# エッジ環境における知的情報処理に向けた低消費電力プロセッサアーキテクチャに関する研究

2019年2月

博士（情報科学）

肥田 格

北海道大学

# エッジ環境における知的情報処理に向けた低消費電力プロセッサアーキテクチャに関する研究

## 論文要旨

本研究は、エッジ環境における低電力な人工知能の実現を促進する、知的な情報処理を取り入れたプロセッサのアーキテクチャに関するものである。

今日の人工知能を活用した情報処理は、エッジデバイスで収集されたデータをネットワークで集積し、 データセンタの高性能なコンピュータが集中的に処理する仕組みを取っている。しかし、集積されるデータ量の爆発的な増加や、個人情報などのデータに対するセキュリティ意識の高まりから、エッジ自体にオフライン下でも機能する人工知能を搭載する必要性が唱えられている。エッジデバイスの多くは電池容量や計算性能が限られており、これらに計算負荷の高い人工知能を埋め込むためには、専用処理回路を用いた電力効率の向上が欠かせない。そこで本研究では、エッジに組み込まれるプロセッサへの応用を見込み、(1)CPU 自体の電力効率を向上させる汎用アクセラレータと学習型分岐予測器の開発、(2) 不揮発性メモリを深層学習回路として利用するための学習則の開発に取り組んだ。

CPU はあらゆる演算を実行できるが、その消費電力のうち演算由来の電力が占める割合は 10%ほどでしかないことが知られている。その他のほとんどの電力は、メモリアクセスや CPU 自体の制御で生じている。動的再構成アクセラレータは、大量の演算器を用いてプログラムの一部を CPU の代わりに並列処理することで、これらの非演算依存電力を縮小し、さらに実行速度も加速させる。また、プログラム実行中に演算器間の接続情報が動的に切り替えられるため、多様なプログラムに対応できる。しかし、この再構成時に生じる電力が却って全体の消費電力を引き上げてしまいかねない欠点を併せ持つ。本研究では、プログラムのコントロール・データフローに着目して、演算器間配線のうち、データの依存性が変化し得るデータパスにのみ動的な再構成を許すことで、柔軟性と電力効率を両立するアクセラレータを開発した。

演算以外の電力を削減する手法として、分岐予測器の高精度化も有効である。今日の RISC アーキテクチャに基づく CPU は、およそ 10 段以上の深い命令パイプラインを有している。分岐予測に失敗すると、パイプラインを一時停止して処理途中の命令をクリアし、さらにメモリから適切な命令やデータを読み出し直さなければならない。したがって、パイプラインが深いプロセッサほど分岐予測精度による消費電力増減の影響を受けやすい。本研究では、ベイズの定理に基づく統計的機械学習の手法を取り

入れた分岐予測器を実際のソフトコアCPUへレジスタ転送レベルで組み込み、予測精度と消費電力をシミュレーションした。その結果、静的な分岐予測器と比べて回路規模が巨大化するものの、精度向上に伴い消費電力が削減され得ることを示した。

上述したようなCPUの高電力効率化をもってしても、低電力な人工知能プロセッサの実現には不十分である。現在の人工知能技術はニューラルネットワークが基本形であるが、ニューラルネットワークによる情報処理のほとんどは信号とシナプス結合重みの積和演算の繰り返しである。したがって、命令およびデータを逐次読み出すノイマン型のアーキテクチャとは、相性が良くないといえる。そこで次に、低電力な不揮発メモリとして期待されているメモリスタを深層学習における積和演算回路として利用するための、学習手法の開拓に取り組んだ。

メモリスタは電流や磁場などの外的要因により抵抗が変化する素子である。本来はその抵抗値を利用して情報を記憶するメモリとして用いられるが、並列に接続されたメモリスタに成り立つ電流および電圧則を利用すると、ニューラルネットワークの積和演算をアナログ回路的に処理することができる。また、メモリスタ自体も低電力動作するメモリであり、低電力な深層学習回路としての応用が期待される。本研究では、3次元積層されたメモリの空間的特徴を深層畳み込みニューラルネットワークに利用するための学習法を開発した。提案手法は、学習制御を簡素化するためのフィルタ重み非共有化と、メモリの構造に適した層毎教師なし事前学習の2点で特徴付けられる。得られる局所結合畳み込みDeep Belief Networkは画像の特徴抽出器として機能し、さらに未知のデータセットにたいする転移学習機能も有することが示された。

また、2次元メモリを用いた全結合ニューラルネットワークのための、実装素子数を半減する学習法も開発した。メモリスタによるニューラルネットワークは、人工ニューロンのシナプス結合における重みの正負符号を表すために、1個のシナプスを2個の素子で表現する必要があった。1個のシナプスを1個の素子で置き換えるためには、あらかじめ重みの符号を知る必要がある。そこで、重みの符号を初期化時の状態で固定したまま学習させる方法を考案し、符号非制約下と同等の能力を学習し得ることを示した。

以上、本論文に記した研究は、エッジデバイスに組み込まれるプロセッサにおいて、CPUとメモリ内情報処理の両観点から低消費電力化に寄与するものである。コンピュータの歴史と同様に、人工知能技術もまた、身近なデバイスに広く普及することで更なる発展を遂げるであろう。これらの研究が、小さな電力のエッジ深層学習の実現と、人工知能技術の更なる拡大に貢献することを期待する。

# 目 次

# 第1章　序論

## 1.1　研究の主旨

本研究は、エッジ環境における低電力人工知能の実現を促進する、知的な情報処理を取り入れたプロセッサのアーキテクチャに関するものである。ネットワークに流れるデータ量の爆発的増加やセキュリティ意識の高まりに伴い、人工知能を用いた情報処理システムを、データセンタ集約型からエッジ分散型へ転換する必要にせまられている。現在の人工知能技術の主流である深層学習は計算負荷が高く、エッジデバイス上でリアルタイムかつ小さな電力で演算するためには、ハードウェア・アーキテクチャの効率化は避けて通れない。そこで、メモリスタを用いた深層学習用低電力積和演算器のための学習方法の開拓、および深層学習のための広範な周辺処理を担うCPUを高電力効率化するアーキテクチャの開発に取り組んだ。

CPUの演算性能を低下させずに消費電力を低減させるためには、メモリアクセスやCPU自体の制御で生じる電力を縮小させればよい。動的再構成アクセラレータは、大量の演算器による命令の並列処理と、プログラム実行中の動的なデータパス再構成によって、様々なプログラムを高速かつ低電力実行する。本研究では、再構成されるデータパスを、分岐命令によりオペランドの依存性が変化し得る箇所のみに制限することで、動的再構成のための電力オーバーヘッドを抑制し、柔軟かつ高電力効率な限定的動的再構成アクセラレータを開発した。また、深い命令パイプラインを有する昨今のCPUにおいては、分岐予測の失敗は電力効率を著しく悪化させる。そこで、ベイズの定理に基づく統計的機械学習手法を応用した単純ベイズ分岐予測器をソフトコアCPUにレジスタ転送レベルで組み込み、予測の高精度化による消費電力削減効果を検証した。

さらに、エッジにおける深層学習エンジンとして、メモリスタを用いた非ノイマン型深層学習回路のための学習則の開拓を目指した。メモリスタは電気抵抗が動的に変化する記憶素子であり、その動作原理およびメモリの回路構造を利用すると、深層学習に要する積和演算をアナログ回路的に処理することができる。本研究では、3次元積層されたメモリを、画像の特徴を抽出する深層畳み込みネットワークとして利用する

ために、 局所結合畳み込み Deep Belief Network を開発した。また、2 次元メモリのクロスバー構造を利用した全結合型ニューラルネットワーク実装において、シナプスの化学的特性に発想を得て、必要素子数を半減させる重み符号固定学習法を開発した。

エッジデバイスに搭載されるプロセッサは、CPU や周辺回路を単一チップ上に集積された System-on-a-chip(SoC) の形をとることが多い。ReRAM や MRAM などのメモリスタ素子は、プロセッサのプログラム格納用メモリとしてフラッシュメモリを置換すると考えられており、いずれは SoC に一般的に埋め込まれるであろうことが予測される。したがって、CPU アーキテクチャおよびインメモリ情報処理の両観点から電力効率を向上させるこれらの研究は、あらゆるエッジデバイスへ人工知能が埋め込まれる未来の実現への寄与が期待される。

## 1.2 研究の背景

2006 年に深層ニューラルネットワークにおける勾配消失問題を簡潔に解決する手法が提示されて以降 [1–3]、深層学習による人工知能技術は加速度的な成長を続けている [4, 5]。これらは少なからずハードウェアの進化に支えられているといえる。例えば、コンピュータやモバイル機器を含むあらゆるモノがネットワークに接続されるようになり、これまでにないスケールで膨大なデータが収集されるようになった [6, 7]。また、集積回路技術の成熟により高速・大容量なコンピュータがより安価に入手できるようになったことで、深層学習の高負荷な計算が現実的な実行時間で処理可能となった [4, 8]。そのような背景から、現在の人工知能 (AI: Artificial Intelligence) を利用した情報処理システムは、ネットワーク上のエッジデバイスは単なるセンシング端末として扱い、AI の実体はデータセンタの高性能なコンピュータ基盤上に設置することが多い。この方式は、世界中から収集したデータを効率的に AI に学習させられるが、一方でネットワークを介した情報通信がアプリケーションを制約するボトルネックともなっている。今日では、大量の個人情報を企業が保有することに対して、セキュリティや倫理上の懸念が強まっている他、例えば自動運転に代表されるようなリアルタイム性を要する応用では通信のレイテンシが大きな障害となる。

そのような問題を技術的に解決し得る一つの方法が、エッジ自体にオフライン下で機能する AI を埋め込むことである。しかし、エッジデバイスは必ずしもクラウド環境のように高性能かつ大電力なハードウェアを搭載できるとは限らない。むしろ小容量の電池で駆動される機器などでは、計算速度よりも低消費電力性が要求されることも

ある。したがって、エッジ AI を実現するためには、高い電力効率の AI 処理専用 LSI が不可欠である。

現在の AI の主流である深層学習は、巨大なニューラルネットワークで構成される。ニューラルネットワークを用いた情報処理は、そのほとんどが単純な積和演算の繰り返しであるため、CPU とメモリ間の通信が速度および電力上のボトルネックとなる。そこで、ニューラルネットワークの演算における命令とデータの局所性に着目し、メモリ自体に積和演算回路を埋め込み、演算装置による逐次的なデータの読み書きを省略する、インメモリ情報処理アーキテクチャが提案されている。特に、メモリスタのようなそれ自体が低電力な不揮発性メモリの構造的特徴を利用して、メモリ内部の電流と電圧によるアナログ積和演算器が実現できれば、CPU はデータのセンシングや入出力データの加工など、AI 周辺処理に集中することができる。この場合、CPU 自体は従来のノイマン型アーキテクチャに対するアプローチを引き継いだ低消費電力化が適用可能であり、これらの相乗効果により高い電力効率で AI を処理するプロセッサを実現することができる。

## 1.3　研究目的

以上に述べた背景から、本研究では、エッジ環境への AI 搭載に貢献する高電力効率なプロセッサを実現するためのアーキテクチャ開拓を目的とする。先に述べた背景から、本研究の位置づけは図 1.1 のように表される。ムーアの法則 [11] に基づく半導体プロセスの微細化に限界が見え始めてから、プロセッサは演算コア数を増やすことで処理性能を向上してきた。この進化の線上には、クラウド基盤を構築するような高性能なコンピュータが位置する。それに対して、本研究ではエッジデバイスに適したプロセッサを目指すため、計算の高速化よりもむしろ、従来の計算性能を維持しつつ消費電力を低下させ、性能に対する電力効率の向上を図る。

この目的の達成のためには、まずプロセッサの消費電力特性を知らなければならない。図 1.2 は、動画処理で用いられるある一般的なプログラムに対する CPU の消費電力を解析し、回路の機能別に分類されている [10]。総消費電力のうち演算をこなす ALU の割合はせいぜい 10%程度しか占めておらず、その他の大部分はメモリへのアクセスや CPU 自体の制御で生じている。これは、専用回路を用いて深層学習を低電力化できたとしても、その学習や推論に必要なデータの収集や加工のために電力効率が低下しかねないことを意味する。したがって本研究では、CPU そのものの高電力効率化と、

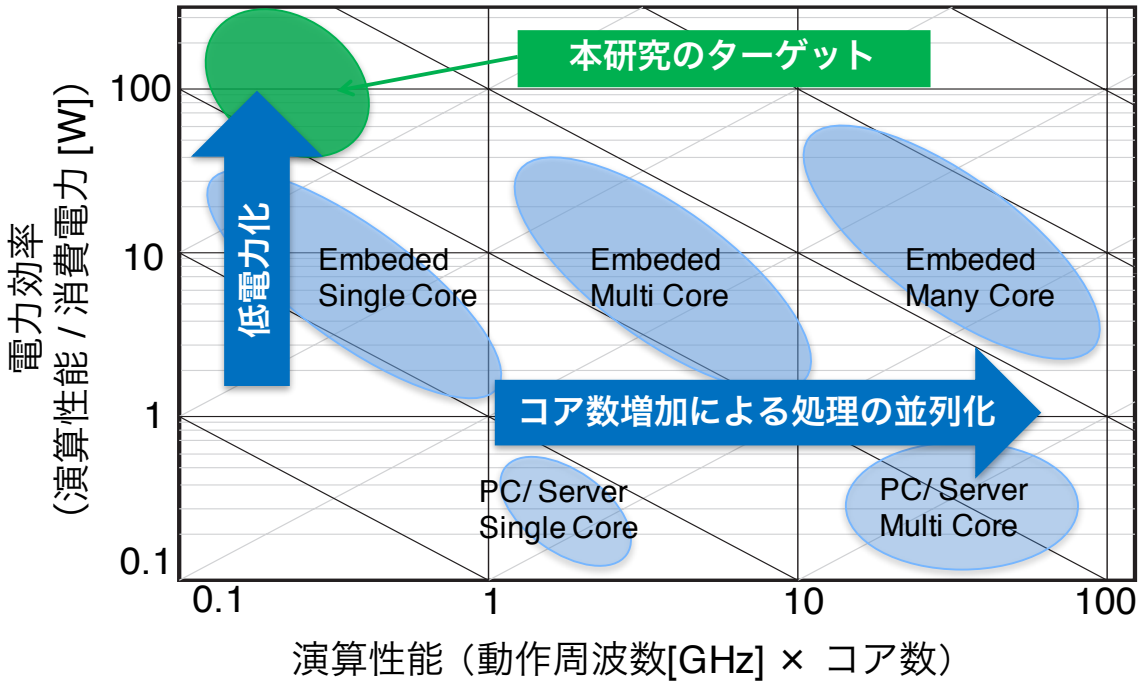



図 1.1 CPU のマルチコア化による処理能力向上と演算性能に対する電力効率の関係 [9]

メモリを用いた深層学習回路の両面から、

- CPU の情報処理を効率化させる動的再構成アクセラレータおよび分岐予測器
- メモリをニューラルネットワークの積和演算回路として活用するための学習手法

の開発に取り組んだ (図 1.3)。

## 1.4 論文構成

### 第 2 章

本章では、柔軟性と電力効率を両立する新たな動的再構成アクセラレータについて述べる。動的再構成アクセラレータは相互接続された数十から数百個のプロセッシング・エレメント (PE) で構成される。各 PE 間はマルチプレクサで接続されており、プログラムを実行している間でも、動的なデータパス再構成が可能である。したがって、様々なプログラムを CPU に代わって並列処理することができる。ところが、従来の動的再構成アクセラレータは、複雑なプログラムに対応するための頻繁な再構成が、却っ

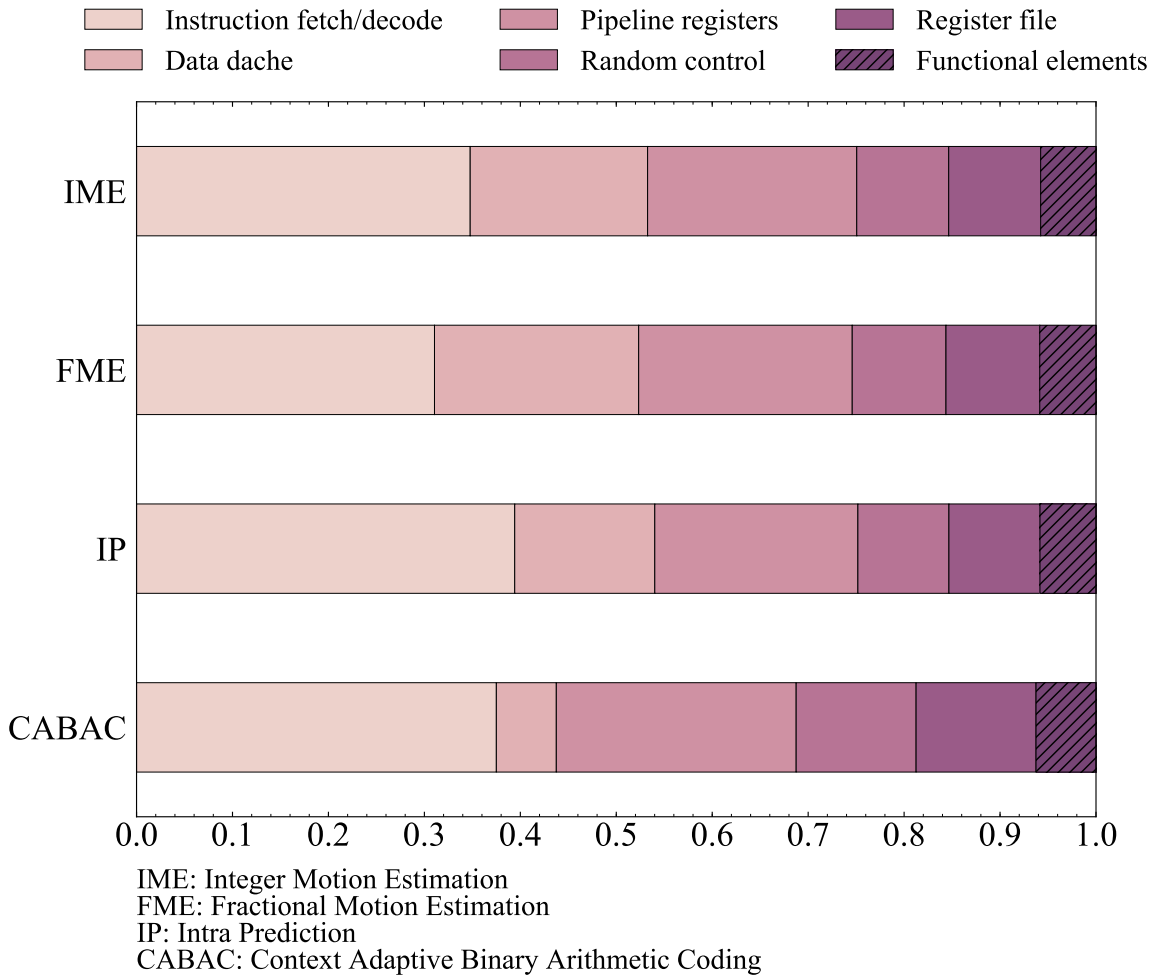



**図 1.2**　RISC プロセッサを用いた H.264 エンコードの各処理におけるデータパス別の消費電力割合 [10]

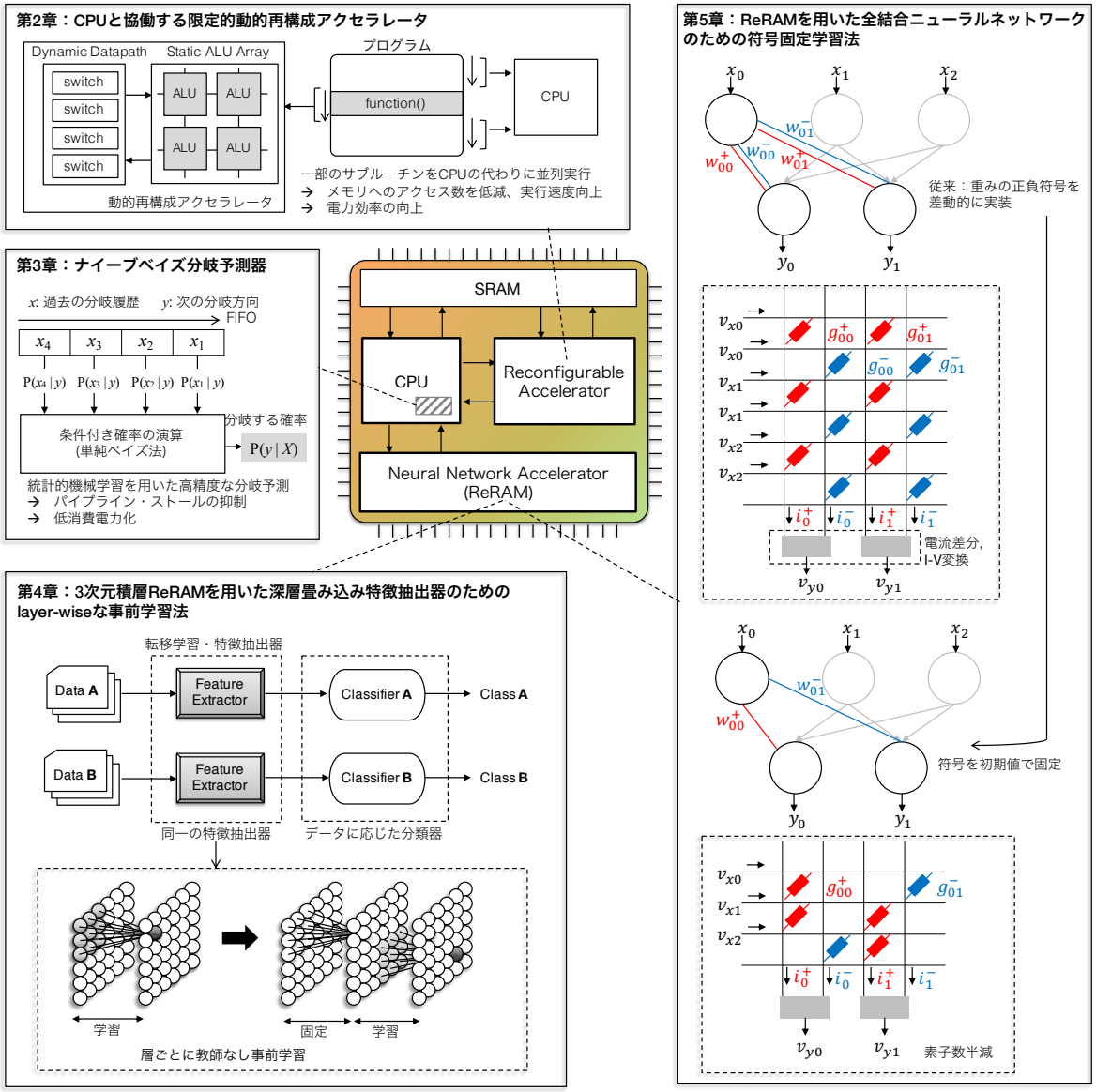



図 1.3 各章で扱う研究内容概観

て消費電力上のオーバーヘッドとなりかねない欠点があった。本研究で開発した手法は、PEアレイの他に演算の途中結果を保存するためのレジスタ・アレイを用意し、PE間の接続はプログラム実行中は不変とし、レジスタ・PE間の接続のみを動的に再構成する。条件分岐命令により命令間で依存性が変化し得るオペランドは、レジスタを介してPEへフォワーディングされる。プログラムのコンテキストが遷移するとレジスタ・PE間のデータパスが再構成され、適切なPEへの経路が生成される。このように、動的再構成されるデータパスを最小限の範囲に収めるアクセラレータを設計し、シミュレーションおよび試作チップによりその電力効率効果を実証する。

