計算機アーキテクチャ　71－11
（1988. 7. 7）

ニューラルネット向きプログラム可能論理演算装置について

PROGRAMMABLE LOGIC UNITS FOR THE NEURAL NET

吉岡　良雄
Yoshio　YOSHIOKA

岩手大学・工学部・情報工学科，盛岡市
Department of Computer Science, Faculty of Engineering,
Iwate University, Morioka-shi 020 Japan.

あらまし　従来から提案されている演算用のプログラム可能論理演算装置（ＰＬＵ）について，その考え方を説明している．そして，そのような考え方に基づいて，ニューラルネット向きのＰＬＵを提案している．このＰＬＵにおける神経回路にはアナログ神経回路を採用している．また，アソシアトロンとパーセプトロンが混在した多層のニューラルネットであるため，実際の神経（あるいは脳）に近い処理が期待できる．しかしながら，学習の方法などが今後の課題である．

Abstract　The programmable logic unit (PLU) has been proposed for the processing in memory type hardware in the previous paper.　On the base of this idea, this paper proposes the PLU for the neural net, whose element is an analogue type circuit.　Being constructed as the multi-layer type net combining the associatron and the perceptron, the proposed PLU will be expected the processing of the real neural (or brain).

１．まえがき

ノイマン型計算機は，高集積化技術の発達や応用プログラムの開発などによって急速に発展した．さらに，このタイプの計算機が急速に発展した要因の一つは，処理アルゴリズムが逐次的に進むので考え易いこと・汎用性が高いことなどから，算術演算ばかりではなく種々の方面に応用されたことが上げられる．そして，画像処理や信号処理などのデータ処理分野の拡大に対して，計算機の処理速度向上や並列処理技術の要求が増々高まってきた．最近では，大規模集積回路（ＬＳＩ）技術の発展や３次元ＬＳＩ技術の開発などによって，このような並列処理が自然に受け入れられるようになってきた．さらには，従来からある神経回路網（ニューラルネット）もＬＳＩ技術の発達によって容易に実現できるようになってきたので，この処理アルゴリズムが最近各所で急速に議論されるようになってきた[1]-[3]．このようなニューラルネットは，コネクションマシン[4]やプログラム可能論理演算装置（ＰＬＵ：Programmable Logic Unit）[5][6]などのような超並列処理マシンと考えることができる．ここで，ＰＬＵは中央処理装置（ＣＰＵ）からのプログラムによって演算回路をダイナミックに変えて，演算データの格納と結果データの読み出しだけで処理を進める装置である．そして従来の逐次型（ノイマン型）計算機のもとで，バック・エンド・プロセッサあるいはプロセッシング・イン・メモリとして動作する．

本報告では，このようなＰＬＵの処理概念を述べ，ニューラルネット向きのＰＬＵについて議論を進める．まず，第２章では，ＰＬＵの提案動機と演算用ＰＬＵシミュレータにおけるプログラム例を示して，ＰＬＵの処理概念について述べる．次に，第３章ではニューラルネットとしてのＰＬＵの構成や動作を説明する．

２．プログラム可能論理演算装置（ＰＬＵ）

本章では，プログラム可能論理演算装置（ＰＬＵ）の提案動機を示し，Hillisの提案している" Connection Machine" との違いについて述べる．次に，演算用ＰＬＵシミュレータのプログラム例を示して，ＰＬＵの処理概念について述べる．

