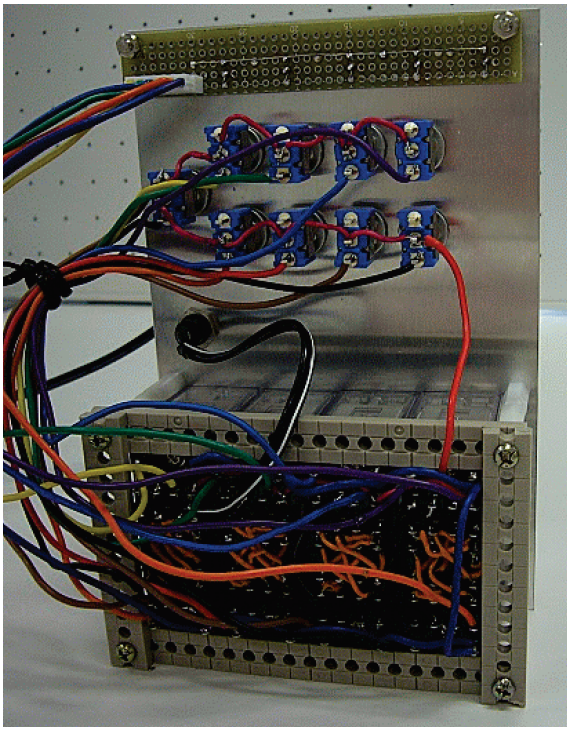
Konrad Zuse がリレー計算機 Z3 を設計したときに考案した二進 1 桁の全加算器で, dual-rail-carry full adder として知られている.

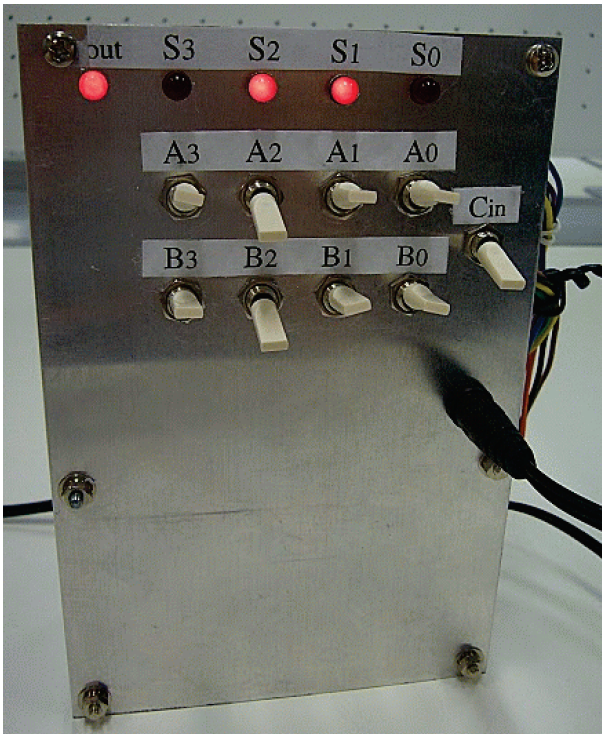


左がその加算回路で, この桁の入力 a, b(0 または 1) により, 右側と左側の 4 回路のトランスファースイッチが動く. 図は a, b が 0, つまりコイルに電流が流れていない定位の状況を描く. 下からの繰上げは右端の C_{IN} $\overline{C_{IN}}$ のいずれかに＋の電圧がかかっていることで判定する. 上への繰上げも同様に C_{OUT} と $\overline{C_{OUT}}$ のいずれかに電圧のかかる状態で示す他, この桁の出力は Out の端末に電圧がかかることで 1 であることを知る.