## 第3章

本章では、単純ベイズ分類器を応用した学習型分岐予測器による低消費電力化について述べる。今日のCPUはおよそ10段前後にわたる深い命令パイプラインを有している。分岐予測の失敗は処理途中の命令のクリアと、メモリからの正しい命令の再読み出しを招くが、これらによる電力効率のロスはパイプラインが深いほど深刻化するため、分岐予測器の高精度化による低消費電力化は重要な視点であるといえる。高精度な分岐予測器としては、近年では機械学習を用いたアーキテクチャが注目されている。動的な分岐予測器は、一般的に過去の分岐パターンから次の分岐方向を推論するが、この学習および推論器として機械学習のアルゴリズムを取り入れた手法である。しかし、特徴を十分に学習させるためには大容量なレジスタが必要で、これは消費電力を増加させてしまう。そこで、単純ベイズ法を用いた統計的機械学習による分岐予測器をソフトコアCPUにレジスタ転送レベルで組み込み、クロック・サイクル・ベースのシミュレーションによる非学習型の分岐予測器との消費電力比較を通して、その有用性を検討する。

## 第4章

本章では、3次元積層不揮発性メモリをNNの積和演算器として利用するための学習手法について述べる。現在の深層学習技術では、一般的に誤差逆伝播法を用いて全ニューロン層を一度に学習させる。メモリスタを用いた3次元積層メモリの場合、シナプス結合は各層間に配置される記憶素子で実現され、アナログ回路的に積和演算が実行される。しかし、層間を跨ぐ誤差逆伝播法はメモリ外部に複雑な制御回路を要する。したがって、自己符号化器や制限付きボルツマンマシンを用いて各層毎 (layer-wise) に順

番に教師なし事前学習を行う方が、より現実的である。また、本研究は3次元メモリの空間的特徴を最大限に活用できる深層畳み込みネットワークへの応用を目指す。そこで、層間配線を簡単化するために、フィルタ間の重みを共有しない (locally-connected) 畳み込み手法を用いる。以上のようにして構築した局所結合畳み込み Deep Belief Network は画像に対する特徴抽出器として機能し、さらに未学習データセットに対する転移学習能力も有することを示す。

### 第5章

本章では、従来の半分の素子数でメモリスタ・ニューラルネットワークを実装する学習手法について述べる。メモリスタを用いた2次元メモリのクロスバー構造を利用すると、全結合型ニューラルネットワークを自然な形で実装できる。ただし、人工ニューロンにおけるシナプス結合荷重は正の値も負の値も取りうるのに対し、負の抵抗値は今のところ実世界に存在しない。したがって、これまでは1個の重みに対して2個の素子を用い、各素子の電流差分をとることで仮想的に重みの正負符号を表現する必要があった。1個の重みを1個の素子で表現するためには、あらかじめ重みの符号を知らなければならない。一般的に、ニューラルネットワークを学習させる前には、各重みを乱数で初期化する。そこで本研究では、初期化時の状態で重みの符号を固定し、学習中に0を跨ぐ符号反転を禁止する学習法を提案し、その学習可能性を検証する。

### 第6章

以上の研究を総括する。

# 参考文献

[1] G. E. Hinton and R. R. Salakhutdinov, "Reducing the Dimensionality of Data with Neural Networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[2] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A Fast Learning Algorithm for Deep Belief Nets," *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, jul 2006.

[3] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy Layer-wise Training of Deep Networks," in *Proceedings of the 19th International Conference on Neural Information Processing Systems*, ser. NIPS'06. Cambridge, MA, USA: MIT Press, 2006, pp. 153–160.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS'12 Proceedings of the 25th International Conference on Neural Information Processing Systems*, 2012, pp. 1097–1105.

[5] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.

[6] E. S. Berner and J. Moss, "Informatics challenges for the impending patient information explosion," *Journal of the American Medical Informatics Association : JAMIA*, vol. 12, no. 6, pp. 614–617, 2005.

[7] F. Mattern and C. Floerkemeier, "From the Internet of Computers to the Internet of Things BT - From Active Data Management to Event-Based Systems and More: Papers in Honor of Alejandro Buchmann on the Occasion of His 60th Birthday," K. Sachs, I. Petrov, and P. Guerrero, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 242–259.

[8] N. P. Jouppi *et al.*, “In-Datacenter Performance Analysis of a Tensor Processing Unit,” in *Proceedings of the 44th Annual International Symposium on Computer Architecture*, ser. ISCA ’17. New York, NY, USA: ACM, 2017, pp. 1–12.

[9] マイクロプロセッサ専門委員会, Ed., **組込みマルチコアハンドブック技術・応用編**. 電子情報技術産業協会, 2013.

[10] R. Hameed *et al.*, “Understanding sources of inefficiency in general-purpose chips,” *ACM SIGARCH Computer Architecture News*, vol. 38, no. 3, p. 37, jun 2010.

[11] J. K. Steehler, “Understanding Moore’s Law—Four Decades of Innovation (David C. Brock, ed.),” *Journal of Chemical Education*, vol. 84, no. 8, p. 1278, aug 2007.

# 第2章　限定的動的再構成アクセラレータ

## 2.1　緒言

ノイマン型コンピュータにおいて情報処理の中心を担うマイクロプロセッサは、その消費電力のうち演算に起因する電力は約10%しかなく、残りの90%は、(1) メインメモリからの命令/データの読み出しおよびデータの書き出し、(2) レジスタファイルへの演算結果読み出し/書き出し、(3) パイプラインレジスタの駆動、のようなデータの読み書きやプロセッサ自体の制御処理で生じることが知られている (図2.1)。この事実は、演算に直接は関係しない90%の電力を縮小することができれば、プロセッサの演算性能を損なわずに、低消費電力・高電力効率化が実現されることを示唆している。

このような非演算依存の電力の大部分は命令の逐次実行方式に由来するため、演算の並列化による電力の削減が期待される。例えば、プロセッサの算術論理演算回路 (ALU: Arithmetic Logic Unit) を二次元アレイ状に複数個配置し、各 ALU 間をマルチプレクサで相互に接続する。そして、プロセッサが処理するプログラムの中で、その実行時間を占める割合が最も高い関数 (ホットパス) をこの ALU アレイ上に展開する。すなわち、関数に含まれる各命令を、命令間のデータ依存性を保持しつつ各 ALU に割り当てる。すると、メモリやレジスタファイルを介さずに命令を並列処理することができるため、前述の (1)〜(3) に挙げたような余剰電力を大幅に削減することができる。

上述したアーキテクチャを根源として、プログラムの一部をプロセッサの代わりに処理する再構成可能なアクセラレータが数多く研究されてきた [2–6]。その中でも動的再構成プロセッサ [2–4] は、汎用性に優れる利点をもつ。動的再構成プロセッサは、次の手順でアクセラレータを利用する。

1. プログラムのコンパイル時にアクセラレータへのマッピング対象となる関数を抽出する。

2. 抽出した関数のコンテキスト毎に回路の構成情報を生成し、アクセラレータに接

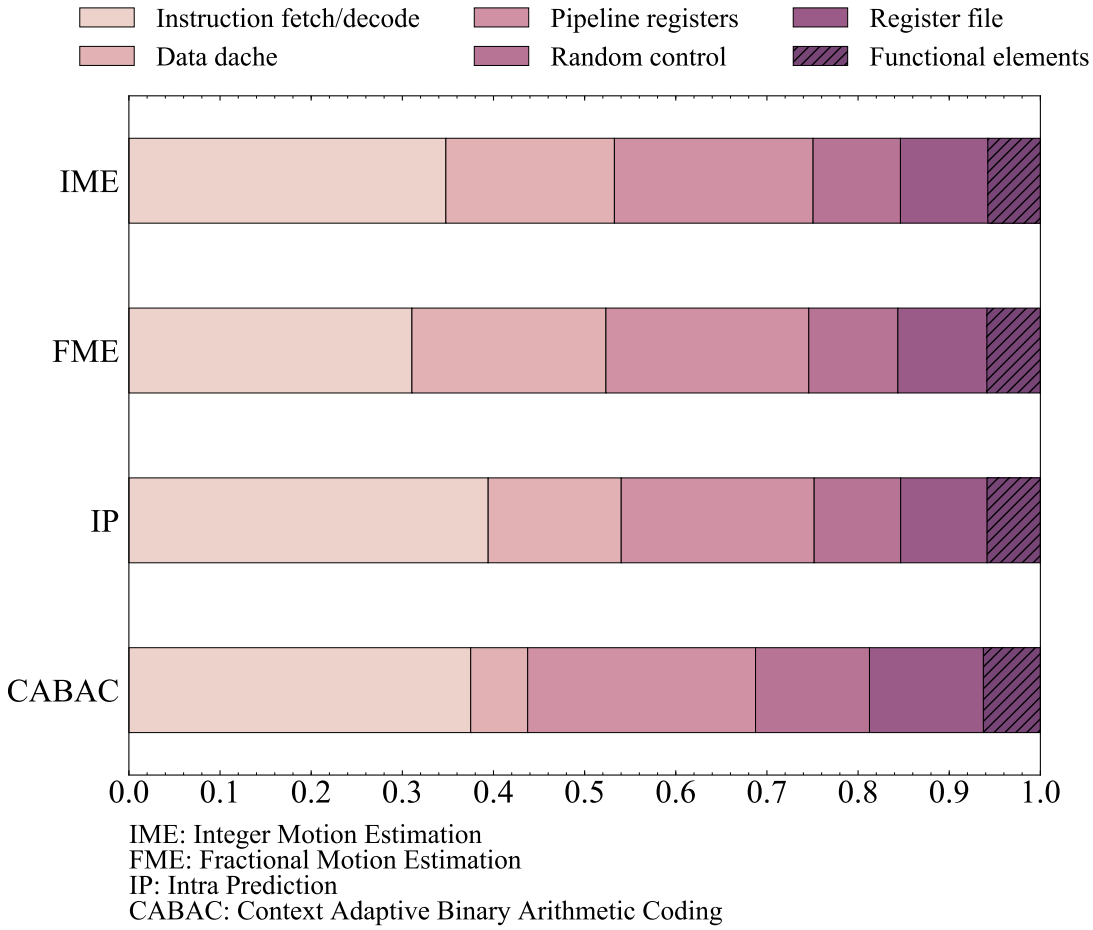



**図 2.1** RISC プロセッサによる H.264 エンコードにおいて、各処理で発生する消費電力のデータパス別割合 [1]

続された構成情報メモリへ書き込む。

3. プログラムが実行され対象関数が呼び出されると、アクセラレータが起動してアクセラレータ内の回路を再構成し (非動的な再構成)、関数を並列的に処理する。

4. もし関数処理中にコンテキストが変化すると、状態遷移制御回路によってコンテキストに対応する構成情報を構成情報メモリから読み出し、アクセラレータの各要素が再構成される (動的な再構成)。

アクセラレータ起動中の動的な再構成により、命令数が多くアクセラレータに展開しきれない関数に対しては、関数を複数のブロックに時分割し、ブロック実行毎に回路を再構成することで処理可能であり、また、関数内部のデータ依存性が関数内部の条

件分岐命令により変化した場合も、マルチプレクサを切り替えるだけで演算を継続することができる。ところが、回路要素の再構成や構成情報読み出し時にも当然電力が生じるため、頻繁に再構成が実行されると却って全体の消費電力が増大しかねないため、応用対象となるアプリケーションや OS の限定 [5] や、アクセラレータにマッピング可能な関数を分岐命令数依存で制限する [6] などの回避策がとられてきた。

そこで本研究では、プログラムのコンテキストにおけるコントロール・データフロー変化の特性に着想を得て、非動的に再構成される ALU アレイと動的に再構成されるデータパスに分割することで、柔軟性と低電力性を両立する限定的動的再構成アクセラレータを開発した。提案アクセラレータを RISC アーキテクチャのソフトコア CPU [7] に統合し、サイクル・ベース・シミュレーションによる性能評価を行った。また、提案アクセラレータのテストチップを試作し、消費電力測定を行った。

本章の構成は、以下の通りである。2.2 節では、一般的なプログラムにおける命令間のデータ依存性について、コントロール・データフローの観点から考察する。2.3 節では、本研究で提案する限定的動的再構成アーキテクチャについて解説する。2.4 節ではシミュレーションおよび測定の結果を示す。

## 2.2　限定的動的再構成アクセラレータ

本章で提案する動的再構成アクセラレータ DYNaSTA は、static ALU array(STA) と dynamic operand forwarding matrix(DYN) の 2 つの回路ブロックで構成される (図 2.2)。STA は命令列を並列実行することで電力効率を向上させ、DYN はデータパスを動的に再構成することで柔軟性を高める。

プログラムからアクセラレータによる加速対象サブルーチンが抽出される (図 2.3(a)) と、サブルーチン内の命令列のデータフロー (図 2.3(b)) をもとに STA の構成情報が生成される。構成情報が構成情報メモリからロードされると、各命令が STA の ALU に割り当てられ、STA のスイッチが適切に設定される。ここで、条件分岐命令によりオペランドの依存性が変化する命令間のデータパスは、DYN に割り当てられる。DYN の構成情報は各コンテキスト毎のデータ依存性情報 (図 2.3(c)) から生成され、アクセラレータ駆動中にサブルーチンのコンテキストが切り替わると、構成情報に基づいて DYN のデータパスが動的に再構成される。