２．１　ＰＬＵの提案動機

高速フーリエ変換装置やデジタルフィルタなどの処理装置は電子回路素子で構成され，素子のゲート遅延時間レベルで演算が可能な専用処理装置である．このため非常に高速である．しかし，このような装置は汎用性に欠け高価なものになる．一方，このような装置の処理を従来の逐次型のプロセッサ（ＣＰＵ）を用いて，逐次的なプログラムで実現することも可能である．この場合，融通性や汎用性は高いが高速性に欠ける．前者の専用装置では，一連の処理をハードウェア空間上に展開して処理を行っているため，処理の時間的なパイプライン型並列性および空間的なマルチプロセッサ型並列性が引き出され，処理の高速化が図れる．また，この処理を時間軸上に展開して逐次的に処理を進める方法が後者であり，ノイマン型の計算機はこれに属する．すなわち，時間を犠牲にしているといえる．従って，処理の高速化と汎用性を兼ね備えるためには，一連の処理をハードウェア空間上に展開して，かつこのハードウェアをプログラム可能にしなければならないことが分かる．プログラム可能論理演算装置（ＰＬＵ：Programmable Logic Unit ）は，このような理由から提案されたものである．さらに，このＰＬＵの処理方式は，トークンなどによって処理開始の同期をとるのではなく，非同期かつアナログ的に処理を進める静的な垂れ流し的データフロー処理である．

このようなＰＬＵを用いた計算機は，図１に示すように，メイン計算機からメモリと同じようにアクセスし，制御することができる．すなわち，ＰＬＵに演算データ群ｘ，処理プログラム（操作）群ｆとプログラム定数ａが書き込まれた後，素子のあるゲート遅延時間後にその処理結果群ｙを読み出す．従って，ＰＬＵはメインの計算機のメモリに処理や演算機能を持ったプロセッシング・イン・メモリ的な装置であるといえる．このようなＰＬＵの特徴を，この考えに似ている装置である″Connection Machine（ＣＭ）″と比較しながら示すならば以下のようになる．

(a).処理要素（処理セル）：ＰＬＵにおける各処理セルは，記憶部に相当するレジスタと演算を行う処理部からなっており，処理部は基本的にワイヤードロジックである．しかし，高い演算機能を必要とする場合，処理部にプロセッサを用いる．これに対して，ＣＭでは１ビット処理用のプロセッサと４Ｋビットのメモリからなるプロセッサ／メモリが１単位の処理セルである．

(b).処理セル間の通信機能：ＰＬＵの処理セル間の結合は，メインの計算機からの処理プログラムのマッピングによって構築される．従って，ＰＬＵでは基本的に処理セル間の通信機能を持たない．一方，ＣＭの処理セル間は，　Boolean n-cube　構造のパケット交換網で接続され，適応ルーチングによって動的なデータ通信が行われる．

(c).処理目的：ＰＬＵは主に高速演算用を目的としているが，処理アルゴリズムを考慮すれば人間の思考や神経の処理などに適応できる．一方，ＣＭは，主に LISP や Prolog などの知識処理を目指している．

このようにＰＬＵとＣＭを比較すると，非常に異なっていることが分かる．

２．２　ＰＬＵシミュレータ

まず，ＰＬＵシミュレータのＰＬＵ構造は，図２に示すものを取り上げる．このような構造は，コンパイラにおいて入力データ部（ＤＲ：Data Register ）への変数の割り当てや命令部（ＩＲ：Instruction Register）へのプログラムのマッピングが容易である．

つぎに，ＰＬＵシミュレータはマイクロプロセッサMC6809で作成し，共有メモリを介してHOSTコンピュータであるパーソナルコンピュータFM77AVのメモリ上に配置されている．すなわちパーソナルコンピュータFM77AVは，メモリ管理レジスタ(MMR) によって$00000～$3FFFFのメモリ空間をもっている．そしてメインＣＰＵ（MC6809）からは，　$0000～$FFFF の 64 KBのメモリ空間内において，４KB単位のウインドウの形でこれらのメモリ空間をアクセスすることができる．通常のプログラムは$30000～$3FFFFで実行し，ＰＬＵシミュレータは$20000～$2FFFFのメモリ空間内で図２に示すＰＬＵをシミュレートしている．メインＣＰＵのメモリ空間でのＰＬＵシミュレータのアドレスは，256 個のデータレジスタを $8000～$83FF に，1792 ステップの命令レジスタを $8400～$9FFF になっている．そして，このＰＬＵには表１に示す命令などがある．さらにメインの計算機が実行するプログラム空間は $6000～$7FFF の 2048 ステップであり，配列やメッサージデータなどのデータ領域は $A000～$BFF0 である．ここで，メインの計算機では表２に示す命令を実行する．