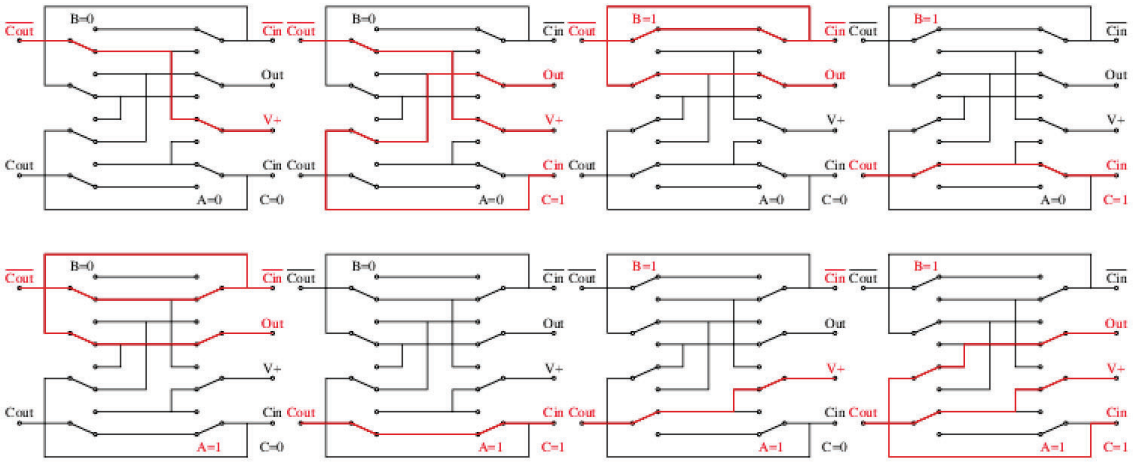
V+ の端末は常に電圧がかかっていて, true(真) になっている. 赤線の回路は, $\overline{C_{OUT}} = \text{true} \wedge \bar{a} \wedge \bar{b}$ になっている.

$Cout = (a \wedge b) \vee (((a \wedge \bar{b}) \vee (\bar{a} \wedge b)) \wedge Cin)$
$\overline{Cout} = (\bar{a} \wedge \bar{b}) \vee (((\bar{a} \wedge b) \vee (a \wedge \bar{b})) \wedge \overline{Cin})$
$Out = (\bar{a} \wedge b \wedge \overline{Cin}) \vee (a \wedge \bar{b} \wedge \overline{Cin}) \vee (\bar{a} \wedge \bar{b} \wedge Cin) \vee (a \wedge b \wedge Cin)$
何年か前に私が作った 4 ビットの加算器はこの写真のようなものであった.

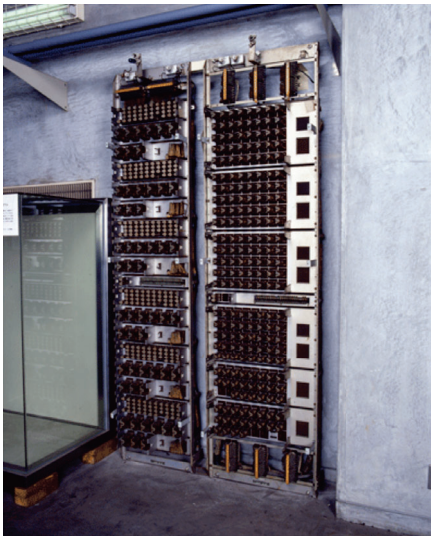




a, b, c が 0, 1 のすべての状態を示すと次のようになる.



電気試験所の ETL Mark II



永田町にあった電気試験所の電子部で大型のリレー計算機 ETL Mark II[2][3] が動き出したという話を聞き, 私は早速その見学に出掛け, 駒宮, 末包両氏から説明を受けた. 左の写真はその一部が科学博物館に展示されているものである.

内部は 2 進, 40 ビット, データ用の内部記憶容量 200 語, 使用継電器 22,253 個.

リレーによる 200 語の記憶装置を持ち 1,000 語まで拡張が可能.

詳細設計は研究室員のみで行い, 製造は富士通信機製造 (現在の富士通) に依頼.

入力装置のテープ読取機, テープせん孔機は新興製作所が制作.

1955 年 11 月に完成し, 約 10 年間電気試験所内外の計算に利用された. というのがその概要である.

ETL MkII の 1 ビット加算回路

ETL のリレー計算機は, 駒宮氏らが開発したリレーの回路式に基く設計で, 上に述べたリレー回路が, データが揃うと一瞬に出力で出るのとは一線を画す. すべての論理回路は表と裏の対の回路で出来ていて, 一段ずつ回路が合っていることを確認しながら演算が進行する方式であった.

ある桁の入力の 3 ビット, A_0, A_1, A_2 を足す回路はこう出来ていた.

$A_0 + A_1 + A_2 = 2d_1 + d_0$ (A_0, A_1, A_2 の 3 ビットを足す)

d_0 がこの桁の和, d_1 が上への繰上げである. d_0, d_1 の求め方は

$d_0 = A_0 \equiv (A_1 \equiv A_2)$

$d_1 = A_1A_2 \vee A_0(\overline{A_1}A_2 \vee A_1\overline{A_2})$

とする. $\equiv$ は等価の記号で, ETL 方式では $\wedge$ を書かない.