このように、DYNaSTA アクセラレータは動的に再構成される回路を必要最低限の範囲に限定することで、電力効率と柔軟性を両立させる。

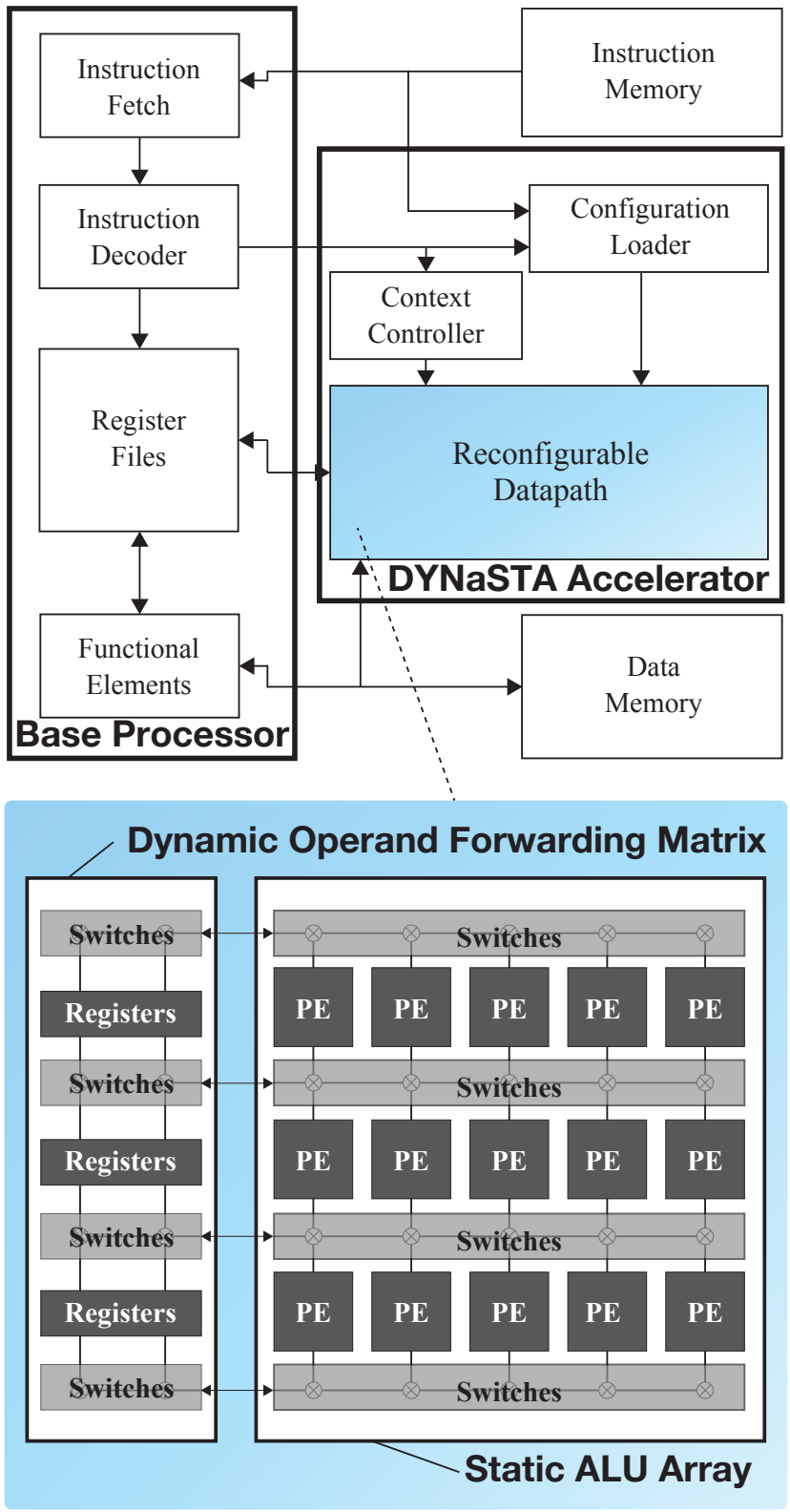



図 2.2 提案する DYNaSTA アクセラレータのアーキテクチャ概観

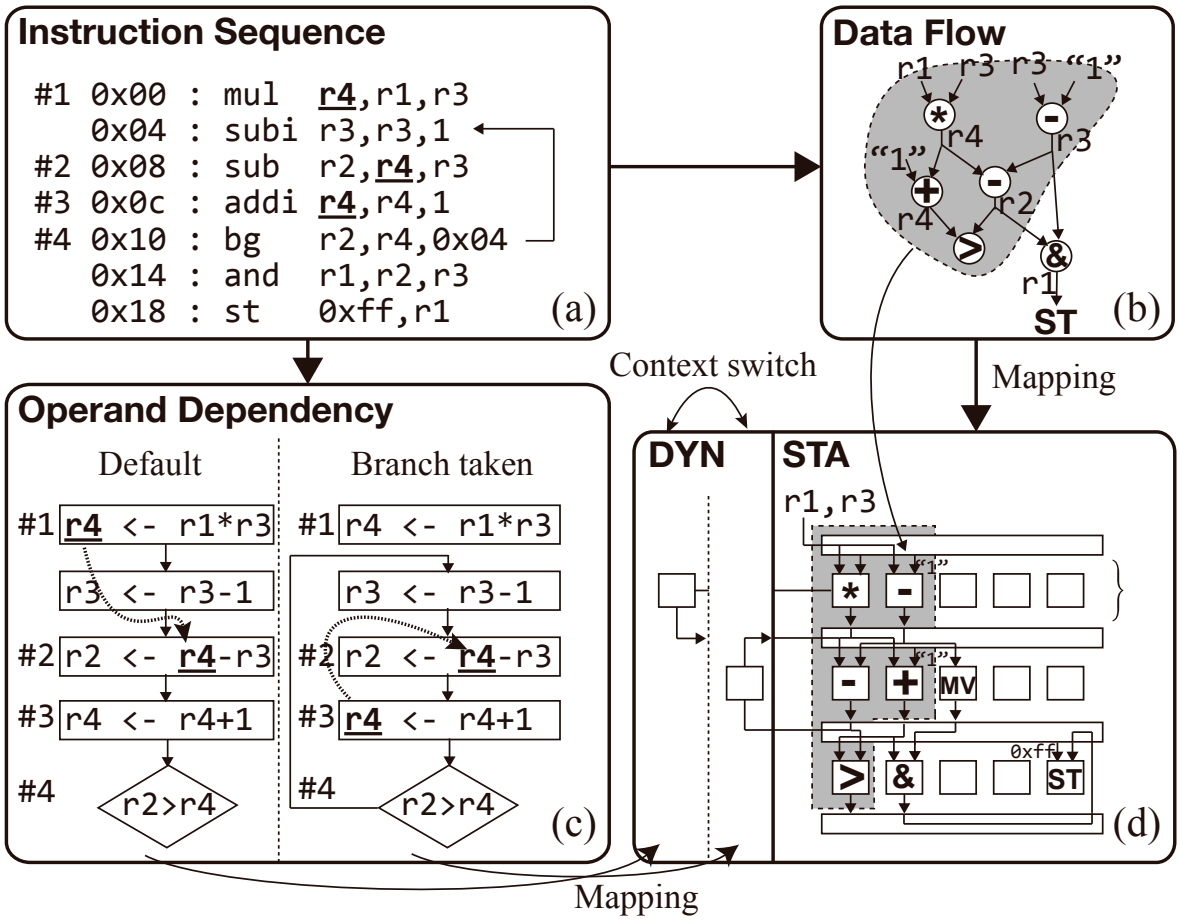



**図 2.3**　DYNaSTA アクセラレータへの命令列のマッピング方式：(a) 命令列の一例、(b) 命令列から抽出されたデータフロー、(c) 命令列から抽出されたオペランド依存情報、(d) DYN および STA へのマッピング

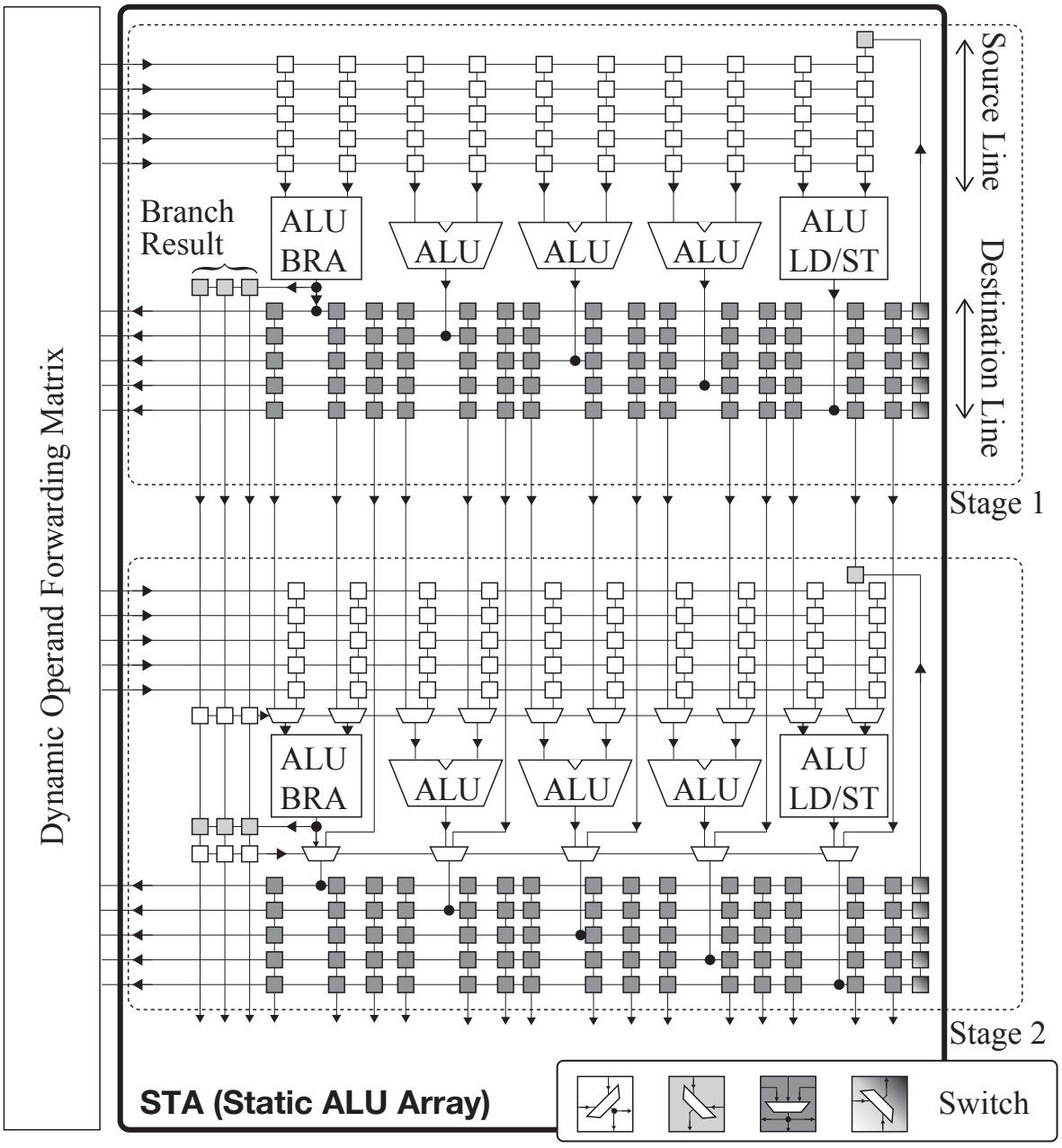



**図 2.4** Static ALU array (STA) のブロック図

### 2.2.1 Static ALU array

STA は ALU 配列とスイッチ配列を複数段積み重ねた回路である (図 2.4)。各 ALU はベースとなるプロセッサの算術論理演算命令をサポートするが、回路面積を縮小するために、分岐命令は各 ALU 配列の左端、ロード・ストア命令は右端の ALU のみがサポートする。スイッチ配列は、ALU 配列への入力配線を制御するソース・ラインと、ALU 配列からの出力配線を制御するデスティネーション・ラインに分割できる。一つの ALU 配列と、その ALU 配列の前後にあるソース・ラインおよびデスティネーション・ラインをまとめてステージとよぶ。

各 ALU および各スイッチの構成は、サブルーチンがアクセラレータ上にマッピングされてからサブルーチン実行終了まで維持される。したがって、オペランドに依存性がない命令同士は同一ステージ内の ALU にマッピングされて並列化されるが、オペランドがいずれかのコンテキストにおいてどちらか一方にでも依存する命令同士は別々のステージにマッピングされなければならない。例えば図 2.3(a) における#2 の sub 命令で用いられる変数 r4 は、#1 の mul 命令の演算結果である r4 に依存するため、#1 と#2 は別々のステージにマッピングされなければならない。

各ステージから出力される演算結果は、他のステージに対して、前段のデスティネーション・ラインから後段のソース・ラインへの直接的な受け渡しと、DYN を介した間接的な受け渡しが可能である。STA のスイッチ構成はサブルーチンがマッピングされた段階で固定されてしまうため、コンテキストの変化によって依存関係が変化した変数は、DYN を介して適切なステージに受け渡される。

### 2.2.2 Dynamic operand forwarding matrix

DYN は、各ステージから出力された演算の途中結果を保存するためのレジスタと、これらレジスタと STA のステージ間の配線情報を制御するスイッチ配列で構成される (図 2.5)。DYN のスイッチ配列は、プログラムのコンテキスト変化に伴うデータフローの変化を解決するために、動的に再構成される。

例えば図 3(c) 中における r4 のような、コンテキストによって依存性が変化する変数は、まず STA から DYN のレジスタへ値を保存するようなデータパスが生成される。そして、アクセラレータ駆動中に分岐命令によってコンテキストが切り替わると、DYN のスイッチが再構成されて適切なステージへフォワーディングするデータパスが生成される (図 2.3(d))。

本研究の消費電力評価で用いたベンチマークのうち、フィボナッチ数を計算するプログラムに対するDYNの動的再構成の様子を、図2.6に示す。図からもわかるように、DYN内の接続情報はプログラム実行中に動的に切り替わるのに対し、STA内の接続情報はマッピング段階の状態を保持する。このように、必要最低限の配線のみを再構成することで、高い電力効率を実現する。

### 2.2.3 Context controller

コンテキスト・コントローラ (図2.7) はコンテキスト遷移を制御する。コンテキストの情報は、構成情報とともにプログラムのコンパイル時に生成される。プロセッサがプログラムの実行を開始すると、コンテキスト情報はコンテキスト・メモリへ、構成情報は構成情報メモリへロードされる。

コンテキスト情報には、各コンテキストにおけるDYNのスイッチの構成情報、コンテキスト実行にかかるクロック・サイクル数および遷移先のコンテキストとサブルーチン実行終了後の戻り値アドレスが含まれる。コンテキスト実行中にはクロック数がカウントされ、クロック・カウンタとコンテキスト情報のクロック・サイクル数が等しくなったとき、条件分岐命令の結果によってコンテキスト遷移がトリガされると、遷移先のコンテキスト情報がロードされてDYN　のスイッチが再構成される。DYNaSTAによるサブルーチン実行が終了すると、プロセッサのプログラム・カウンタに戻り値アドレスが書き込まれる。プロセッサは、サブルーチンコール終了後の命令からプログラムの実行を再開する。

### 2.2.4 DYNaSTA プロセッサ

DYNaSTAアクセラレータは、典型的なRISCプロセッサと図2.8のように統合される。本研究では、ベース・プロセッサとしてLattice Semiconductor社のソフトコアCPUであるLatticeMico32 [7] を採用した。プロセッサとDYNaSTAアクセラレータは、プロセッサのレジスタファイルの一部を共有する。プロセッサとアクセラレータ間で引数や戻り値の受け渡しが必要な場合は、共有レジスタファイルを利用する。