このようなＰＬＵシミュレータによって作成したプログラム例を図３と図４に示す．図３におけるＰＬＵでの演算は図５のようになり，並列に演算が行われていることが分かる．そして，この場合，ＰＬＵでの演算時間は演算時間がもっとも長くかかる除算の演算時間とＰＬＵ内のゲート遅延時間Δの和で与えられる．ただし，演算時間>>Δであることが必要である．従って，メイン計算機の″wait″命令での待ち時間はほぼこの和である．ま

た，図４のプログラム例のように，ＰＬＵではメイン計算機の処理に関係なくすべての演算が並列に行われているので，" if" 文の条件によってそれぞれに対応する演算を進めるよりもむしろすべての演算を行ってから結果を条件によって取り出す方がよい．

以上に示したプログラム例から，ＰＬＵを用いた計算機の処理には次のことが要求される．

(1)．ループ文のように，メイン計算機の介入が多くなるようなマッピングは避け，ループ文を展開してマッピングする必要がある．
(2)．メイン計算機は，ＰＬＵに演算データを格納してから結果を得るまで待たなければならないので，メイン計算機のスループットを上げるためにマルチプログラミング手法を導入する必要がある．なおこの待ち時間はコンパイル時に並列演算や演算時間から計算して" wait" 命令のオペランド" time" にセットする．
(3)．ＰＬＵを利用したプログラム作成は，ＰＬＵの動作を十分理解していなければ，ＰＬＵの特徴を引き出すことができない．そこで，高速フーリエ変換（ＦＦＴ）などのようによく利用されるプログラムについては，プログラムパッケージとして提供されることが望ましい．

以上，ＰＬＵの提案動機を示し，ＰＬＵシミュレータにおけるプログラム例を示してＰＬＵの処理概念を述べてきた．

３．　ニューラルネット向きＰＬＵ

本章では，２章で述べたＰＬＵの提案動機にしたがったニューラルネット向きのＰＬＵを提案する．

３．１　ニューラルネット向きＰＬＵの構成

電気神経回路（Electronic Neuron Circuit ）は，図６に示すようになる．すなわち，他のニューロンとの接続の大きさを決める抵抗部（シナプシス），入力信号をある関数に従って処理するオペアンプ部（神経セル），出力信号を他のニューロンに送る抵抗部（軸索）からなる．この例としてはHopfield[13] のニューロンモデルがあり，これはアナログ的に動作する神経回路モデルである．そして，これが網状に接続されたものがニューラルネットである．この回路網を参考に，ＰＬＵのように構成すると図７のようになる．この図はニューラルネットの１層分が示されており，結合部（Connecting Point）には，図８に示すように，他の神経回路との結合の強さ $s_{ik}^{(m)}$ および $w_{ij}^{(m)}$ を格納するレジスタ（ＣＲ：Connection Register)がある．このレジスタは，図２における命令レジスタに対応するものであり，メイン計算機のデータバスに接続されてメモリとしてアクセスされる．また，入力部（ＩＤＲ：Input Data Registers）と出力部（ＯＤＲ：Output Data Registers ）もメイン計算機のデータバスに接続されＩＲと同様メモリとしてアクセスされる．なお，図７ではアナログ神経回路を取り入れているが，オペアンプ部にプロセッサを配置してディジタル処理を行うことも考えられる．

このようなニューラルネット向きＰＬＵは，次の２つの機能を持っている．すなわち，結合の強さ $w_{ij}^{(m)}$ のすべてがゼロである場合，多層のパーセプトロンとなる．また，ｉ＝ｋを除く結合の強さ $s_{ik}^{(m)}$ のすべてがゼロである場合，多層のアソシアトロンとなる．従って，このＰＬＵは，多層のパーセプトロンとアソシアトロンが混在したニューラルネットであるといえる．

このように提案当時のＰＬＵの演算器の機能の推移を考えると，図９に示すように推移している．すなわち，２章で示した演算用ＰＬＵはもっとも演算機能が高く，ニューラルネット向きＰＬＵはもっとも演算機能が低いといえる．