これでよい理由はこうである.

$d_0 = A_0\overline{A_1A_2} \vee \overline{A_0}A_1\overline{A_2} \vee \overline{A_0A_1}A_2 \vee A_0A_1A_2$ — A_0, A_1, A_2 に 1 が奇数個ある

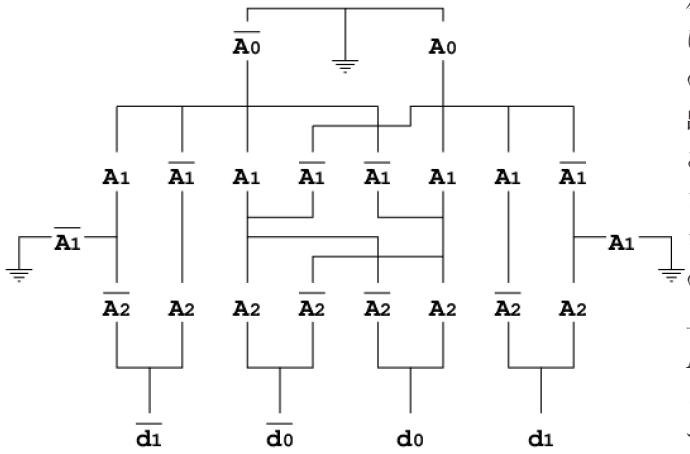
$= A_0(\overline{A_1A_2} \vee A_1A_1) \vee \overline{A_0}(A_1\overline{A_2} \vee \overline{A_1}A_2)$ — $A_0, \overline{A_0}$ を括り出す

$= A_0(A_1 \equiv A_2) \vee \overline{A_0}(\overline{A_1 \equiv A_2}) = A_0 \equiv (A_1 \equiv A_2)$ — $xy \vee \overline{xy} = x \equiv y$

$d_1 = \overline{A_0}A_1A_2 \vee A_0\overline{A_1}A_2 \vee A_0A_1\overline{A_2} \vee A_0A_1A_2$ — A_0, A_1, A_2 に 1 が 2 個以上ある

$= (\overline{A_0} \vee A_0)A_1A_2 \vee A_0(\overline{A_1}A_2 \vee A_1\overline{A_2})$ — A_1A_2, A_0 を括り出す

$= A_1A_2 \vee A_0(\overline{A_1}A_2 \vee A_1\overline{A_2})$ — $x \vee \overline{x} = 1$



左が 1 ビット分の加算回路である. A や $\overline{\text{A}}$ はすべて make スイッチで, A があれば, その裏回路 $\overline{\text{A}}$ のあるのが見える.

出力は d_0 と d_1 で, これにも裏の出力がある.

まず右の d_1 を見ると, すぐ上に A_2 があり, さらに右に折れて A_1 がある. これが上の d_1 の式の第 1 項 A_1A_2 の直列である. d_1 の上に 2 本に別れて登る線は, $\overline{\text{A}_1}\text{A}_2$ の直列と, $\text{A}_1\overline{\text{A}_2}$ の直列が並列になっている.

これは更に上の A_0 と直列になっている. 式でいえば, $A_0(\overline{A_1}A_2 \vee A_1\overline{A_2})$ のところだ.

次に d_1 から登る線を辿ると, すぐ上で 2 本に別れ, そのそれぞれが中段で 2 本に別れるから, 4 つの式らしいと分る.

一番右は $\text{A}_0\text{A}_1\text{A}_2$ の直列である. A_2 の上で左へ分岐した線は, $\overline{\text{A}_1\text{A}_0}$ を通過する.

d_1 から $\overline{\text{A}_2}$ を通った線は, $\overline{\text{A}_1}\text{A}_0$ と $\text{A}_0\overline{\text{A}_1}$ の並列線を通過し, この 4 本が並列, つまり $\vee$ で繋がり, d_1 の最上段の式になる.

Shannon の加算回路

全加算回路は, この桁の入力の 0,1 も, 繰上がってきた 0,1 も区別しないから, 入力の 1 の個数の関数と思ってよい. こういう関数をう General Symmetric Function という.

Shannon の加算回路はこういう発想から出発する.

General Symmetric Function は Knuth の The Art of Computer Programming の第 4A 巻の 7.1.1 に出ている. ちょっと引用する.