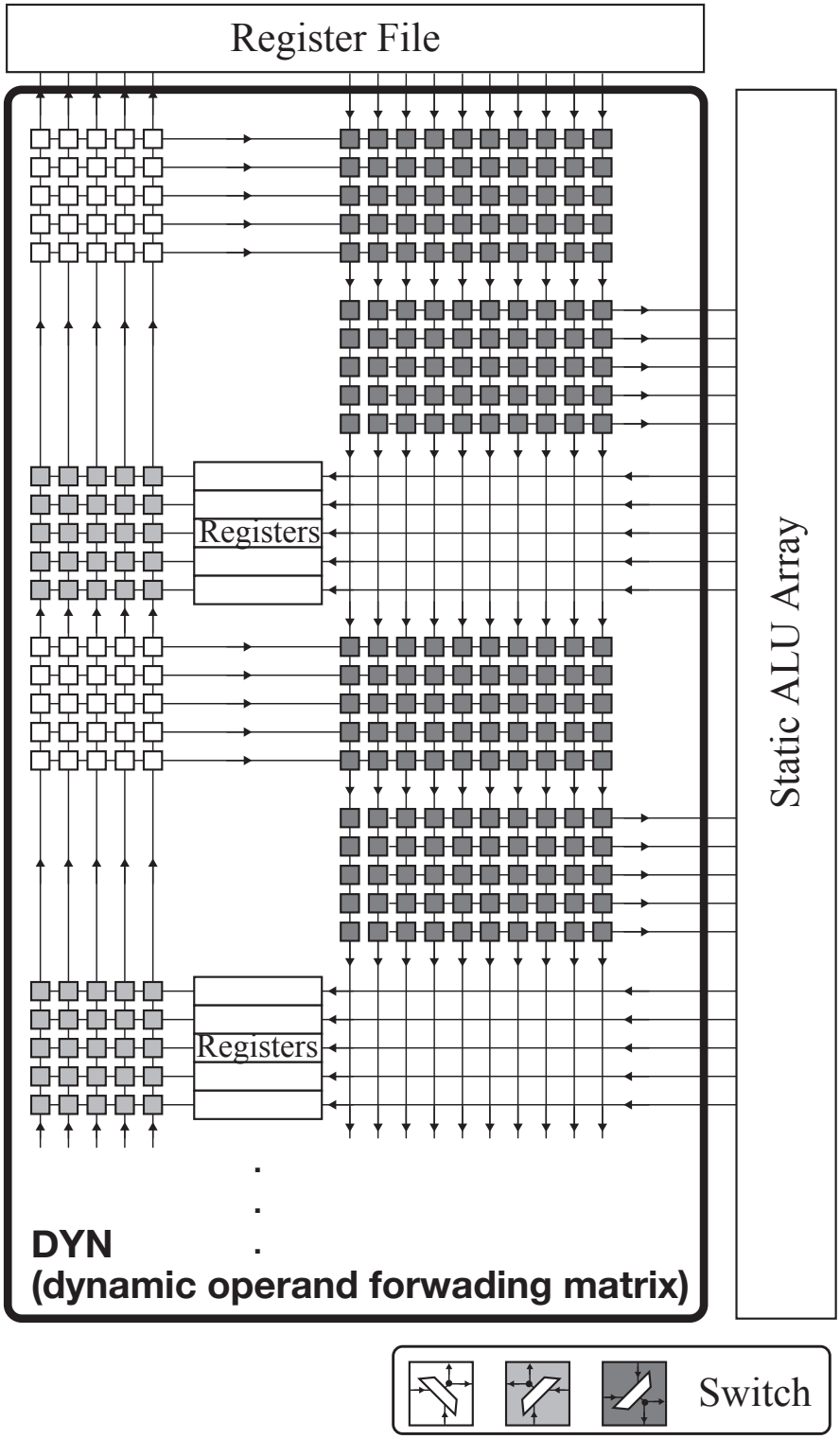



**図 2.5**　Dynamic operand forwarding matrix (DYN) のブロック図

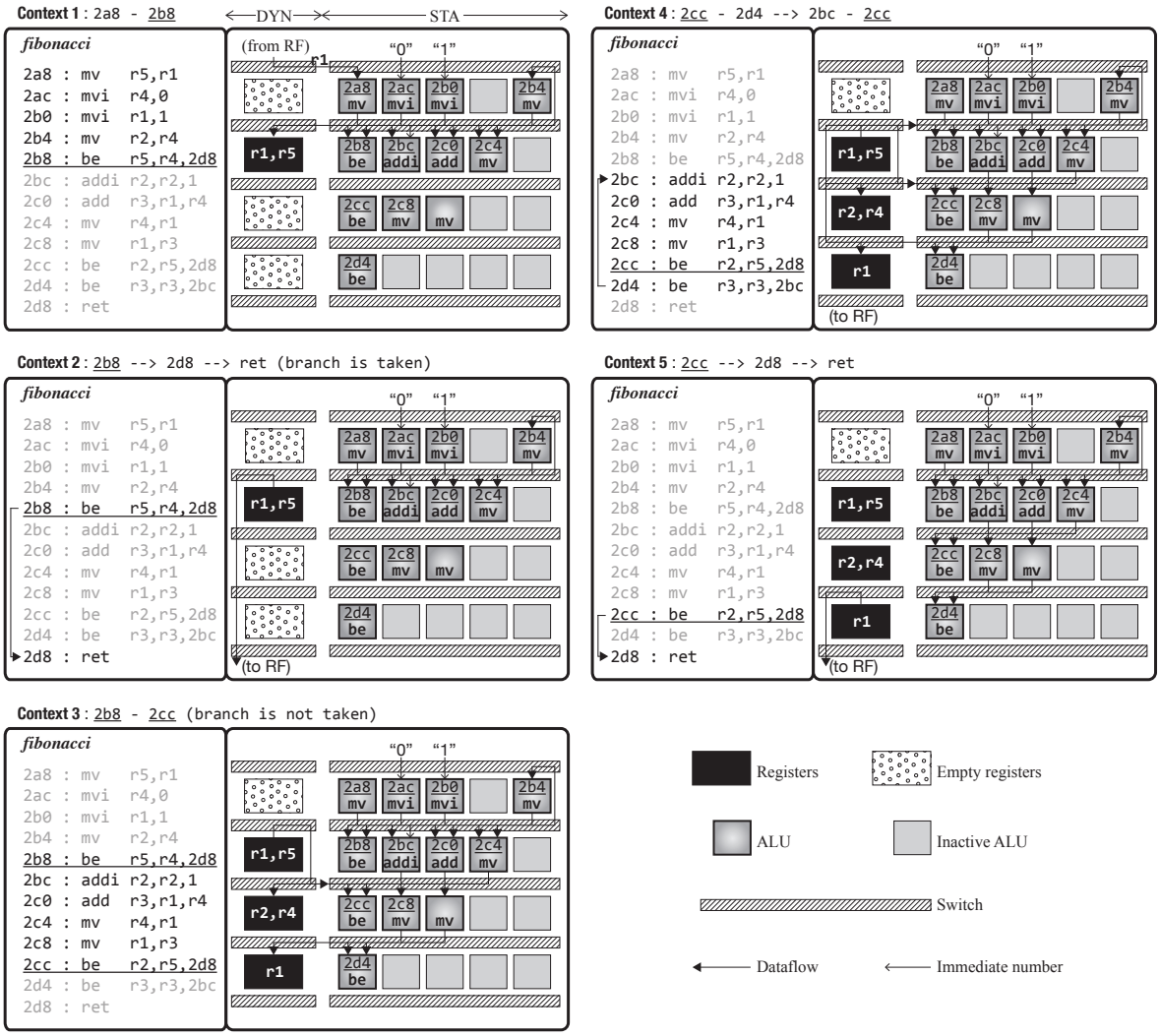



図 **2.6** STA への命令列マッピングおよび DYN の動的再構成の一例

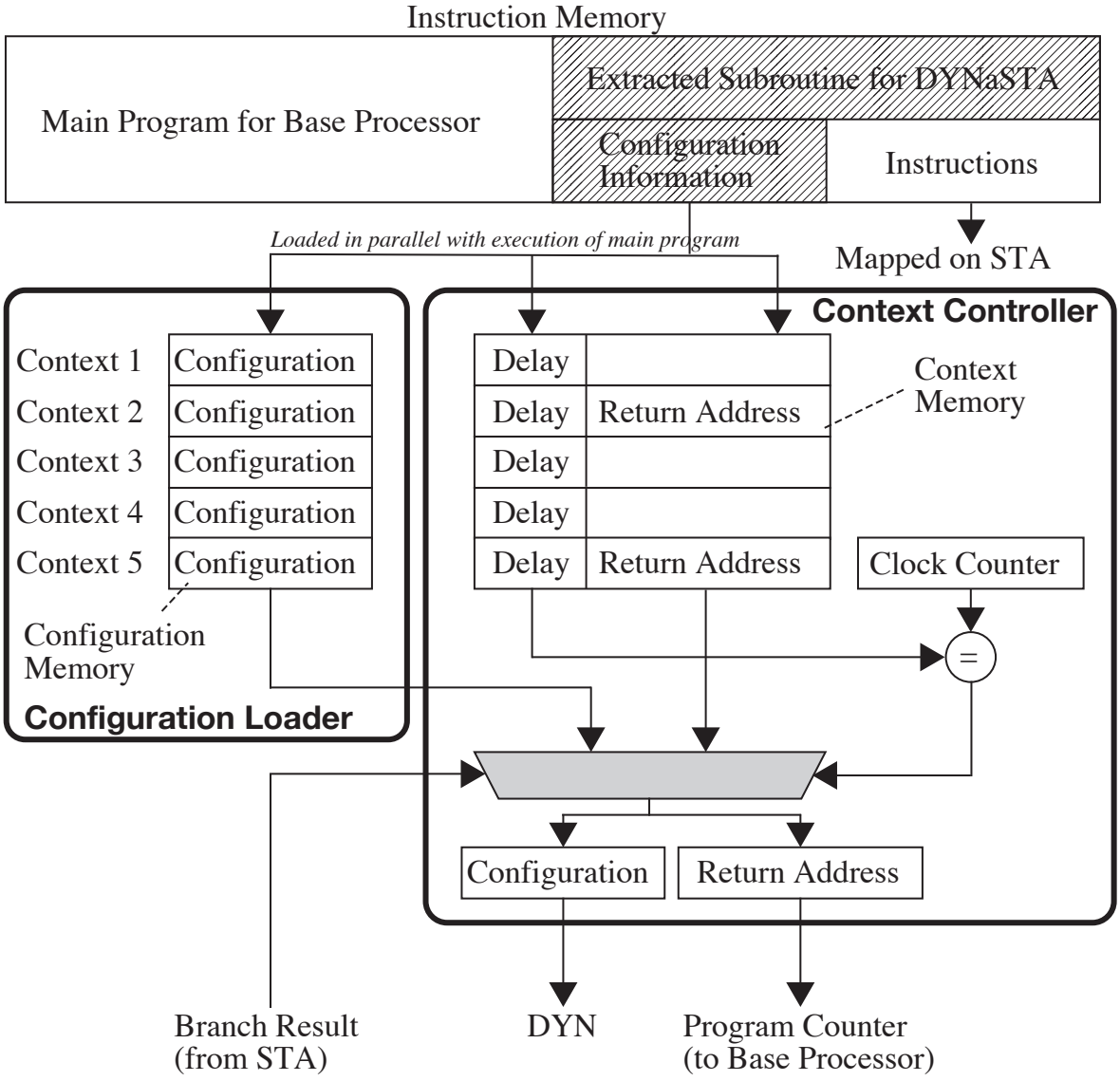



図 **2.7**　動的再構成を司るコンテキスト・コントローラのブロック図

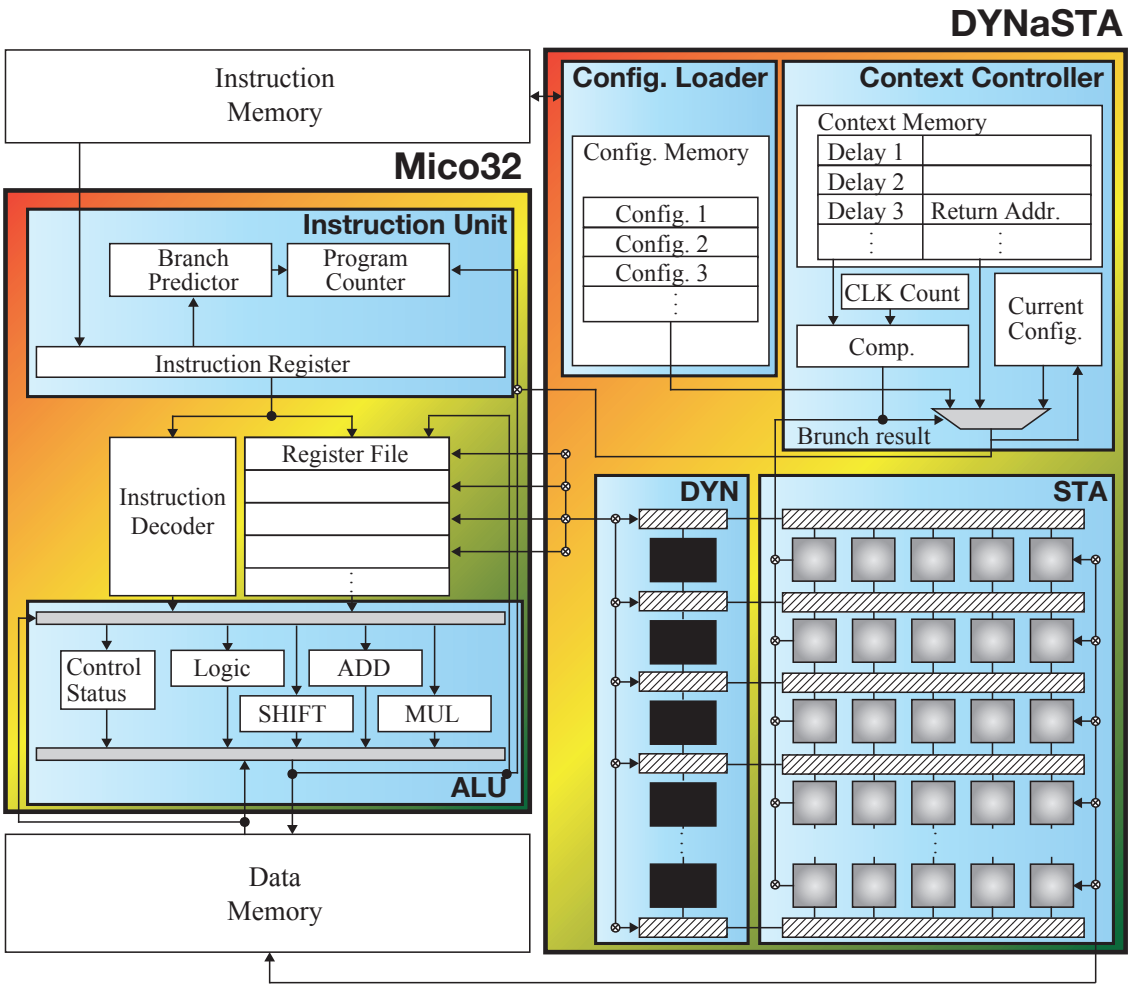



図 **2.8** LatticeMico32 と DYNaSTA アクセラレータの統合アーキテクチャ

## 2.3　評価

### 2.3.1　命令レベルの並列性

STA の各ステージに含まれる ALU の数は、消費電力と命令レベルの並列性から決定される。ALU が多すぎると、命令が割り当てられていない ALU による無駄な静的電力が増え、少なすぎると、本来は並列化可能な命令を同一ステージにマッピングすることができず、演算性能が低下する。そこで、いくつかのベンチマーク・プログラムをアクセラレータにマッピングし、ALU の使用率と命令の並列性を調査し、図 2.9 に示す結果を得た。

図 2.9 において、実線は ALU 使用率を、破線は命令レベル並列性を、x 軸は 1 つのステージに含まれる ALU(PE: Processing element) の個数を表す。ALU の使用率は、ステージあたりの ALU 数が増えるにつれて線形に減少した。命令レベルの並列性は、CRC32 と sbox においては 1 ステージあたりの ALU 数に対して不変であるが、Sepia Filter では ALU 数が 4 のとき低下した。したがって、本研究では 1 ステージあたり 5 個の ALU を実装し、消費電力の評価を行った。

### 2.3.2　消費電力シミュレーション

表 2.1 に示すベンチマーク・プログラム [8] を用いて、クロック・サイクル・ベースで消費電力をシミュレーションした。ベースとなるプロセッサとして LatticeMico32 [7] を用い、DYNaSTA アクセラレータの 1 ステージあたり ALU 数は 5 個、STA のステージ数は 10 段とした。

消費電力は、各プログラムのホットパス実行時の電力のみを見積もった。すなわち、プロセッサの場合はプログラムを開始から終了まで全実行し、そのうちホットパス実

**表 2.1**　シミュレーションで用いたベンチマーク・プログラム [8]

| Application | # of inst. | # of br. | PE utilization [%] | # of contexts |
|---|---|---|---|---|
| **fibonacci** | 12 | 3 | 24 | 5 |
| **sbox** | 25 | 2 | 50 | 5 |
| **crc32** | 18 | 2 | 36 | 5 |
| **sepia filter** | 22 | 1 | 44 | 3 |

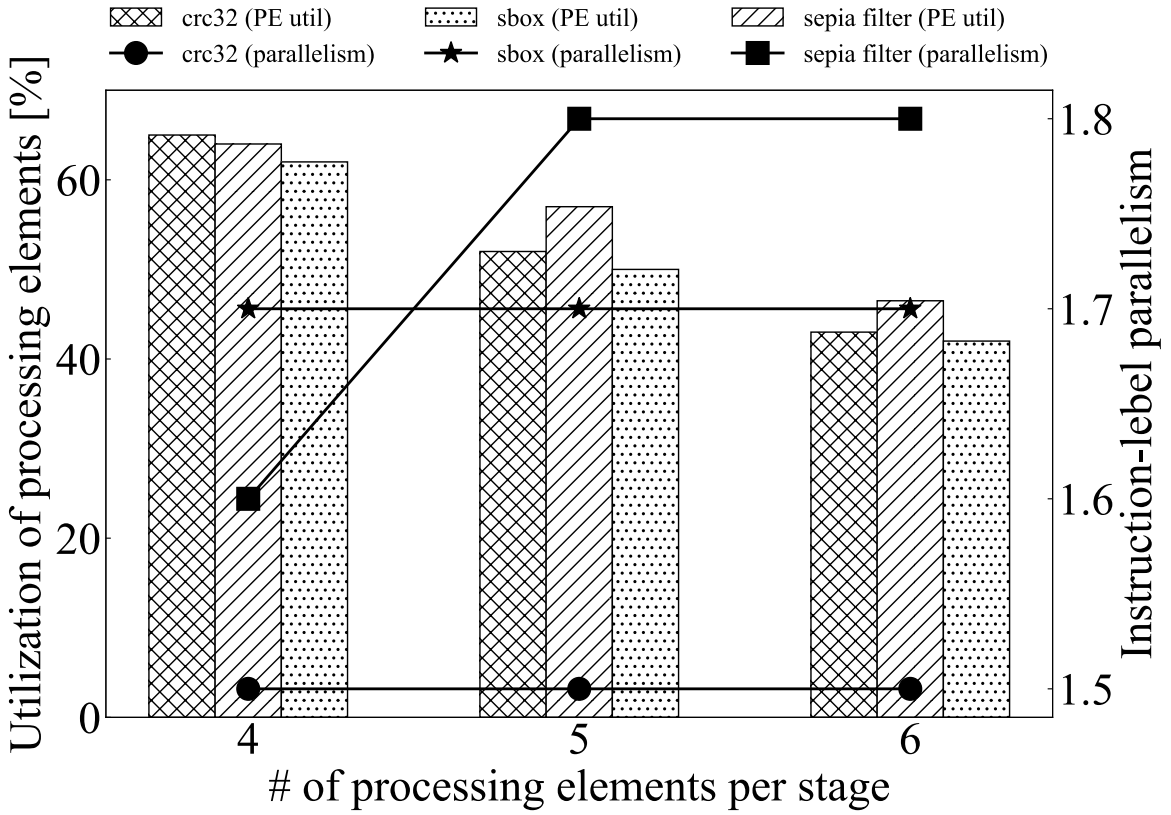



**図 2.9** 1 ステージあたりに含まれる PE 数に対する PE 占有率および命令レベル並列性の変化

:w

行中の消費電力のみをモニタした。一方 DYNaSTA アクセラレータの場合は、まず実行ファイルをディスアセンブルしてホットパスを抽出し、次にホットパスのみを再コンパイルしてアクセラレータの構成情報を生成した。そして、テストベンチ上で構成情報をアクセラレータの構成情報メモリとコンテキストメモリに、ホットパスの命令列を命令メモリ・モジュールにロードし、かつプロセッサのレジスタファイル・モジュールに対して適当な初期値を設定して、ホットパス部分のみの実行電力を解析した。また、命令メモリおよびデータメモリにアクセスする際の消費電力は、シミュレーションから得られた各メモリへのアクセス回数と一般的なオンチップ SRAM へのアクセス時に生じる消費電力 [9] から見積もった。

得られた消費電力の見積もりを図 2.10 に示す。LatticeMico32 は 1 クロックサイクル毎に命令メモリから命令をフェッチしなければならないのに対し、DYNaSTA アクセラレータは構成情報をロードするときにのみ命令メモリへアクセスし、サブルーチン

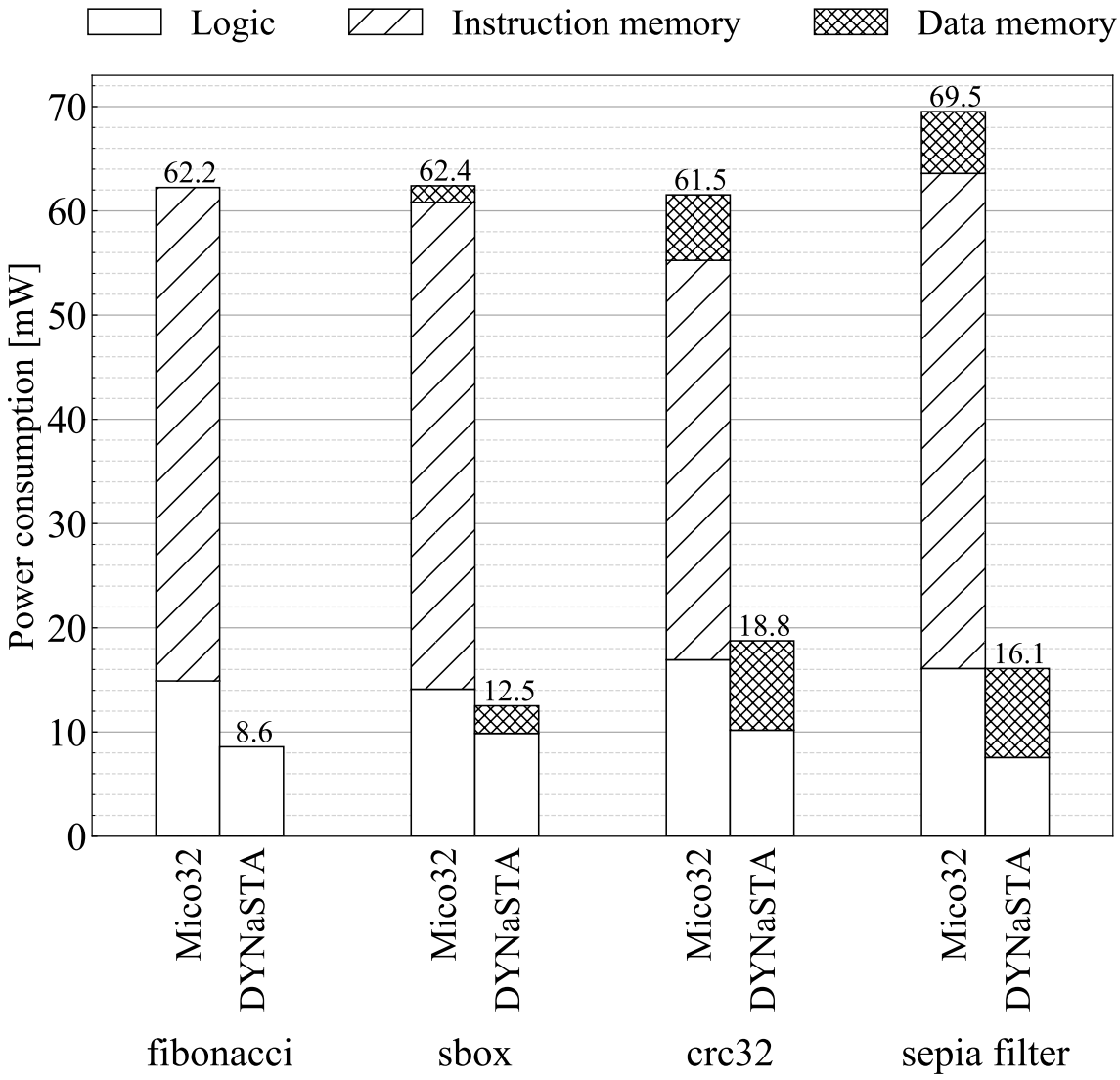



**図 2.10**　LatticeMico32 と DYNaSTA アクセラレータの消費電力見積もり比較

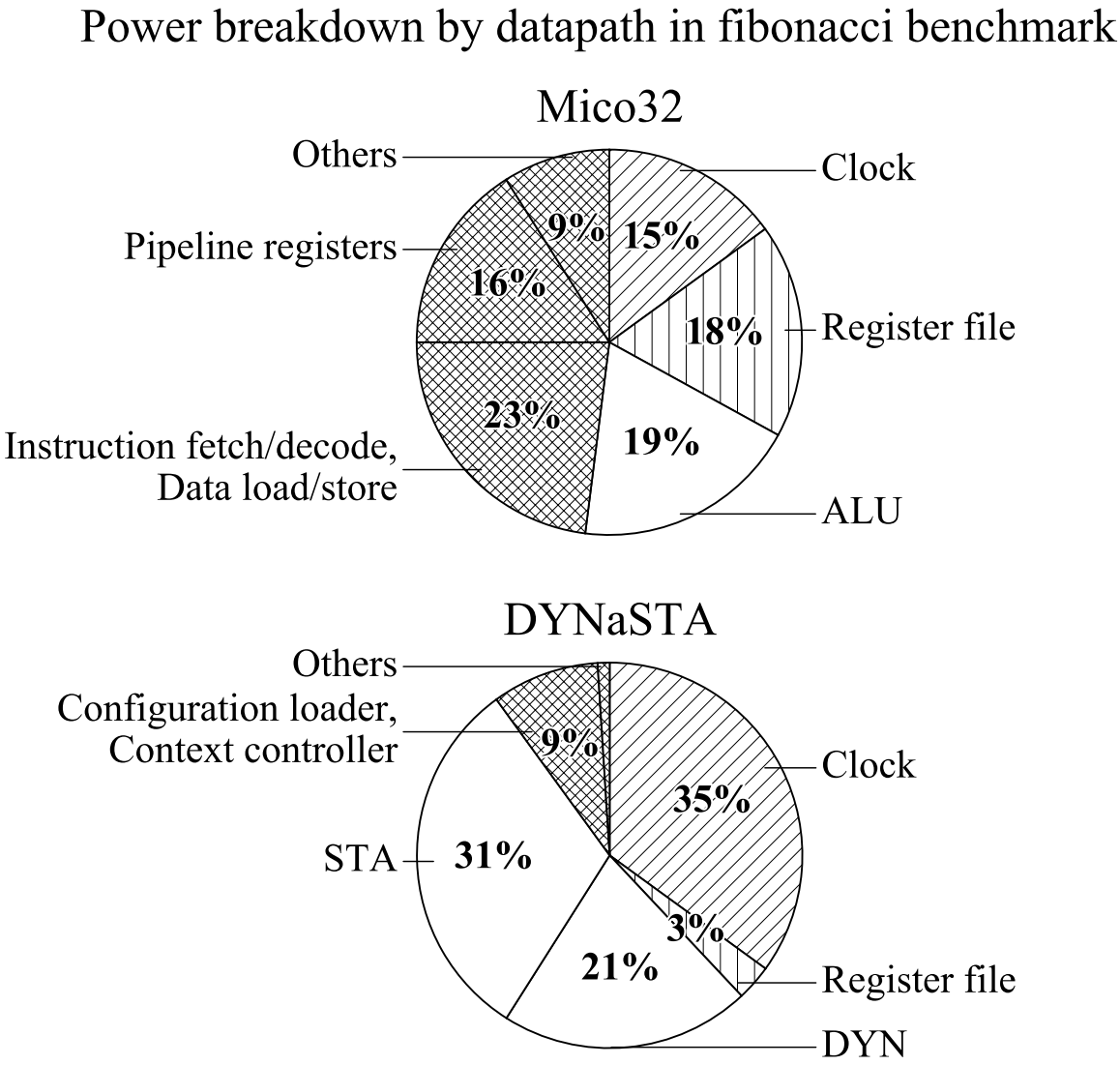



**図 2.11** フィボナッチ数計算プログラム (*fibonacci*) に要した消費電力のデータパス別内訳比較

**表 2.2**　実行時間とエネルギー効率の改善率

| Application | Processor | Time [µs] | Rate | Impr. | Energy [nJ] | Rate | Impr. |
|---|---|---|---|---|---|---|---|
| **fibonacci** | Mico32 | 2.93 | 1 | **44%** | 182 | 1 | **92%** |
| | DYNaSTA | 1.63 | 0.55 | | 13 | 0.07 | |
| **sbox** | Mico32 | 0.60 | 1 | **40%** | 37 | 1 | **88%** |
| | DYNaSTA | 0.36 | 0.60 | | 4 | 0.11 | |
| **crc32** | Mico32 | 15.11 | 1 | **27%** | 929 | 1 | **78%** |
| | DYNaSTA | 11.08 | 0.73 | | 207 | 0.22 | |
| **sepia filter** | Mico32 | 48.10 | 1 | **54%** | 3343 | 1 | **89%** |
| | DYNaSTA | 22.26 | 0.46 | | 358 | 0.10 | |

実行中は命令フェッチの必要がない。したがって、DYNaSTA アクセラレータで演算を実行した場合は LatticeMico32 で実行した場合と比べて、演算で生じる消費電力 (図中 Logic) だけでなく命令メモリアクセスで生じる消費電力 (図中 inst. memory) も大幅に削減され、69 から 86%の電力の縮小に成功した。

次に、フィボナッチ数を計算するベンチマーク・プログラム (fibonacci) で生じた電力のうちメモリアクセス以外の電力に対して、その内訳を回路の機能毎に解析した (図 2.11)。LatticeMico32 における演算回路 (図中 ALU+LD/ST Unit) は、DYNaSTA アクセラレータにおける STA と DYN と見なすことができ、これは図中の青いブロックに該当する。赤いブロックは制御回路、緑のブロックはレジスタファイル･アクセス、黄色いブロックはクロック・ラインで生じた電力であり、図から明らかなように、DYNaSTA では演算に直接関与しない消費電力が削減されたことがわかった。

また、ゲート数およびトグル率は表 2.3 であった。ゲート数で比較すると DYNaSTA アクセラレータは LatticeMico32 の 18.5 倍の規模であるが、トグル率はわずか 0.06 倍であった。これは、LatticeMico32 の演算が逐次実行であるのに対し DYNaSTA アクセラレータは並列実行であることに起因し、したがって逐次実行に必要な制御に関わる電力が削減されたと考えられる。

これらのシミュレーション結果は、本章の冒頭で述べた (1) メインメモリからの命令/データの読み出しおよびデータの書き出し、(2) レジスタファイルへの演算結果読み出し/書き出し、(3) パイプラインレジスタの駆動、のような非演算依存電力の縮小

表 2.3 論理合成後のゲート数および fibonacci 実行時のトグル率

| | | Gate count [k] | Average toggle rate [%] |
|---|---|---|---|
| **DYNaSTA** | DYN | 43.6 | 20.0 |
| | STA | 322.9 | 1.8 |
| | Others | 80.2 | 9.1 |
| | All | 446.8 | 4.9 |
| **Mico32** | | 24.1 | 76.8 |
| **Ratio (DYNaSTA / Mico32)** | | **18.48** | **0.06** |