３．２　学習の方法

図７に示すｍ層の神経回路の一つをもう少し具体的にその構成を示すと図１０のようになる．この神経回路における関係式は，次式で与えられる．

$$\frac{d}{dt} U_i^{(m)} \cdot C = \sum_j \{ w_{ij}^{(m)} V_j^{(m)} - |w_{ij}^{(m)}| U_i^{(m)} \} R + \sum_k \{ s_{ik}^{(m)} V_k^{(m-1)} - |s_{ik}^{(m)}| U_i^{(m)} \} R$$

ここで，$U_i^{(m)}$ および $V_i^{(m)}$ は，ｍ層神経回路のオペアンプ入力電圧および出力電圧である．また，$w_{ij}^{(m)}$ $(-1 \leqq w_{ij}^{(m)} \leqq 1)$ は同じ層の神経回路ｊの出力との結合の強さであり，$s_{ik}^{(m)}$ $(-1 \leqq s_{ik}^{(m)} \leqq 1)$ は前層ｍ－１の神経回路ｋの出力との結合の強さである．さらに，オペアンプの伝達関数を $A(U_i^{(m)})$ とすれば，$V_i = A(U_i^{(m)})$ となる．従って，この回路の安定状態 $(dU_i^{(m)}/dt = 0)$ におけるｎ個の関係式から，この層の神経回路の入力信号 $V_k^{(m-1)}$ （$1 \leqq k \leqq n$）に対する出力信号 $V_i^{(m)}$ を求めることができる．

この関係式を各層に適用すれば，Ｎ層からなるＰＬＵ

におけるデータレジスタ部からの出力信号$V_k^{(0)}$（$1 \leqq k \leqq n$）に対するN層の神経回路からの出力信号$V_i^{(N)}$が求められる．従って，標準データ｛$V_1^{(0)}$，$V_2^{(0)}$，－－－，$V_n^{(0)}$｝に対する望ましい出力データを｛O_1，O_2，－－－，O_n｝とおけば，エラー

$$E_p = \frac{1}{2} \Sigma_i (V_i^{(N)} - O_i)^2$$

が最小になるように，神経回路間の結合の強さ$s_{ij}^{(m)}$およびw$_{ij}^{(m)}$の値を変化させる．これによって，標準データの学習が可能となる．しかしながら，実際には，非常に多くのパラメータを変化させなければならないので，あるアルゴリズムに従った学習方法を提案しなければならないと思われる．このアルゴリズムについては，今後の課題としたい．なお，このような学習の処理は，メイン計算機が行う．

３．３　簡単なＰＬＵシミュレータの構成

２．２に示したＰＬＵシミュレータと同様の方法で，図７に示すＰＬＵのニューラルネット層が１層の場合におけるシミュレータの構想（現在作成中である）について以下に述べる．２．２に述べたＰＬＵシミュレータと同様，以下のようにメイン計算機の番地を割り当てる．すなわち，$8000～$803F を64個の入力データレジスタ部，$8040～$807F を64個の出力データレジスタ部，$9000～$9FFF を64×64個の結合の強さs_{ik}の格納部，$A000～$AFFF を64×64個の結合の強さw_{ij}の格納部とする．

以上，ニューラルネット向きＰＬＵの構成を提案し，現在作成中の簡単なシミュレータの構想を示した．今後はこれをもとにもっとミューラルネット向きＰＬＵの有効性の議論を進める予定である．さらに，もっと高度なＰＬＵシミュレータの作成を行い，ＶＬＳＩによる実現を試みたい．

４．　まとめ

従来から提案されている演算用のプログラム可能論理演算装置（ＰＬＵ）について，その考え方を説明した．そして，そのような考え方に基づいて，ニューラルネット向きのＰＬＵを提案した．このＰＬＵにおける神経回路にはアナログ神経回路を採用した．また，アソシアトロンとパーセプトロンが混在したものであるため，実際の神経（あるいは脳）に近い処理が期待できる．しかしながら，学習の方法が今後の課題である．今後，これらのＰＬＵシミュレータを通して，このようなＰＬＵのＶＬＳＩ化を試みたい．

［参考文献］

［１］．Hopfield,J.J. and Tank,D.W. :"Computing with neural circuits: A model," Science, Vol.233, pp.625-633, Aug. 1986.