$S_{k_1,k_2,\ldots,k_r}(x_1, \ldots, x_n)$ is true if and only if νx is either k_1 or k_2 of ... or k_r. (TAOCP 7.1.1)

$S_{1,3,5}(v, w, x, y, z) = v \oplus w \oplus x \oplus y \oplus z$,

$S_{3,4,5} = \langle vwxyz \rangle$,

$S_{4,5} = \langle 00vwxyz \rangle$

という具合である. (νx は x を二進法にしたときの 1 の数.) $S_{1,3,5}(v, w, x, y, z)$ は 5 変数で, 1 が 1 個か 3 個か 5 個のとき, 値が 1 になるというので, 排他的論理和になる. $S_{3,4,5} = \langle vwxyz \rangle$ の $\langle, \rangle$ は中央値の記号だが, ここでは多数決と考えてよい. 最後の $S_{4,5} = \langle 00vwxyz \rangle$ は, 5 変数に「さくら」の 00 を入れた多数決で実現している.

TAOCP には, 4 変数と 5 変数の対称関数を実現する最良の論理関数の図がある. その S_2 を確かめているのが

次である.

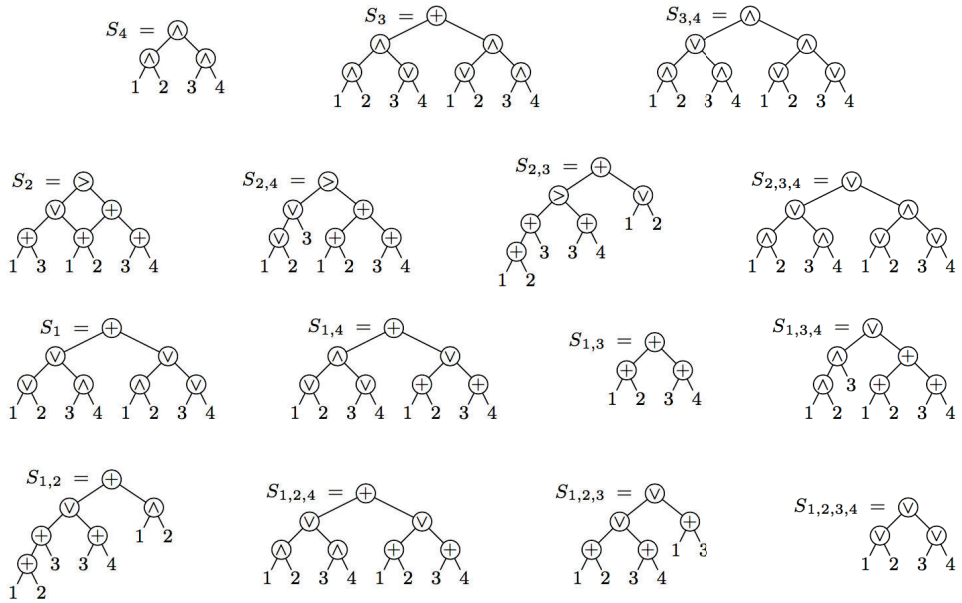
x_1 から x_4 で 4 変数のすべての組合せを用意する. 図に従って論理演算を実行する. 最後に得られ結果を見ると, 最初の変数にちょうど 2 個の 1 のあるときだけ, 1 であり, 正しく演算できることが分る. (1 の上に小さい ∘ がついている.)

$S_2(x_1, x_2, x_3, x_4)$

x_1	0000 0000 1̊1̊1̊1 1̊111
x_2	0000 1̊1̊1̊1 0000 1̊111
x_3	0011̊ 00̊11 00̊11 0011
x_4	0̊101̊ 0̊101 0̊101 0101
$x_5 = x_1 \oplus x_3$	0011 0011 1100 1100
$x_6 = x_1 \oplus x_2$	0000 1111 1111 0000
$x_7 = x_3 \oplus x_4$	0110 0110 0110 0110
$x_8 = x_5 \vee x_6$	0011 1111 1111 1100
$x_9 = x_6 \oplus x_7$	0110 1001 1001 0110
$x_{10} = x_8 \wedge \overline{x_9}$	0̊001̊ 0̊1̊1̊0 0̊1̊1̊0 1̊000

二五進法加算回路の最後にあった 2 out of 5 は, $S_2(A, B, C, D, E)$ であった.

4 変数の対称関数の絵はこれだ. $x < y = \overline{x} \wedge y$, $x > y = x \wedge \overline{y}$ が使われている.