に成功したことを示している。さらに、DYNaSTA の命令レベル並列性により実行速度も 1.4 から 2.2 倍に向上し、実行時間と平均消費電力から見積もられた電力効率は 4.5 から 13 倍に改善された (図 2.2)。

### 2.3.3 チップの試作と測定

シミュレーションにより DYNaSTA アクセラレータの有意性が示されたため、実チップの試作を行った。UMC 社の 0.18µm CMOS プロセスを利用し、チップ面積の制約上、STA のステージ数は 4 段とした (表 2.4)。また、プロセッサは実装せずアクセラレータのみを試作対象としたため、レジスタファイルはプロセッサ (LatticeMico32) と同仕様で設計し DYNaSTA に埋め込んだ (図 2.12)。

試作したチップを用いて、消費電力を測定した (図 2.13、図 2.14)。LatticeMico32 を FPGA に実装し、試作チップへのテストベクタおよびクロック信号は FPGA ボードから供給した。また、試作チップの駆動電圧として、コア電圧 3.3V および I/O 電圧 1.8V を外部電源から供給した。DYNaSTA アクセラレータからの出力信号を FPGA 経由で PC モニタし、正常に動作していることを確認した後、電源に電力アナライザを接続してチップ駆動中の消費電力を測定した。

シミュレーションで用いたベンチマークのうち fibonacci プログラムに対して、構成情報をロードして STA にマッピングする configuration phase と演算実行中の running phase の消費電力を、クロック周波数をスイープしてそれぞれ測定し、シミュレーションで得られた結果と比較した (図 2.15)。ここで、クロック供給源である FPGA の最大

表 **2.4**　試作した LSI の概要

| **Specification** | |
|---|---|
| **Technology** | UMC 0.18µm 1P6M CMOS |
| **Package** | 48-pin DIL |
| **Core area** | 1.06 mm × 1.06 mm |
| **Gate count** | 86.5 K |
| **Supply voltage** | 1.8 V core / 3.3 V IO |
| **Clock frequency** | 100 MHz |
| **Size of register file** | 32 bit × 4 word |
| **Number of stages** | 4 |
| **Number of PE per stage** | 5 |
| **Number of operatable contexts** | 6 |

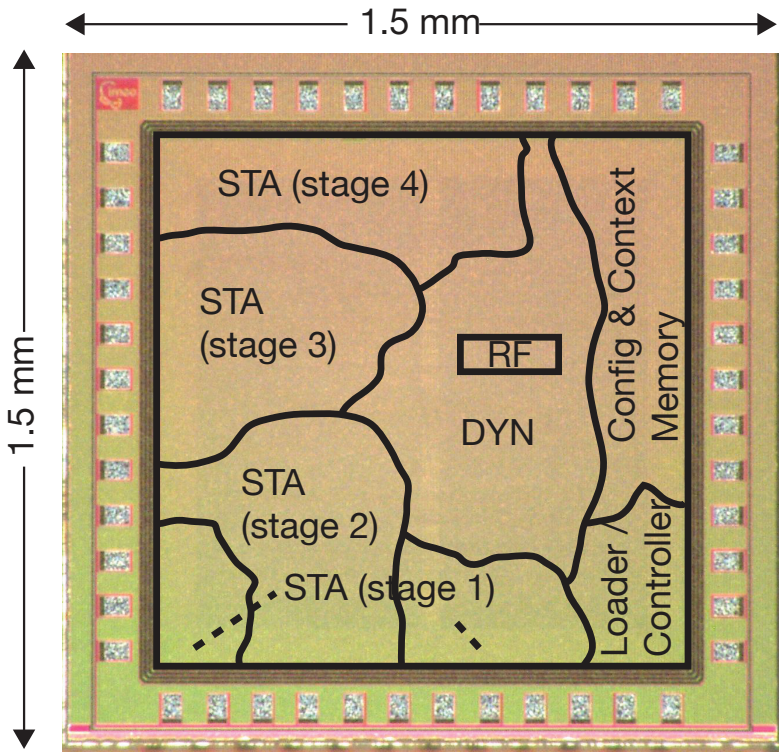



図 **2.12**　試作したチップの写真

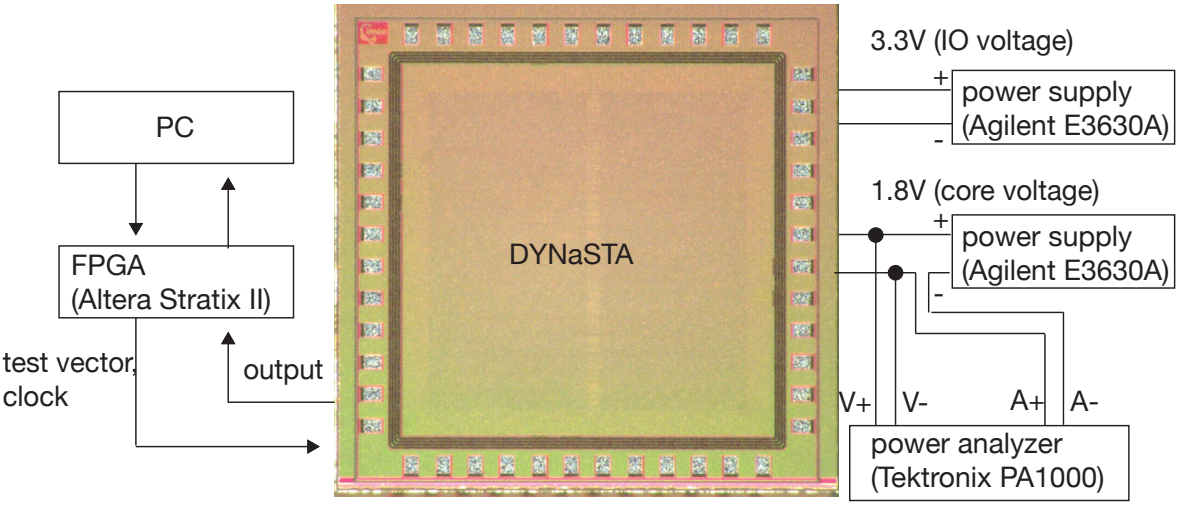



図 **2.13** 消費電力の測定環境

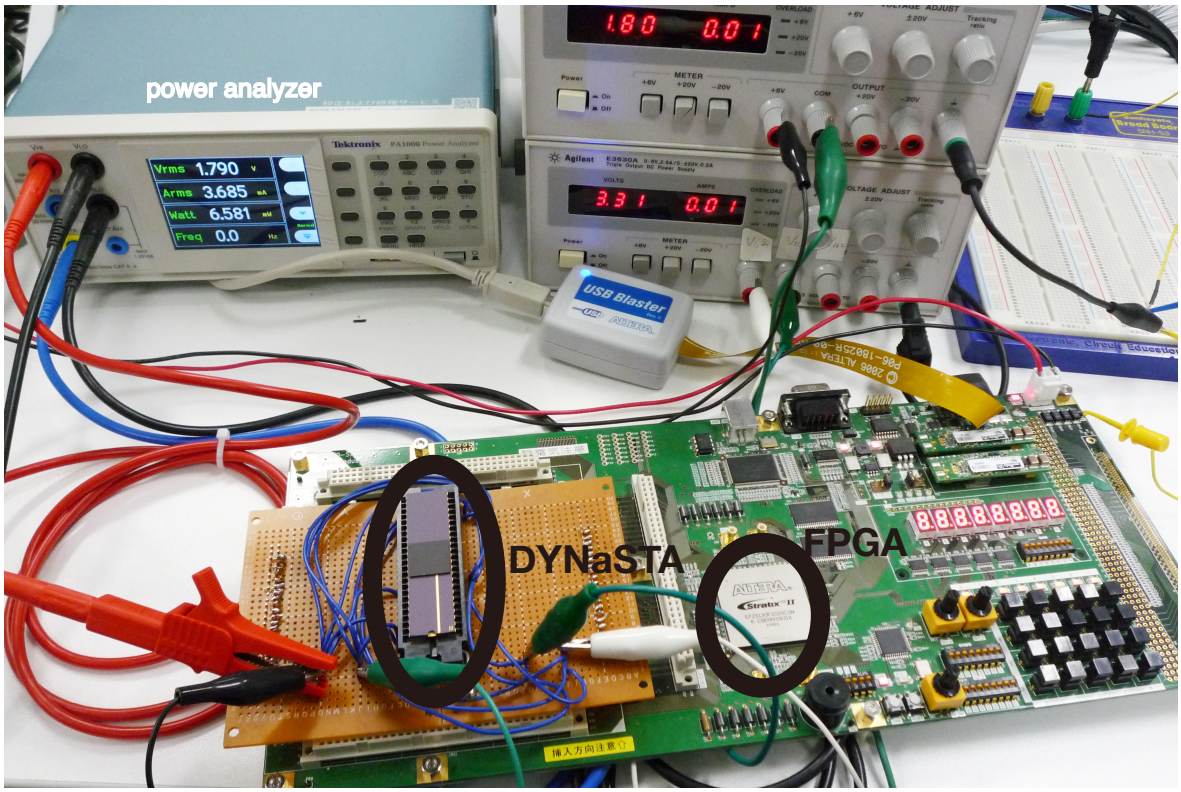



図 **2.14** 電力測定の様子

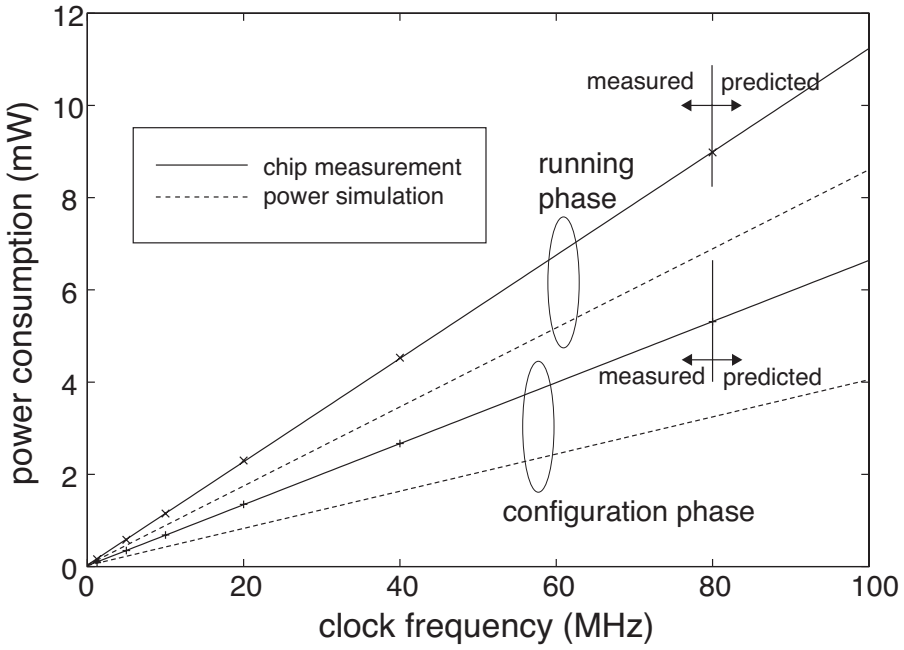



**図 2.15**　クロック周波数に対する消費電力の変化

**表 2.5**　100MHz 動作時における消費電力のシミュレーション結果と測定結果

| | **Configuration phase [mW]** | **Running phase [mW]** |
|---|---|---|
| **Simulation** | 4.03 | 8.57 |
| **Measurement (interpolated)** | 6.63 | 11.20 |

クロック周波数が 80MHz であるため、実測値に対しては線形補間して 100MHz までの消費電力をプロットしている。

100MHz における消費電力は、それぞれ表 2.5 に示す結果を得た。約 2.5mW の誤差が見られるが、configuration phase および running phase で誤差の値はほぼ同一であるため、シミュレーションで再現しきれない温度特性や寄生容量などが原因であると考えられる。今回はクロック・サイクル・ベースで消費電力を見積もったが、より低レイヤでデバイスの特性を反映したシミュレーションを行うことで、誤差は縮小されると期待される。

## 2.4 結言

本章では、電力効率と柔軟性を両立する限定的動的再構成アクセラレータ DYNaSTA について述べた。DYNaSTA アクセラレータは、動的に再構成されるデータパス (DYN: Dynamic operand forwarding matrix) と、静的な再構成のみが許される ALU アレイ (STA: Static ALU array) で構成される。STA は命令を並列処理することで電力効率を向上させ、DYN は条件分岐命令によるオペランド依存性の変化を解消するためにプログラム実行中に動的に再構成される。提案する DYNaSTA アクセラレータをクロック周波数 100MHz、CMOS 0.18µm プロセスの条件で、クロックサイクルベースのシミュレーションによる評価と、チップの試作を行った。シミュレーションの結果として、ベンチマーク・プログラムを DYNaSTA アクセラレータで実行すると、プロセッサで実行した場合と比べて消費電力は 69 から 86%が削減され、実行速度は 1.4 から 2.2 倍向上し、電力効率は 4.5 から 13 倍に改善された。本手法は、DYN と STA で構成される演算ステージを複数段積み重ねて構成されるため (図 2.16)、演算実行時にはアクティブなステージが波のように伝播するという特徴がある。したがって、非活性なステージに対してパワー・ゲーティングを施すことにより、プロセッサコア数の増加に伴い顕著となっているダーク・シリコン問題 [10] を解決しうる有効な手法と考えられる。

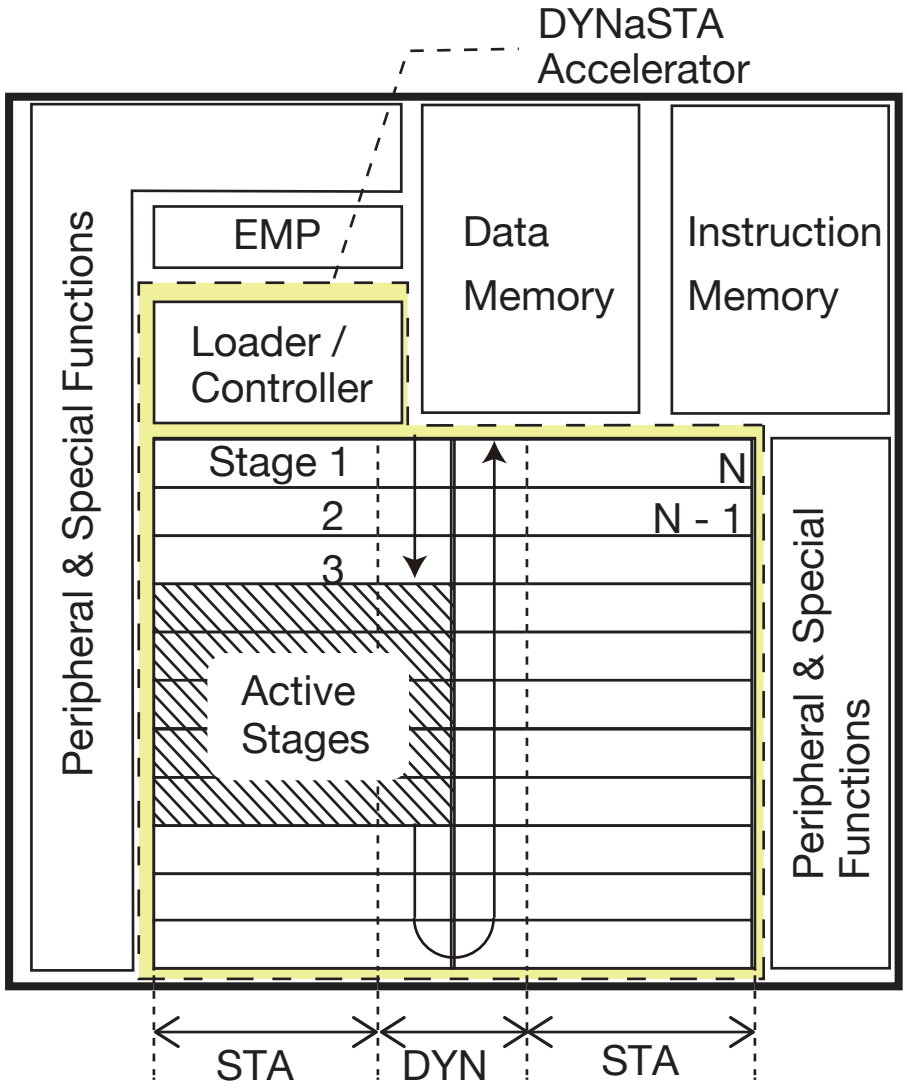



**図 2.16**　DYNaSTA SoC concept toward “ Dark Sillicon ” era.

# 参考文献

[1] R. Hameed *et al.*, "Understanding sources of inefficiency in general-purpose chips," *ACM SIGARCH Computer Architecture News*, vol. 38, no. 3, p. 37, jun 2010.

[2] M. Motomura, "A Dynamically Reconfigurable Processor Architecture," *Microprocessor Forum*, 2002.

[3] F. J. Veredas, M. Scheppler, W. Moffat, and B. Mei, "Custom implementation of the coarse-grained reconfigurable adres architecture for multimedia purposes," in *Proceedings - 2005 International Conference on Field Programmable Logic and Applications, FPL*, vol. 2005. IEEE, 2005, pp. 106–111.

[4] Yoshiki Saito, T. Sano, M. Kato, V. Tunbunheng, Yoshihiro Yasuda, Masayuki Kimura, and H. Amano, "MuCCRA-3: A low power dynamically Reconfigurable Processor Array," in *2010 15th Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, jan 2010, pp. 377–378.

[5] S. Swanson and M. B. Taylor, "Greendroid: Exploring the next evolution in smartphone application processors," *IEEE Communications Magazine*, vol. 49, no. 4, pp. 112–119, apr 2011.

[6] N. Ozaki, Y. Yasuda, M. Izawa, Y. Saito, D. Ikebuchi, H. Amano, H. Nakamura, K. Usami, M. Namiki, and M. Kondo, "Cool Mega-Arrays: Ultralow-Power Reconfigurable Accelerator Chips," *IEEE Micro*, vol. 31, no. 6, pp. 6–18, nov 2011.

[7] Lattice, "LatticeMico32 Open, Free 32-Bit Soft Processor," 2018.

[8] M. R. Guthaus, J. S. Ringenberg, D. . Ernst, T. M. Austin, T. Mudge, and R. B. Brown, "MiBench: A free, commercially representative embedded benchmark suite," in *Proceedings of the Fourth Annual IEEE International Workshop*

*on Workload Characterization. WWC-4 (Cat. No.01EX538).* IEEE, 2001, pp. 3–14.

[9] M. G. Katevenis, “3.1 On-Chip SRAM,” 2003.

[10] H. Esmaeilzadeh, E. Blem, R. St. Amant, K. Sankaralingam, and D. Burger, “Dark Silicon and the End of Multicore Scaling,” *IEEE Micro*, vol. 32, no. 3, pp. 122–134, may 2012.