［２］．Graf,H.P., Jackel,L.D., and Hubbard,W.E.: "VLSI implementation of a neural network model," Computer, 3, pp.41-49, March 1988.

［３］．Widrow,B. and Winter,R.:"Neural nets for adaptive filtering and adaptive pattern recognition," Computer, 3, pp.25-39, March 1988.

［４］．Hillis,W.D.:"The connection machine," the MIT press, 1985.

［５］．吉岡：〃メモリステップにプログラム可能な演算機能をもつメモリ装置の提案〃，信学論(D), J66-D, 10, pp.1153-1160, (1983-10).

［６］．吉岡：〃プログラム可能論理演算装置を用いた計算機の実現性について〃，信学論(D), J69-D, pp.1246-1255, (1986-09).

［７］．吉岡：〃プロセッシング・イン・メモリに基づくプログラム可能論理演算装置について〃，データフローワークショップ，pp.175-182, (1987-10).

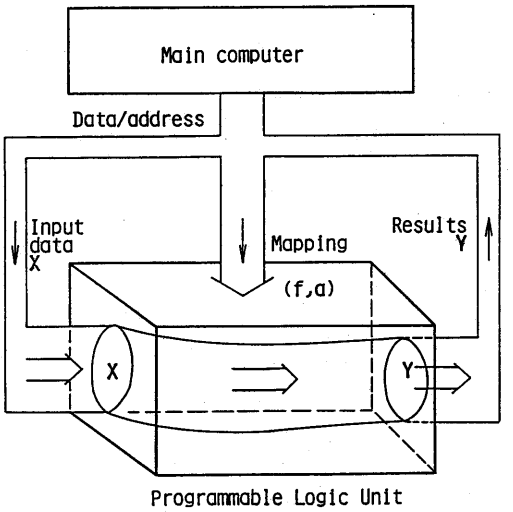

図1　プログラム可能論理演算装置（ＰＬＵ）の概念

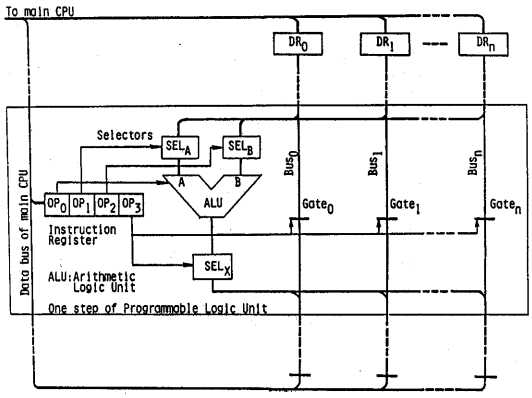

図2　演算処理に適するＰＬＵの構成例

```
stno addr main ins addr plu ins   souce program
0001                              ' Test program 1
0002                              dimension a(100)
0003                              dimension b(100)
0004                              dimension u(100)
0005                              dimension v(100)
0006                              dimension x(100)
0007                              dimension y(100)
0008 6000 F0200006                read n
0009                              '------
0010 6004 81070008                for i=1,n
     6008 92006010
     600C 82080708
     6010 90080506
     6014 92006038
0011 6018 88000809                t1=a(i)
0012 601C 8801080A                t2=b(i)
0013              8400 03090A0B   t3=t1+t2
0014              8404 04090A0C   t4=t1-t2
0015              8408 05090A0D   t5=t1*t2
0016              840C 06090A0E   t6=t1/t2
0017 6020 E0000000                wait
0018 6024 890B0208                u(i)=t3
0019 6028 890C0308                v(i)=t4
0020 602C 890D0408                x(i)=t5
0021 6030 890E0508                y(i)=t6
0022 6034 9200600C                endfor
0023                              '------
0024                              end
```