さて Shannon の加算回路は次の式による.

c_{k+1}	c_k		c_{j+1}	c_j		c_2	c_1	
	a_k	$\cdots$	a_{j+1}	a_j	$\cdots$	a_2	a_1	a_0
	b_k	$\cdots$	b_{j+1}	b_j	$\cdots$	b_2	b_1	b_0
	s_k	$\cdots$	s_{j+1}	s_j	$\cdots$	s_2	s_1	s_0

a_j, b_j, c_j はこの桁への入力で, s_j はこの桁の出力である. 繰上げの出力 c_j もあるが, それは左隣の列の c_{j+1} で示す.

すると各桁の演算は, 対称関数を使ってこう書ける.

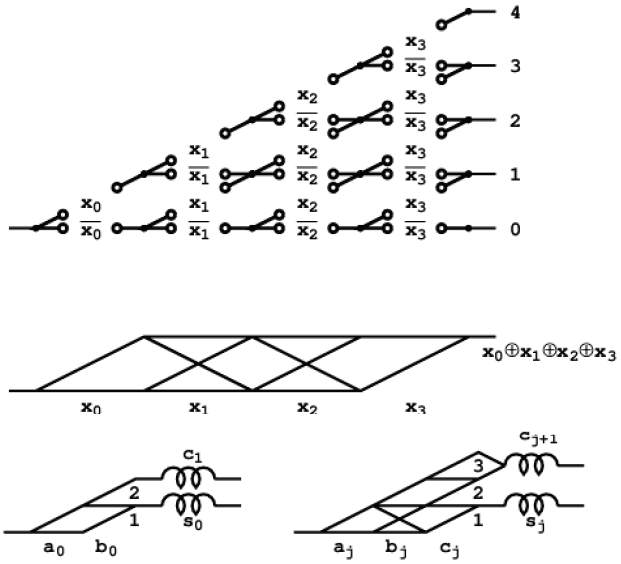
$s_0 = a_0\overline{b_0} \vee \overline{a_0}b_0 = S_1(a_0, b_0)$

$c_1 = a_0 b_0 = S_2(a_0, b_0)$

$s_j = S_{1,3}(a_j, b_j, c_j)$

$c_{j+1} = S_{2,3}(a_j, b_j, c_j)$

従って加算回路を実現するには, transfer スイッチ網で 0,1,2,..,n out of n 回路を作ることである.
右の上の図は 4 変数での回路である.
中の図は上の図を途中で折り返し, 1 と 3 の出力を同じ口から取り出すものである. これで $S_{1,3}(x_0, x_1, x_2, x_3)$ が出来る.
一番下が 1 桁分の加算回路で, まず右の図を見ると, 下の出口が中の図と同様の $s_j = S_{1,3}(a_j, b_j, c_j)$, 上の出口が $c_{j+1} = S_{2,3}(a_j, b_j, c_j)$ になっている.
左側は単に 0 の桁に繰上げの入力がないだけの図である.



おわりに

2018 年 10 月 28 日から 12 月 1 日まで, 東京理科大学近代科学資料館で, 企画展「パラメトロンとリレー計算機」〜日本発コンピュータの開発〜 が開催された. ここには上述の私のリレー加算器を始め, いくつものリレー計算機が展示された. 関連イベントには, 富士通川崎工場の Facom 138A 見学ツアーもあった.
リレーによる機械計算の理解が深まればよいと願っている.

参考文献

[0] 山田昭彦, スイッチング理論の原点を尋ねて, −シャノンに先駆けた中嶋章の研究を中心に−, 電子情報通信学会 基礎・境界ソサイエティ, Fundamental Review Vol.3,No.4 pp.9-17,2010 年 4 月, https://www.jstage.jst.go.jp/article/essfr/3/4/3_4_4_9/_pdf

[1] チャールズ イームズ, レイ イームズ, コンピュータ・パースペクティブ:計算機創造の軌跡 (ちくま学芸文庫) 2011.

[2] 駒宮安男, リレー計算機 ETL Mark I, Mark II, 情報処理 Vol.17,No.6.

[3] M.Goto,Y.Komamiya,R.Suekane,M.Takagi,S.Kuwabara,Theory and Structure of the Automatic Relay Computer E.T.L. Mark II, 電試研報 No.556(Sept.1956.)

討論

Q. 大座畑 スイッチング理論の仕事で中嶋と Shannon のどちらが早いかは, 研究ノートを見れば分るのではないか.
A. 80 年も前の話なので, そういうものはもうないのではないと思う.