# 第3章　単純ベイズ法を用いた動的分岐予測器

## 3.1　緒言

分岐予測器の予測精度はCPUの電力効率を大きく左右する一要因である。昨今のRISCアーキテクチャに基づくCPUは、およそ10数段におよぶ深い命令パイプラインを有する。もし分岐の予測に失敗すると、パイプラインを一時停止し、処理中の命令のクリアと適切な命令およびデータを読み出し直す必要が生じる。これによるスループットの低下や消費電力の増大は、パイプラインが深ければ深いほど深刻化する。

高い精度で分岐方向を予測するために、過去の分岐パターンから次の分岐方向を統計的に推論する手法が広く用いられている [1–3]。その中で、ニューラルネットワークを用いた機械学習による予測手法が国際的な分岐予測コンテストで高い成績を示し続けている [4]。本研究では、ニューラル分岐予測器の分岐予測コアをベイズ統計学的機械学習手法に置換した分岐予測器 [5] で置き換える。このナイーブベイズ分岐予測器をソフトコアCPUにレジスタ転送レベルで組み込み、その消費電力削減効果を検証する。

## 3.2　パーセプトロン分岐予測器

人工ニューロン [6–8] で構成されるニューラルネットワークは、あるデータセットからデータの特徴を学習することができ、 画像や音声、文章、統計量などに対するパターン認識用途で広く用いられている [9–11]。CPUにおける条件分岐命令の分岐方向もまた、その履歴をとると時系列に相互関係のあるデータセットとみなすことができ、ニューラルネットワークを用いた予測が有効であることが示されている [12]。

最も単純なパーセプトロン分岐予測器のアーキテクチャを図3.1に示す。命令アドレスの下位$n$bitに対して$2^n$個の別個のパーセプトロンを有し、各パーセプトロンの重みはカウンタで表される。命令アドレスに対応する重みとFIFO(First-In First-Out)で格納されるグローバル分岐履歴 (GBH: Global Branch History) から、人工ニューロンの

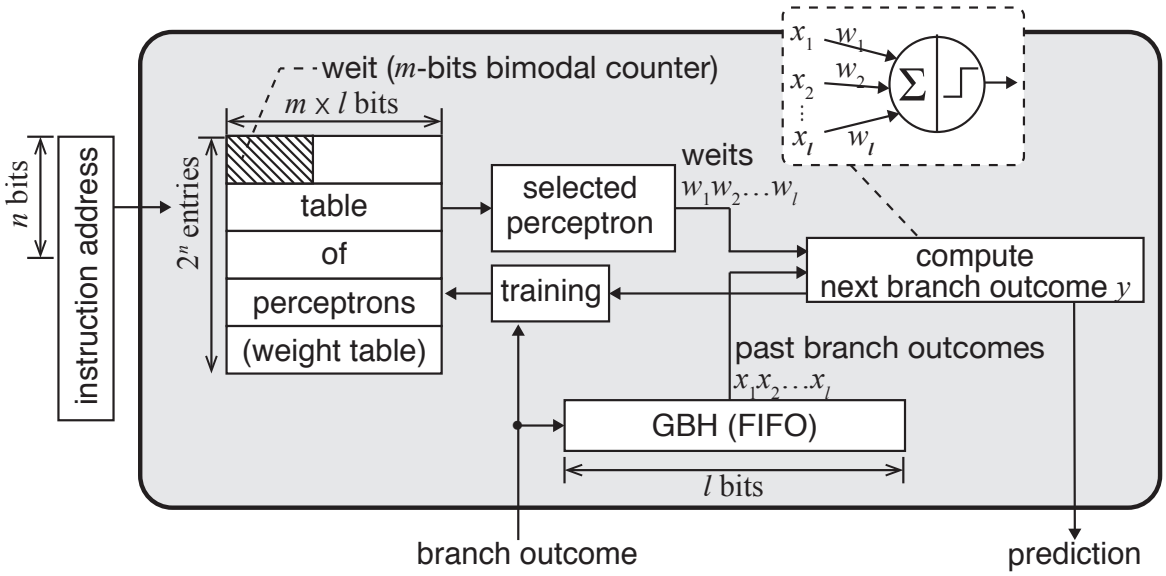



**図 3.1**　パーセプトロン分岐予測器 [12].

信号伝達的に次の分岐方向を予測し、予測結果と正しい結果から重みを学習する。本研究で用いるナイーブベイズ分岐予測器は、このパーセプトロン分岐予測器と同様にグローバル分岐履歴を特徴量として扱うが、推論および学習機構をベイズ統計学的手法に置換する。

## 3.3　ナイーブベイズ分岐予測器

### 3.3.1　単純ベイズ法を用いた分岐予測

単純ベイズ法はベイジアンネットワークの最も簡単なモデルであり、複数の子ノードと単一の親ノードで構成される。各子ノードは特徴変数 $x$ を、親ノードはカテゴリ変数 $y$ を表され、子ノードと親ノードは条件付き確率表 (CPT: Connditional Probability Table) で関係付けられる。計算を単純化するために、特徴変数間に独立性を仮定しており、この点が単純ベイズ法の特徴でもある。カテゴリ変数が $y = c$ である事後確率 $P(y = c \mid x_1, x_2, ..., x_n)$ を、条件付き確率表に基づいて次のように計算することで、所望の推論を行う。

$$P(y = c \mid X) = \frac{P(x_1 \mid y = c)P(x_2 \mid y = c)...P(x_n \mid y = c)P(y = c)}{P(x_1, x_2, ..., x_n)} \tag{3.1}$$

単純ベイズ法を用いた分類器は E メールのスパムフィルタなど広範囲に応用 [13, 14]

されており、分岐予測を行う場合には、説明変数 $X = \{x_1, x_2, ..., x_n\}$ が過去の分岐の履歴に、カテゴリ変数 $y$ が予測するべき次の分岐方向に対応する 3.2。ここで、各変数 $x_i$ および $y$ は、分岐条件が不成立のとき 0 を、成立の 1 をとる。したがって、事後確率 $P(y = 0 \mid X)$ と $P(y = 1 \mid X)$ の大小比較により、次の分岐の成否が予測される。

ナイーブベイブ分岐予測の回路アーキテクチャを、図 3.3 に示す。一般的に、単純ベイズ法は小数点を用いた大量の除算や乗算を要するため計算負荷が高く、回路化すると 5 から 10 クロックサイクルのレイテンシが生じてしまう [13, 14]。しかし、最もシンプルな命令パイプラインは

- IF : Instruction Fetch (命令フェッチ)
- ID : Instruction Decode (命令デコード)
- EX : Execution (実行)
- MA : Memory Access (メモリアクセス)
- WB : Write Back (データ書き出し)

のように 5 段程度で構成することができるため、汎用性を高めるためには、分岐の予測は IF から EX までの 2 クロックサイクル内で完了されなければならない。そこで、次に述べるように確率計算を簡単化して低遅延な分岐予測アーキテクチャを設計した。

まず、条件付き確率表の各尤度 $P(x_i \mid y)$ および事前確率 $P(y)$ は、各変数が 0 または 1 しかとらないことから

$$\begin{cases} P(y = 0) = 1 - P(y = 1) \\ P(x_i = 0 \mid y = c) = 1 - P(x_i = 1 \mid y = c) \end{cases} \tag{3.2}$$

の関係が成り立つため、$P(x_i = 0 \mid y = c) + P(x_i = 1 \mid y = c) = 2^n - 1$ の条件を満たす $n$bit アップ/ダウン飽和カウンタで実装される。

事前確率表 $P(y)$ を構成する $p(y = 0)$ および $p(y = 1)$ は、分岐命令が実行された時、その分岐命令の成否 $t$ にしたがって次のように更新される。

**if** $t = 0$ **then**
  $p(y = 0) \leftarrow p(y = 0) + 1$
  $p(y = 1) \leftarrow p(y = 1) - 1$
**end if**

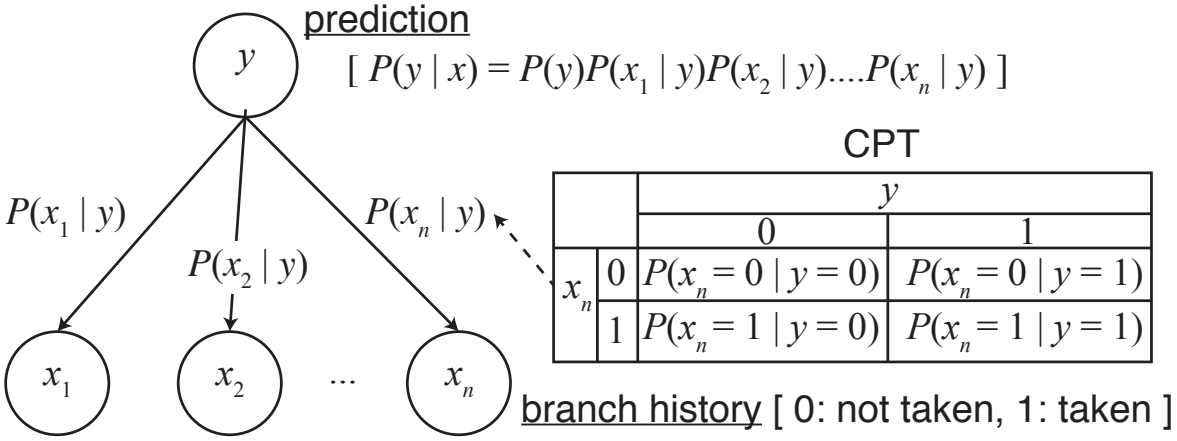



**図 3.2**　単純ベイズ法を用いた分岐予測：変数 $x_i$ は過去の分岐履歴を、$y$ は予測される次の分岐を表す。

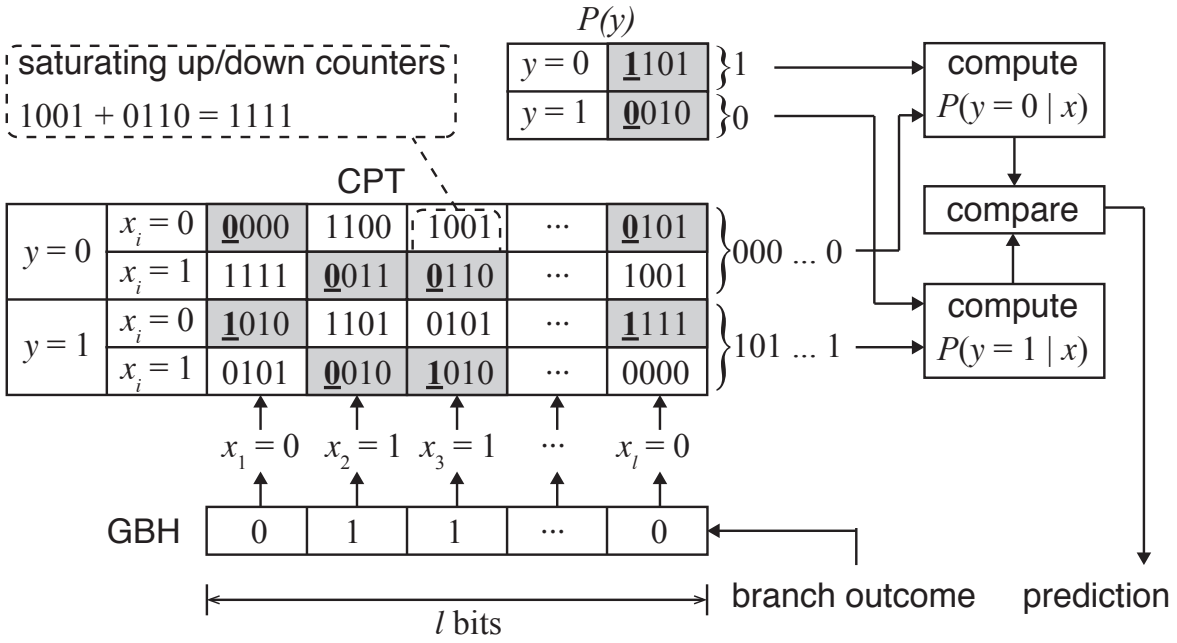



**図 3.3**　条件付き確率表から事後確率を計算するためのハードウェア・アーキテクチャ

$t = 1$ の場合も同様である。

グローバル分岐履歴はシフトレジスタで実装され、各履歴 $x_i$ に対して 4 個のカウンタ $p(x_i, y)$ が存在する $(x_i = 0 or 1, y = 0 or 1)$。これらは $P(y)$ と同様に以下のような更新ルールが適用される。

**if** $t = 0$ **then**
  **if** $x_i = 0$ **then**
    $p(x_i = 0, y = 0) \leftarrow p(x_i = 0, y = 0) + 1$
    $p(x_i = 1, y = 0) \leftarrow p(x_i = 1, y = 0) - 1$
  **end if**
**end if**

これらのカウンタを用いて事後確率 $P(y = 0 \mid X)$ および $P(y = 1 \mid X)$ が計算される。ここで、事後確率は式 3.1 にしたがって計算されるが、分岐予測では各変数が 2 値であるため $P(y = 0 \mid X) + P(y = 1 \mid X) = 1$ が成立し、予測には厳密な確率値ではなく大小関係のみがわかれば十分である。

したがって、まず各事後確率で共通の分母を省略し、

$$P(y \mid X) \propto P(y)P(x_1 \mid y)P(x_2 \mid y)...P(x_n \mid y) \tag{3.3}$$

さらに、次式のように両辺の対数をとる。

$$\log P(y \mid X) \propto \log P(y) + \log P(x_1 \mid y)... + \log P(x_n \mid y) \tag{3.4}$$

このようにして、除算と乗算を取り除き加算のみに置き換えることで、回路化のために計算量を削減する。

さらに、式 3.4 の対数同士の加算を、各カウンタの最上位ビット同士の加算で置き換えられる [5] ことで加算器を縮小する。ここで、最上位ビット同士の加算は、すなわち 1 がセットされたビットをカウントする処理と等価である。そこで読み出されたビット列をいくつかの小さなまとまりに分割し、“1” の個数をカウントするルックアップテーブル (LUT: Look-Up Table) に入力する。その出力をパイプライン化された小さな加算器で加算を行い、事後確率を得る 3.4。これらの工夫により回路面積および遅延を縮小することで、次節で述べる CPU への組み込みに適したアーキテクチャを設計した。

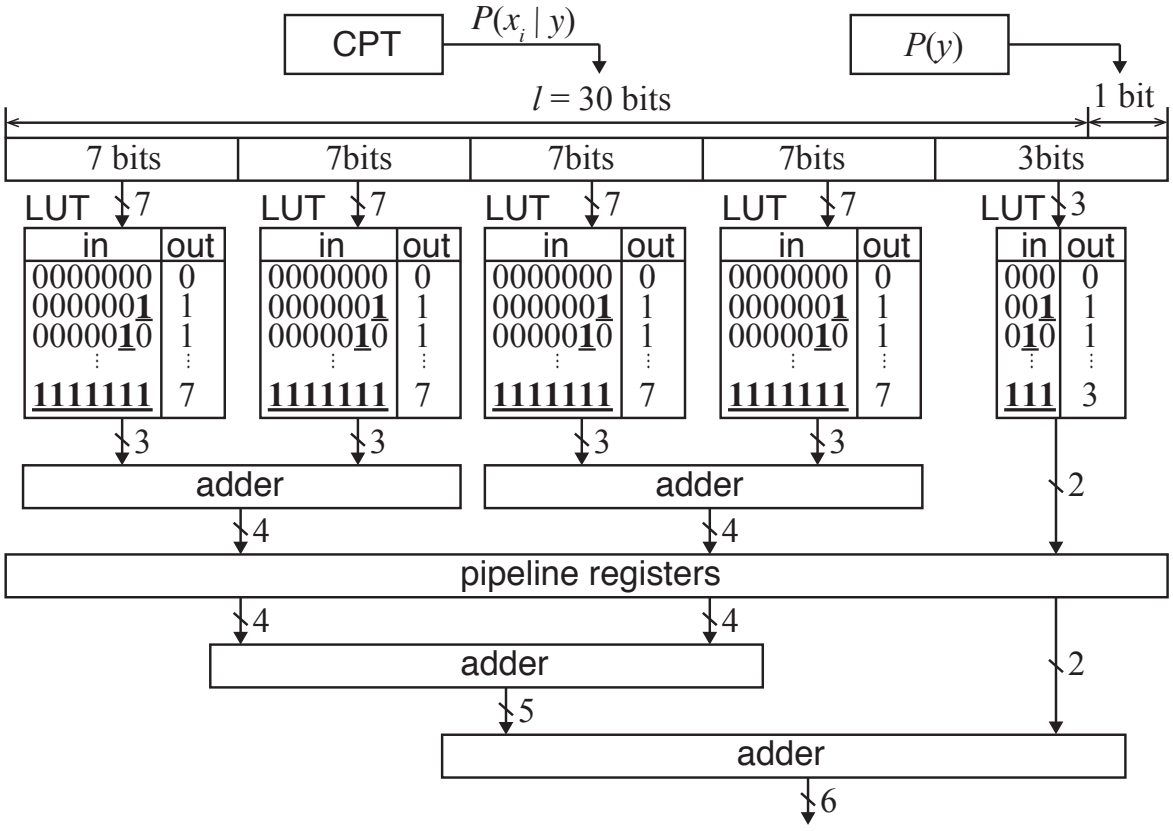



**図 3.4**　大小比較のための事後確率 $P(y = c \mid x_1, x_2, ..., x_n)$ はルックアップテーブルとパイプライン化された加算器で計算される。

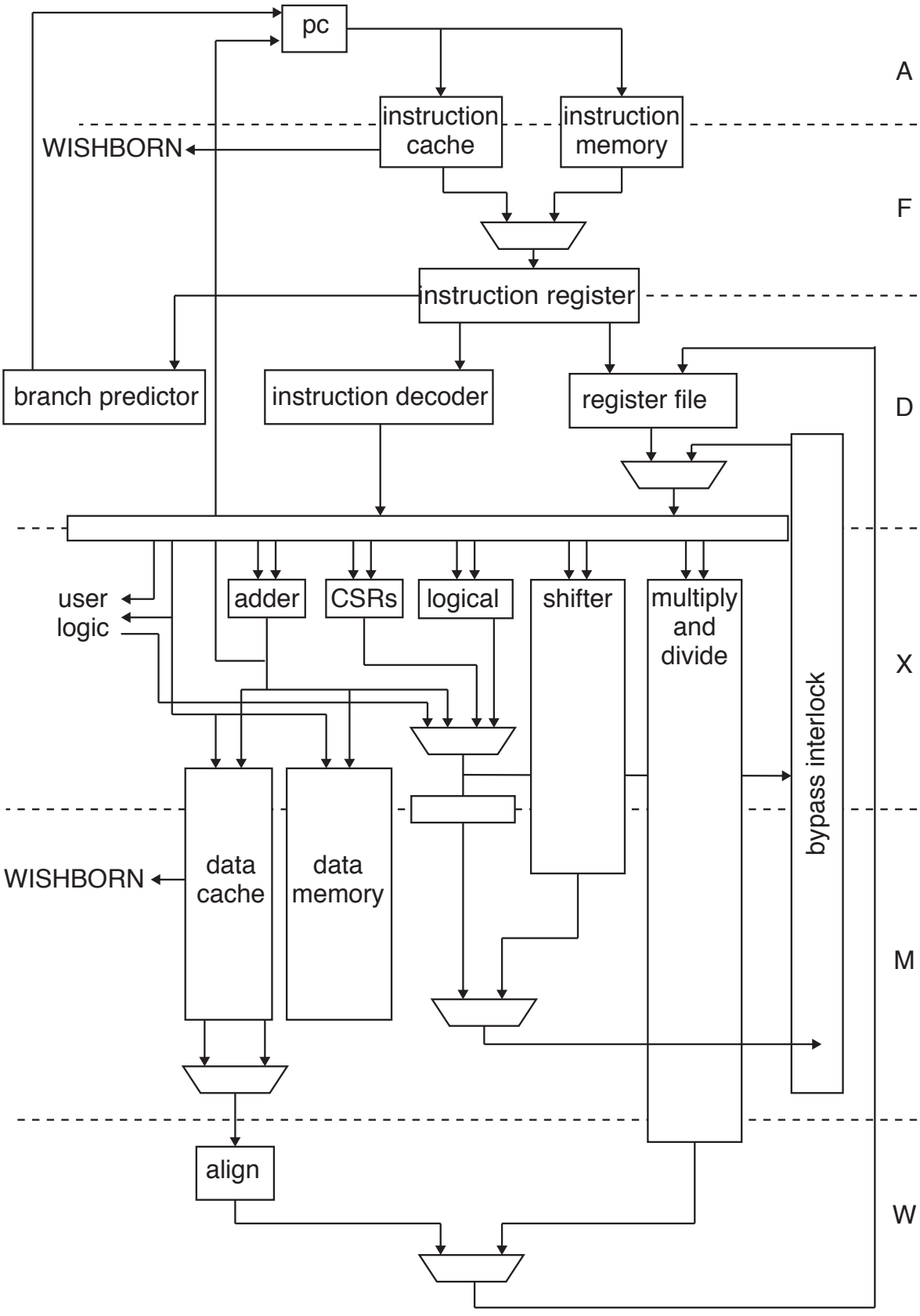



**図 3.5** LatticeMico32 のブロック図と命令パイプライン [15].

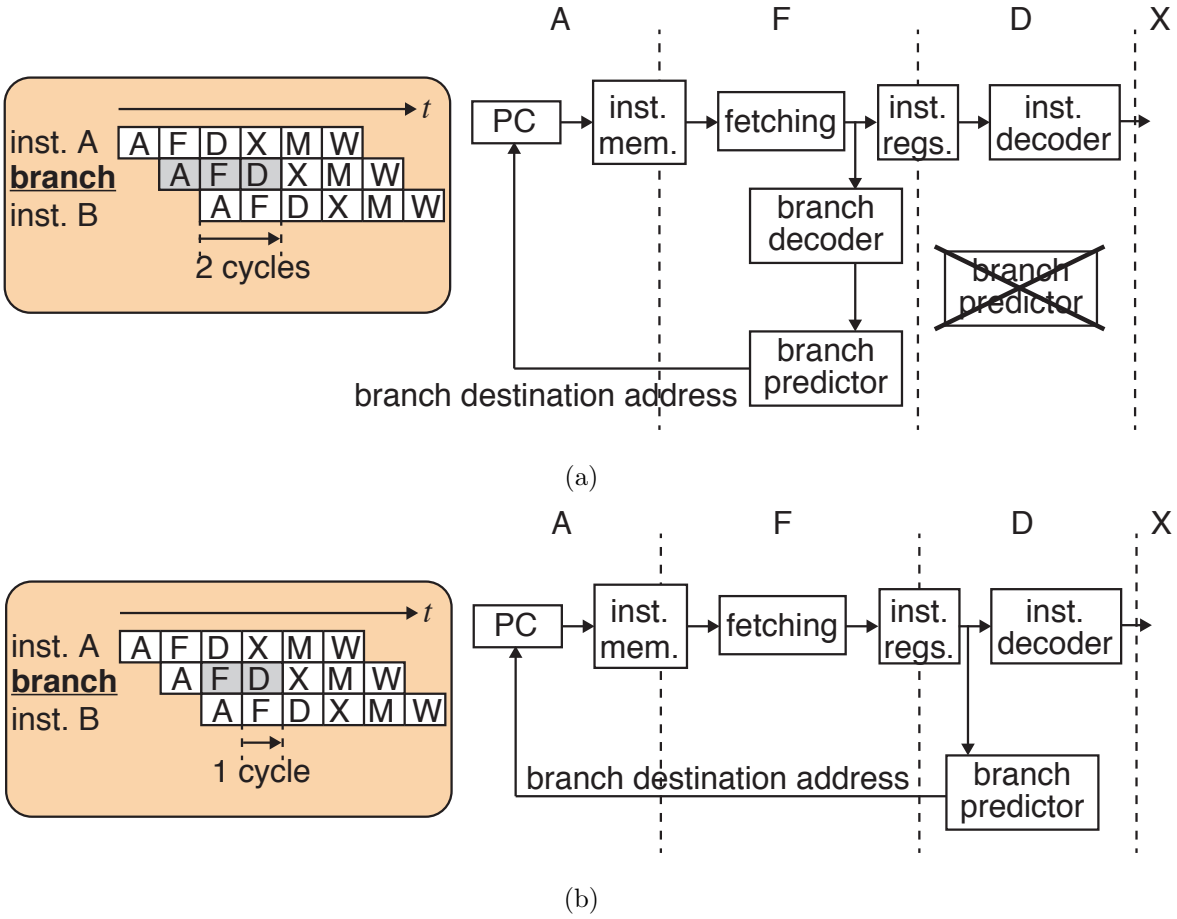



**図 3.6**　(a)LatticeMico32 は分岐命令を D ステージで予測し、X ステージで実行する。(b) 分岐予測器を F ステージに移動させて実行まで 2 クロックサイクルの余裕を確保した。

### 3.3.2　ナイーブベイズ分岐予測器の CPU への適用

以上のように設計したナイーブベイズ分岐予測器をソフトコア CPU である LatticeMico32 [15] に埋め込むために、まず LatticeMico32 のデータパスを調査した。LatticeMico32 は全部で 6 段の命令パイプラインを有し 3.5、分岐の予測は D ステージで実行される。この場合、D ステージで命令がデコードされてから X ステージまで 1 クロックサイクルの余裕しかなく、設計したナイーブベイズ分岐予測器の遅延設計と適合しない。そこで、F ステージにフェッチされた命令が分岐命令かどうかを判断する分岐命令デコーダを設置し、分岐予測器はこのデコーダの直後に設置することで、予測から実行まで 2 クロックサイクルを確保した 3.6。