図3　配列を用いたプログラム例１

```
stno addr main ins addr plu ins   source program
0001                              ' Test program 2
0002 6000 F14DA000                write "a = "
0003 6004 F0200000                read a
0004 6008 F14DA00A                write "b = "
0005 600C F0200001                read b
0006 6010 F14DA014                write "c = "
0007 6014 F0200002                read c
0008                              '
0009              8400 05010103   a1=b*b
0010              8404 05000204   a2=a*c
0011              8408 05050406   a3=4*a2
0012              840C 04030607   a4=a1-a3
0013              8410 08070809   a5=a4//2
0014              8414 040A010B   a6=-b
0015              8418 030B090C   a7=a6+a5
0016              841C 040B090D   a8=a6-a5
0017              8420 0508000E   a9=2*a
0018              8424 060C0E0F   x1=a7/a9
0019              8428 060D0E10   x2=a8/a9
0020              842C 040A0711   aa=-a4
0021              8430 08110812   ab=aa/2
0022              8434 060B0E13   y1=a6/a9
0023              8438 06120E14   y2=ab/a9
0024 6018 E0000000                wait
0025                              '
0026 601C 9007040A                if a4>=0
0027 6020 92006038                then
0028 6024 F14DA01A                write "¥nx1 ="
0029 6028 F120000F                write x1
0030 602C F14DA024                write "¥nx1 ="
0031 6030 F1200010                write x2
0032 6034 92006048                else
0033 6038 F14DA02A                write "¥ny1 ="
0034 603C F1200013                write y1
0035 6040 F14DA030                write "¥ny2 ="
0036 6044 F1200014                write y2
0037                              endif
0038 6048 F14DA036                write "¥n"
0039 604C FFFFFFFF                end
```

図4　２次方程式の根の計算プログラム例２

表1　PLUの命令の一部

命令コード	意味
01xx00zz	X→Z
0200yyzz	Y→Z
03xxyyzz	X＋Y→Z
04xxyyzz	X－Y→Z
05xxyyzz	X＊Y→Z
06xxyyzz	X／Y→Z
07xxyyzz	X^{y} →Z
30xx00zz	sin(X) →Z
31xx00zz	cos(X) →Z
32xx00zz	tan(X) →Z

尚、xx,yy,zzはそれぞれ変数X，Y，Zに対応するレジスタ番号である。

表2　メイン計算機の命令の一部

命令コード	意味
81xx00yy	X→Y
82xxyyzz	X＋Y→Z
83xxyyzz	X－Y→Z
88xxyyzz	X(Y) →Z
89xxyyzz	X→Y(Z)
90xxccyy	比較とスキップ
9200addr	goto addr
9300addr	call addr
94000000	return
E000time	wait
F02000aa	read A
F12000aa	write A
F14Daddr	write "message"
FFFFFFFF	program end

尚、xx,yy,zzはそれぞれ変数X，Y，Zに対応するデータレジスタ番号である。また、ccは条件コードであり、00（＝），01（≠），02（＞），03（＜），04（≧），05（≦）である。さらに、"wait"命令のオペランド"time"は、PLUでの演算を待つためのメイン計算機の待ち時間である。

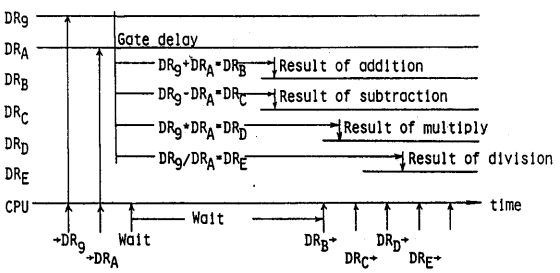

図5　PLUでの演算におけるタイムチャート

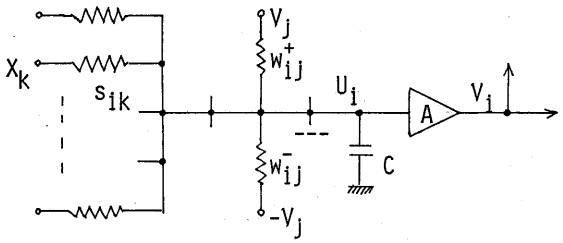

図6　電気神経回路

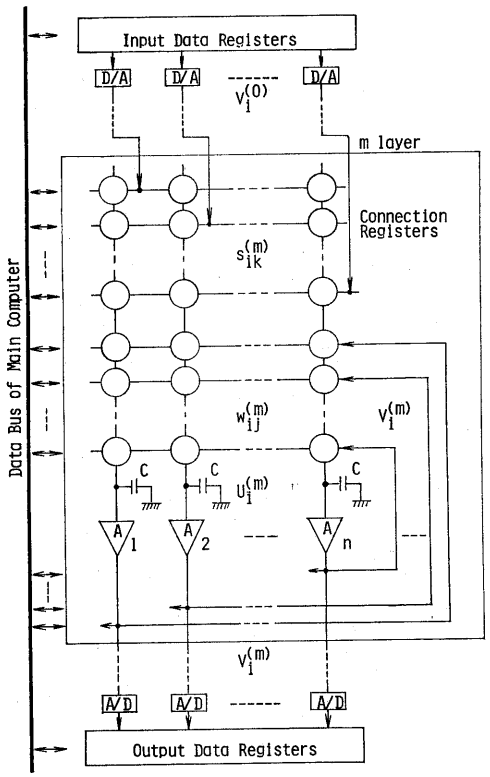

図7　ニューラルネット向きPLU

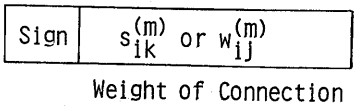

図８　結合の強さ$s_{ik}^{(m)}$および$w_{ij}^{(m)}$を格納するレジスタの構成

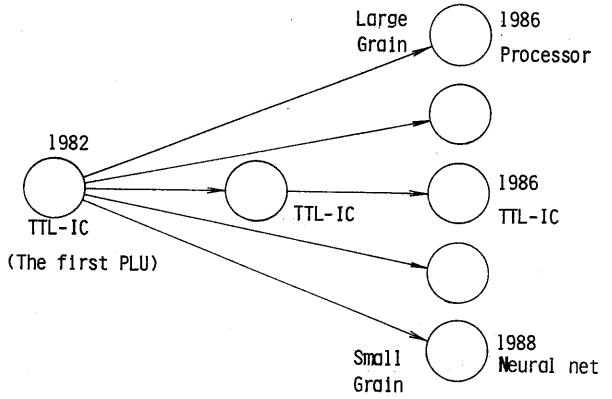

図９　ＰＬＵにおける演算機能の推移

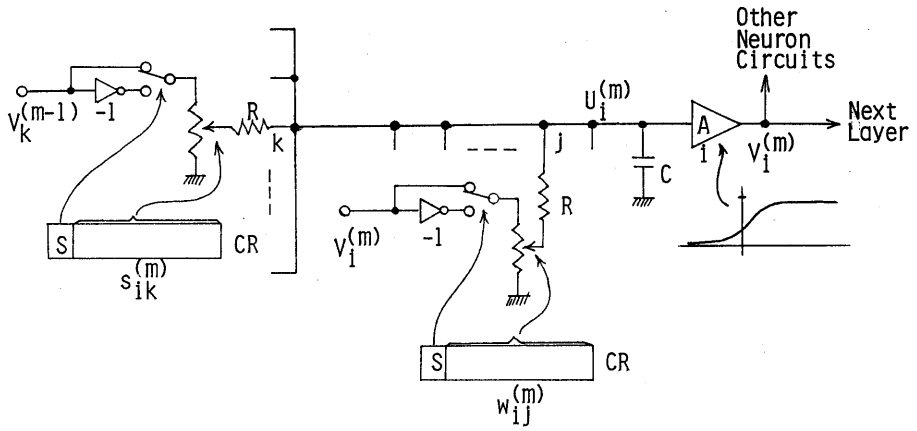

図１０　図７における神経回路ｉの概念図