ナイーブベイズ分岐予測器のアーキテクチャを図 3.7 に示す。条件付き確率表および事前確率表は、命令アドレスの下位 $n$ ビットをハッシュ値として $2^n$ のページ数を持

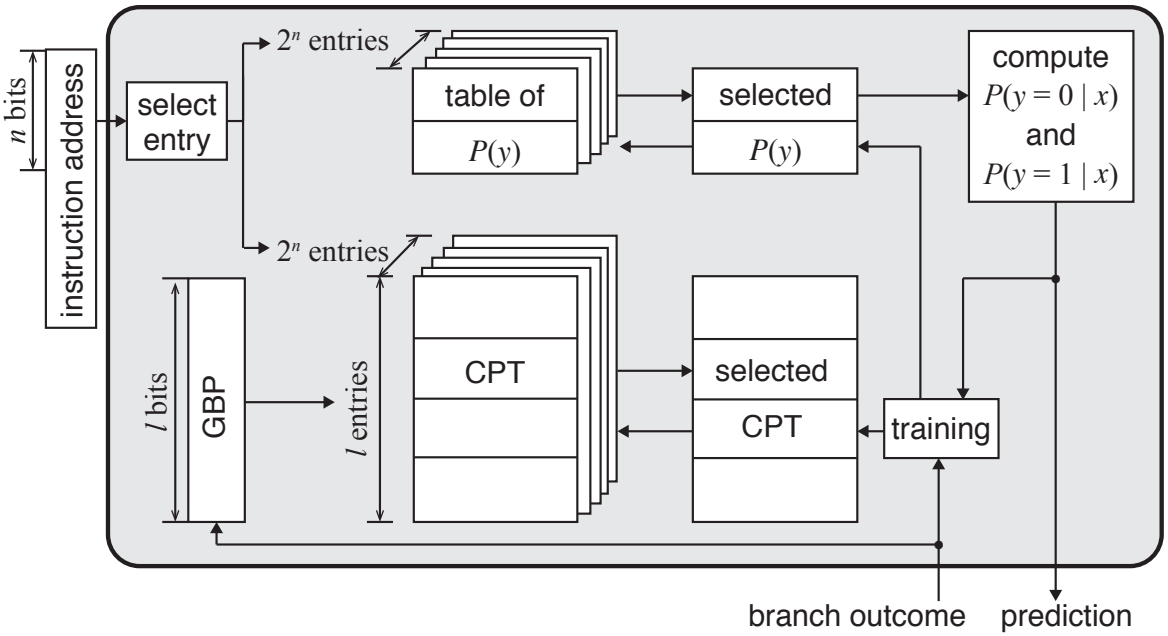


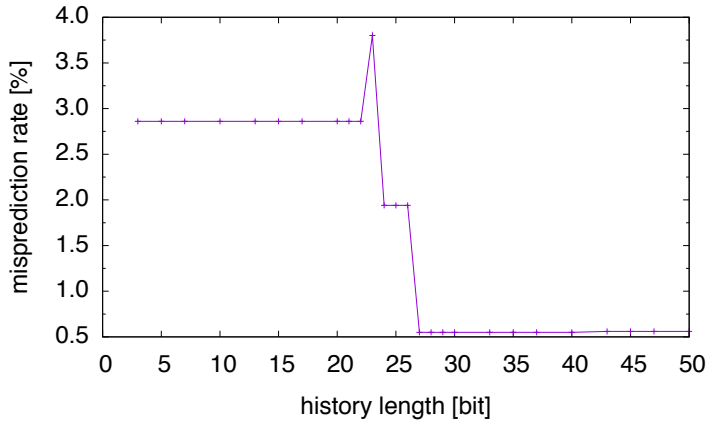

**図 3.7** ナイーブベイズ分岐予測器 (NBBP) のアーキテクチャ

つ。これは、パーセプトロン分岐予測器において同様に $2^n$ 個のパーセプトロンを用いていたことを参考にした。分岐命令がフェッチされると、命令アドレスの下位ビットとグローバル分岐履歴にしたがって事前確率 $P(y)$ および尤度 $P(x_i \mid y)$ が読み出され、事後確率 $P(y = 0 \mid X)$ と $P(y = 1 \mid X)$ の大小比較を行い、分岐成否を予測する。分岐命令が実行されて真の結果が判明すると、GBH および事前確率、尤度が更新される。

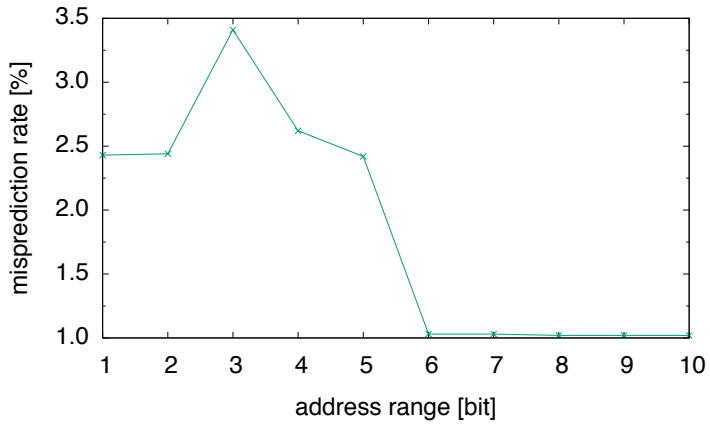
## 3.4 評価

### 3.4.1 分岐予測精度

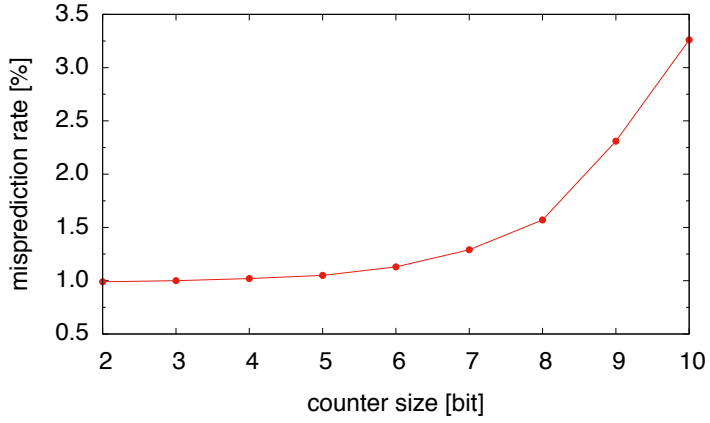
ナイーブベイズ分岐予測器の予測の精度と回路規模は、条件付き確率表およびグローバル分岐履歴のサイズで決定される。そこで、ベンチマーク・セット [16] の中から最も単純なプログラムを選び、分岐履歴の長さ $l$、命令下位アドレスの長さ $n$、カウンタのビット数 $m$ をそれぞれ変化させて予測精度をシミュレーションし、最適なパラメータを探索した。各パラメータに対する予測ミス率は図 3.8 のようになった。履歴およびアドレスのサイズを大きくすると、より密な粒度で特徴を学習できるため、ミス率が低下した。一方で、確率カウンタのビット数を増やすとミス率は上昇した。これは、カウンタの最上位ビットのみを予測に用いているために、ビット数が増えてカウンタ値




(a) address range = 8, counter size = 4




(b) history length = 30, counter size = 4




(c) history length = 30, address range = 8

**図 3.8**　各パラメータに対する予測ミス率 (a) グローバル分岐履歴の長さ、(b) 命令アドレスの範囲、(c) 確率カウンタのサイズ

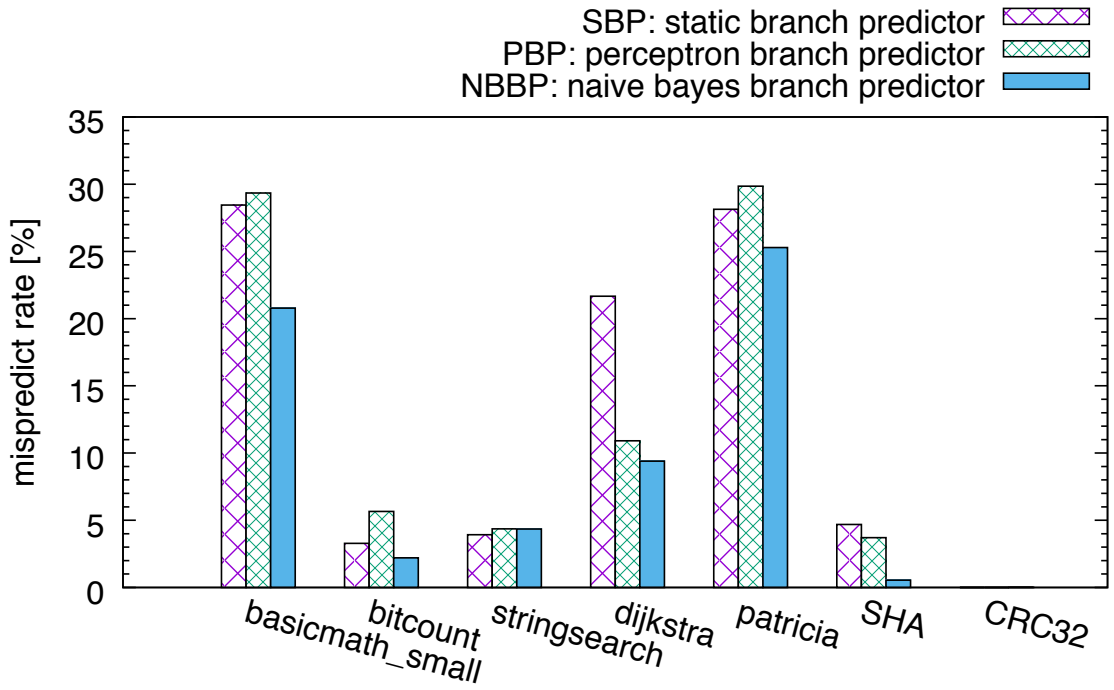



**図 3.9** ナイーブベイズ分岐予測器とパーセプトロン分岐予測器、静的分岐予測器の予測精度比較

のダイナミックレンジが広くなると十分に学習が進まなかったためと考えられる。これらの結果から、グローバル分岐履歴の長さを 30bit、命令アドレスの範囲を 8bit、カウンタのサイズを 4bit と設定した。

次に、ナイーブベイズ分岐予測器とパーセプトロン分岐予測器、静的分岐予測器の予測精度を比較した。静的分岐予測器は、オリジナルの LatticeMico32 に組み込まれている分岐予測器である。分岐履歴の学習は行わず、条件分岐命令の種類に応じて、後方条件分岐は不成立、前方条件分岐は成立と、一意に分岐の成否を予測する。シミュレーションに用いた 7 種のベンチマーク・プログラムのうち、5 種に対してナイーブベイズ分岐予測器が最も良い精度を示した (図 3.9)。これは、ナイーブベイズ分岐予測器は、パーセプトロン分岐予測器と比べて線形分離不可なデータに対しても十分に学習しうるためだと考えられる。また、他の 2 種に対しては静的分岐予測器が最高精度を示したが、これらのプログラムはより簡単な構造であったと考えられる。

予測精度が高ければ高いほど実行時間は短縮される (図 3.10)。本研究ではシンプルなベンチマークとシンプルな CPU を採用したためわずかな向上にとどまったが、より複雑なベンチマークプログラムや深いパイプラインを有する CPU を用いれば、より顕

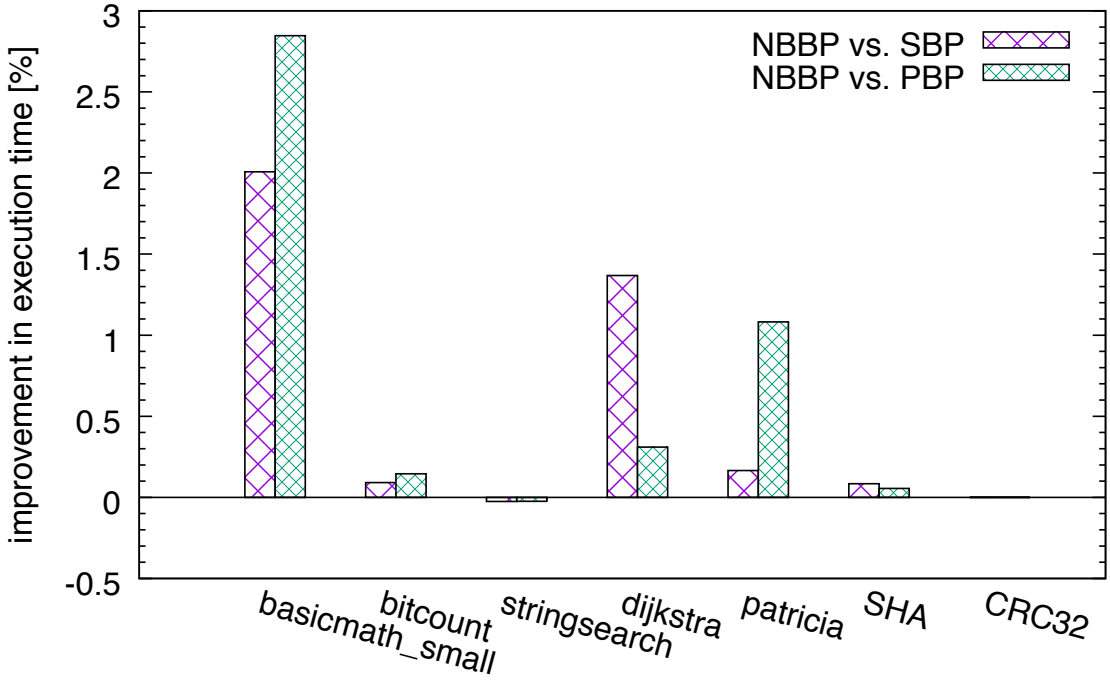



**図 3.10**　各ベンチマークプログラムに対する実行時間の改善率

著な効果が得られると期待される。

### 3.4.2　消費電力

次に、ナイーブベイズ分岐予測器および静的分岐予測器を搭載した各 LatticeMico32 に対して、クロックサイクル・ベースのシミュレーションにより得られたトグル率とメモリアクセス回数から消費電力を見積もった。シミュレーションに際し、プロセスを 0.18 µm CMOS、コア電圧を 1.8 V、クロック周波数を 100 MHz として論理合成を行なった 3.1。LatticeMico32 を用いると、図 3.11 のようなプロセッサが想定されるが、このうち LatticeMico32 の CPU コア電力 (ロジック電力) および命令メモリとデータメモリへのアクセスで生じる電力 (メモリ電力) のみを評価対象とする。ロジック電力は Synopsys 社の Power Compiler により解析する。ただし、ナイーブベイズ分岐予測器の条件付き確率表および事前確率表はレジスタで実装するには規模が大きく、実用上は SRAM として実装されると考えられるため、電力解析の対象からは除外する。かわりに、各確率表へのアクセス回数と、SRAM へのアクセスで生じる電力データ [17] から、そのアクセス消費電力を求め、ロジック電力に加えた。また、命令メモリおよび

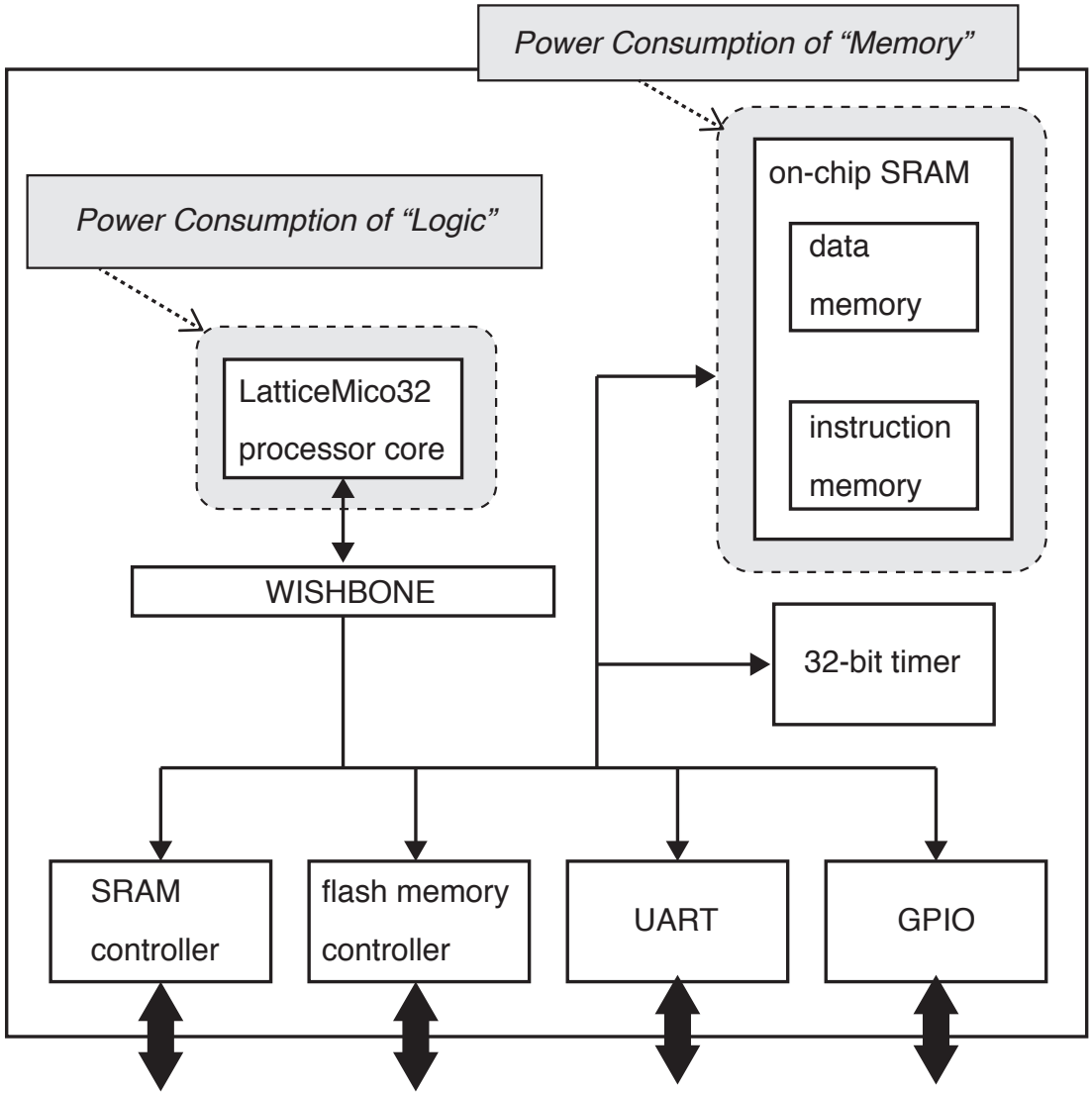



**図 3.11** LatticeMico32 を用いたプロセッサの想定図。CPU コアの消費電力とメモリアクセスの消費電力を見積もった。

**表 3.1**　論理合成時の各条件

| **Specification** | |
|---|---|
| **Technology** | UMC 0.18 µm CMOS |
| **Core area** | 3.0 mm × 3.0 mm (CPT accounts for 70 %) |
| **Supply voltage** | 1.8 V |
| **Clock frequency** | 100 MHz |
| **Size of CPT** | 122 kbits |
| **Size of $P(y)$ table** | 2 kbits |
| **Size of instruction memory** | 800 kByte |
| **Size of data memory** | 800 kByte |

**表 3.2**　ベンチマーク・プログラム basicmath_small を 1 回実行したときの消費電力見積もり

| | **Logic** | | | **Memory** | | | |
|---|---|---|---|---|---|---|---|
| | $P_{logic}$ [mW] | time [ms] | $W_{logic}$ [mJ] | $P_{mem}$ [mW] | access [ktimes] | $W_{mem}$ [mJ] | total [mJ] |
| **SBP** | 12.7 | 3.03 | **0.039** | 70 | 295 | **20.7** | **20.73** |
| **NBBP** | 41.9 | 2.97 | **0.124** | 70 | 289 | **20.3** | **20.42** |

データメモリについても同様に、シミュレーションで得られたアクセス回数と SRAM 電力データ [17] から、その消費電力を求めてメモリ電力とした。

ベンチマーク・プログラムのうち basicmath_small を 1 回実行した際の消費電力は表 3.2 のようであった。表のうち、“Logic” は前述したロジック電力を、“Memory” は同様にメモリ電力を意味する。ナイーブベイズ分岐予測器は、静的分岐予測器と比較して約 1.5 倍の回路規模を要する (表 3.3)。したがって、予測精度の改善により実行時間は短縮された (表 3.2 中の time) にも関わらず、ロジック電力は増大した。一方で、メモリアクセス回数 (表 3.2 中の access) の縮小によりメモリ電力が減少されたことで、全体の消費電力は低下した。

この結果から、プログラムを $n$ 回繰り返し実行したときの消費電力を、それぞれ次式によって計算した。

$$W_{logic}\ [\mathrm{J}] = P_{logic}\ [\mathrm{W/s}] \times \text{execution time [s]} \times \text{iteration}$$

$$W_{mem}\ [\mathrm{J}] = P_{mem}\ [\mathrm{W/time}] \times \text{access freqency [time]} \times \text{iteration}$$

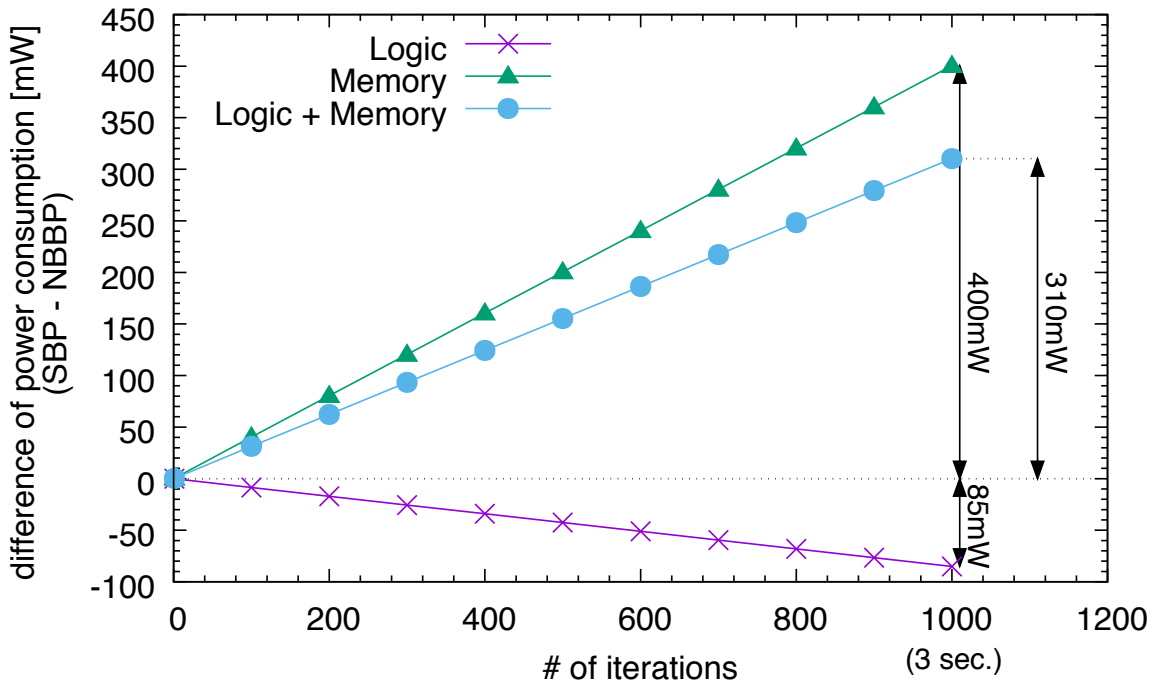



**図 3.12** ベンチマーク・プログラムを繰り返し実行した場合の消費電力差分：0 より下は静的分岐予測器の方が低電力で、0 より上はナイーブベイズ分岐予測器の方が低電力であることを示す。

**表 3.3** 各分岐予測器を LatticeMico32 に組み込んだ場合のゲート数およびコア面積

| | **Gate count** | **Circuit area** [$mm^2$] |
|---|---|---|
| **Mico32 with SBP** | 30 6k | 0.287 |
| **Mico32 with PBP** | 44 8k | 0.382 |
| **Mico32 with NBBP** | 51 4k | 0.482 |

計算結果をプロットすると (図 3.12)、繰り返し回数に対する消費電力削減効果は、

$$\text{difference of power consumption [mW]} = 0.32n$$

となり、すなわち反復回数が増すほどメモリ電力の減少による消費電力削減の恩恵を受け、プログラムを 1000 回 (約 3 秒間) 実行した際には 320 mW の電力が削減される見込みを得た (図 3.12)。

## 3.5　結言

本章では、分岐予測の高精度化による消費電力の削減効果について述べた。分岐予測を高精度化させると命令パイプラインがストールしづらくなり、メモリへのアクセス回数も少なくなるため、無駄な電力の発生を抑制させることができる。一方で、高精度な分岐予測器による回路の大規模かは電力の増加も招く。

本研究では、単純ベイズ法を用いた学習型分岐予測器を設計し、ソフトコア CPU にレジスタ転送レベルで組み込んでシミュレーションを行なった。このナイーブベイズ分岐予測器は、パーセプトロン分岐予測器や静的分岐予測器よりも高い予測精度を示した。また、静的分岐予測器と比べて回路規模が大きいため CPU 自体の消費電力は増加してしまうが、一方で予測の高精度化によるメモリアクセス電力の削減が、この電力オーバーヘッドを上回り、システム全体の消費電力は低減されることがわかった。本研究では簡素なベンチマークと CPU を用いたが、より実用的なプログラムや複雑な CPU を用いた場合には、この電力削減効果はより顕著になると考えられる。

# 参考文献

[1] T.-Y. Yeh and Y. N. Patt, "Two-level adaptive training branch prediction," in *Proceedings of the 24th annual international symposium on Microarchitecture - MICRO 24*. New York, New York, USA: ACM Press, 1991, pp. 51–61.

[2] M. Evers, P.-Y. Chang, and Y. N. Patt, "Using hybrid branch predictors to improve branch prediction accuracy in the presence of context switches," in *Proceedings of the 23rd Annual International Symposium on Computer Architecture*, ser. ISCA '96. New York, NY, USA: ACM, 1996, pp. 3–11.

[3] C.-C. Lee, I.-C. K. Chen, and T. N. Mudge, "The bi-mode branch predictor," in *Proceedings of the 30th Annual ACM/IEEE International Symposium on Microarchitecture*, ser. MICRO 30. Washington, DC, USA: IEEE Computer Society, 1997, pp. 4–13.

[4] D. A. Jiménez, "Oh-snap: Optimized hybrid scaled neural analog predictor," *IN PROCEEDINGS OF THE 3RD CHAMPIONSHIP ON BRANCH PREDICTION, HTTP://WWW.JILP.ORG/JWAC-2*, 2011.

[5] J. Singer, G. Brown, and I. Watson, "Branch Prediction with Bayesian Networks," in *First Workshop on Statistical and Machine learning approaches applied to ARchitectures and compilaTion*, 2007, pp. 96–112.

[6] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, Dec 1943.

[7] D. O. Hebb, *The Organization of Behavior*. Wiley, 1949.

[8] F. Rosenblatt and F. Rosenblatt, "The Perceptron: A Probabilistic Model for Information Storage and Organization in The Brain," *PSYCHOLOGICAL REVIEW*, pp. 65–386, 1958.

[9] P. Y. Simard, D. Steinkraus, and J. C. Platt, "Best practices for convolutional neural networks applied to visual document analysis," in *Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings.*, vol. 1. IEEE Comput. Soc, 2003, pp. 958–963.

[10] D. C. Cireşan, U. Meier, J. Masci, L. M. Gambardella, and J. Schmidhuber, "Flexible, high performance convolutional neural networks for image classification," *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence - Volume Volume Two*, pp. 1237–1242, 2011.

[11] G. Hinton *et al.*, "Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, nov 2012.

[12] D. A. Jiménez and C. Lin, "Dynamic branch prediction with perceptrons," in *Proceedings HPCA Seventh International Symposium on High-Performance Computer Architecture*. IEEE Comput. Soc, 2001, pp. 197–206.

[13] M. N. Marsono, M. W. El-Kharashi, and F. Gebali, "Binary LNS-based naïve Bayes inference engine for spam control: noise analysis and FPGA implementation," *IET Computers & Digital Techniques*, vol. 2, no. 1, p. 56, 2008.

[14] Deng Zhijie, Wang Yong, and Tao Xiaoling, "Method of network traffic classification using Naïve Bayes based on FPGA," in *IEEE Conference Anthology*. IEEE, jan 2013, pp. 1–3.

[15] Lattice, "LatticeMico32 Open, Free 32-Bit Soft Processor," 2018.

[16] M. R. Guthaus, J. S. Ringenberg, D. . Ernst, T. M. Austin, T. Mudge, and R. B. Brown, "MiBench: A free, commercially representative embedded benchmark suite," in *Proceedings of the Fourth Annual IEEE International Workshop on Workload Characterization. WWC-4 (Cat. No.01EX538)*. IEEE, 2001, pp. 3–14.

[17] M. G. Katevenis, “3.1 On-Chip SRAM,” 2003.

# 第4章　3Dメモリを深層学習に活用するための局所畳み込み層間学習法

## 4.1　緒言

ディープラーニングを含むニューラルネットワークの技術は急速に開発されており、人間の専門家でさえうまく処理できない問題も解決できることが期待されている。これらのテクノロジは、現在、豊富なリソースとストレージを備えた強力なコンピューティングアーキテクチャでのみ実行可能であるため、サーバーやデータセンターで主に使用されている。しかし、これらの技術は近い将来、モノのインターネット (IoT)、すなわちエッジシステムにも展開される必要がある。IoT エッジシステムは現在センシングデバイスなどとして使用されているが、ニューラルネットワークのテクノロジはエッジに知性をもたらし [1]、ヘルスケアや防災のような様々な新しいアプリケーションをリアルタイムで処理することを可能にする。

ほとんどの IoT エッジシステムはバッテリに制約があるため、アーキテクチャ上の工夫とデバイスの工夫の両方を組み合わせ、エネルギー効率の高いエッジ AI システムを実現することが重要である。デバイスに関する多くの研究の中で、めもりすたデバイスは現在のシリコンデバイスを代替するために集中的に研究されてきた [2–4]。メモリスタデバイスは、異なる抵抗によって異なる値を表すが、それらの抵抗値を利用することによって、算術演算装置とメモリの両方として機能する。さらに、3D メモリスタデバイスは、2D 対応デバイスよりも回路面積においてさらに効率的である。したがって、この 3D 構造は新しいアーキテクチャを検討するときに活用する必要がある。例えば、人間の脳を含むほとんどの物質的な物体が本来 3D 構造を有するという事実に焦点を当てると、ニューラルネットワークはそのような 3D メモリ素子の内部構造を利用することによって効率的に実現することができる。

本稿では、上記の背景に動機付けられて、小型で電力効率の良いエッジ AI システムを実現するために、3D メモリスタデバイスを意識したニューラルネットワークモデルを提案する。このモデルは、局所結合畳み込み Deep Belief Network(LCCDBN) と呼ば

れ、畳み込みニューラルネットワーク (CNN) と Deep Belief Network(DBN) に基づいて開発した。3 次元メモリスタデバイスは製造および実施においていくつかの厳しい制約があり、十分な性能を達成しながらこれらのデバイスにおける様々な展開可能な構造を探求する。提案する LCCDBN の重要な特徴と貢献は以下の通りである。

- 3D メモリスタデバイスでニューラルネットワークを実現するとき、畳み込みの層間接続は電極間の導線によって実現される。装置の欠陥を回避するためには、これらのワイヤの複雑さを軽減する必要がある。より具体的には、重なり合うワイヤは制限されている。我々のモデルでは、この問題は畳み込みを隣接するデータ (すなわち局所データ) のみに限定することによって解決される。

- 実装の観点からは、レイヤ単位の教師なし学習は、wire by wire の教師つきトレーニングよりも優先される。fine-tuning(または教師あり学習) はオフチップメモリを利用して重みを管理するために各層のコントローラのような追加の装置技術を必要とするが、層ごとに重みレイヤを事前設定する教師なし学習はそのような技術を必要としない。我々は、この制約の下で様々な構造を考慮するだけでなく、この制約を緩和することによる影響も評価する。

- 提案するモデルは、ノイズなどに対する 3D メモリデバイスの脆弱性を考慮するために、DBN から生じる確率的性質を利用する。

LCCDBN は、アプリケーションに応じたカスタマイズ可能性を提供するために、その出力が再構成可能なハードウェアまたはソフトウェアによって開発される分類器に供給される特徴抽出器として 3D メモリデバイスに実装されると仮定する。したがって、既存の転移学習手法は、エッジ AI システム全体を効率的に構築するために、特徴抽出に利用するのが妥当である。

本研究では、様々にパラメータを変えて、実行可能で適切な LCCDBN 構造を定量的に調べた。層ごとの教師なし学習を使用するという厳しい制約により、層間接続 (すなわち導線) の大幅な減少にもかかわらず、我々の LCCDBN が従来の DBN と同等の性能を提供することを実証した。また、実装コストを犠牲にして、この制約を緩和するとパフォーマンスが大幅に向上することを明らかにした。デバイス実装に困難が生じるが、この緩和は取り組む価値があると考えられる。

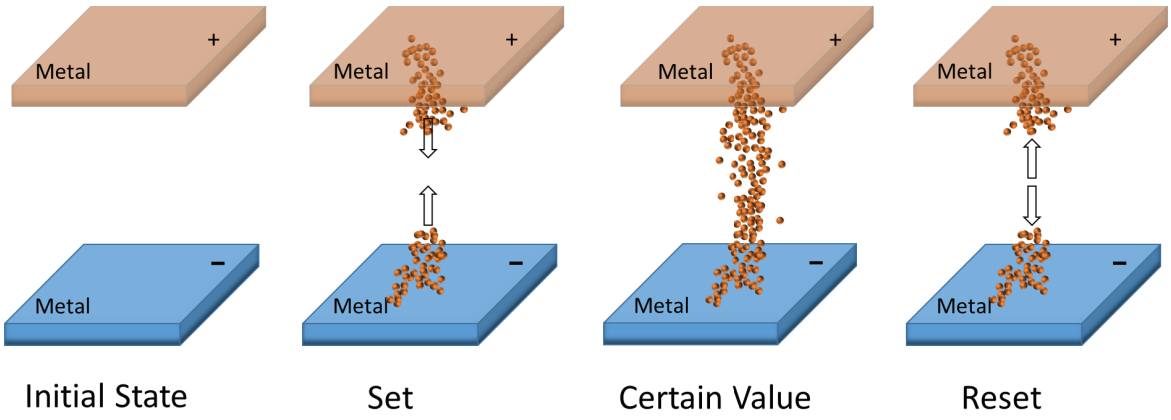



**図 4.1**　コンダクタンスの制御構造模式図

## 4.2　3 次元積層メモリスタ

CMOS デバイスに代わるものとして、相変化メモリ (PCM) [5]、磁気ランダムアクセスメモリ (MRAM) [6]、および抵抗ランダムアクセスとメモリ (ReRAM) [7] など、様々な新しいタイプのメモリスタデバイスが学術的および産業的の両方で研究されてきた。これらのメモリは、従来のフラッシュメモリと同様に不揮発性で低電力でありながら、従来のＤＲＡＭと同じくらい高速にデータにアクセスすることができる。さらに興味深いことに、それらは高速なランダムアクセスメモリとしてだけでなく、それらのアナログの振る舞いを利用することによってコンピューティングユニットとしても働くことができる。

これらの記憶装置では、導線を利用してデータが保存される。(簡潔のため、本稿では電極間の支配的な導電パス (例 : ReRAM のフィラメント) を表すために「導線」という用語を使用する。) 図 4.1 は、２つの電極間の PCM などのいくつかの記憶装置におけるコンダクタンス制御のメカニズムを表す。導線は十分に高い電圧下で制御される。コンダクタンスは 2 つの電極間の算術演算 (特に乗算演算の重み) により電圧制御で、異なるデータまたは値を表す。by $v = R(w)i$, where $v$ is voltage, $i$ is current, and $R()$ is a generalized resistance depending on the state variable $w$, which has a relationship between time $t$ and electric charge $q$ (i.e., $dw/dt \propto dq/dt$) [8]. ここで、電圧 $v$、電流 $i$、時間 $t$ と電荷量 $q$ に対して $dw/dt \propto dq/dt$ の関係が成り立つ $w$ に依存する可変抵抗 $R()$ に対して、コンダクタンスは $v = R(w)i$ で決定される [8]。より具体的には、抵抗（すなわち、コンダクタンスの逆数）は、短期間での電流（または電圧）の変化量によって変化する [9–11]。これらの特徴は、記憶装置の基礎を表している。これらの導線がど

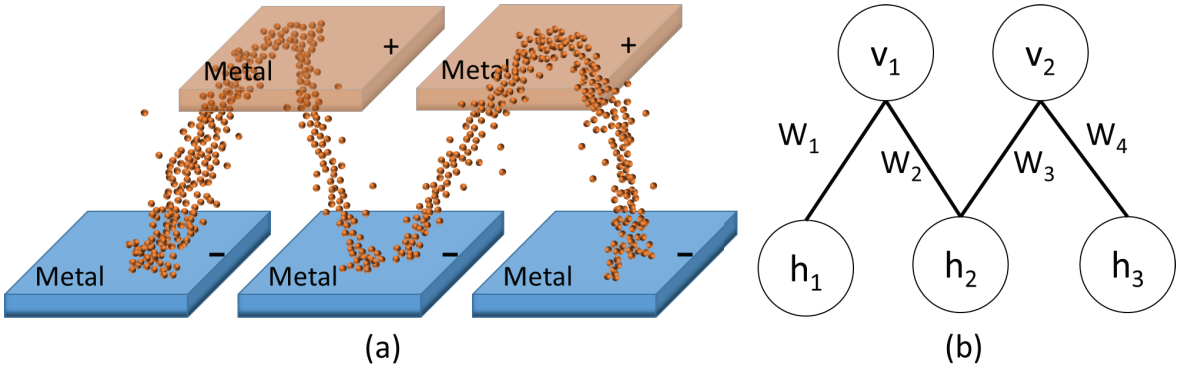



**図 4.2** デバイスを踏まえたニューラルネットワーク (a)3D 構造の例 (b) 対応するニューラルネットワークのモデル

のように制御されるかという点で、記憶装置は大きく段階的制御とワンタイム制御の 2 つに分類することができる。PCM [12] のような前者の装置では、学習プロセスにおける重みの反復的な増加および減少の間に、電圧パルスによって段階的に重みが更新される。一方、ReRAM のような後者では、重みは予め決められており、一度に電圧パルスによって導線に加えられる。どちらのタイプの記憶装置でも、初期状態での重みは 0 と見なされますが、特定の値にプログラムされた後の重みは 0 から 1 の範囲の実数を表す。なお、本研究では特定のメモリスタデバイスを想定していません。

現在のデバイスでは、導電性ワイヤは通常、図 4.2 に示すように 2 つの電極間に一対一の関係で実現することができる [13]。したがって、これらのデバイスは、ニューラルネットワークの畳み込みを実行するために多数のクロスバー接合部を有するグリッド構造で実装される必要がある。実際、ほとんどのクロスバー構造は、CNN では使用されない。これは、図 4.2(a) に示すように、導電性ワイヤを複数の電極間を橋渡しすることで解決することができ、このような手法は 3D メモリスタデバイスで広く研究されている [14]。この技術を利用し、導線を適切に制御することで (図 4.2(a))、記憶装置全体は効率的に畳み込みを実行することができる (図 4.2(b))。各ワイヤのコンダクタンスは重み ($w_i$) を表し、信号が上部電極から下部電極に渡されるときに、$h_2 = w_2 \times v_1 + w_3 \times v_2$. のように、重みと入力信号の積 ($v_j$) が計算される。

層間の配線削減は、集積化の観点からだけでなく、デバイス製造問題の観点からも不可欠である。前者は、面積の縮小による低消費電力消費という点で明白である。一方、3 次元メモリスタデバイスでニューラルネットワークを実現する場合、後者は、層間の配線が複雑すぎたり多すぎたりすると、電極間でワイヤが重なり合う（または絡

み合う）ことから生じる。重なっているワイヤは装置に重大な欠陥を引き起こすので、それらは、畳み込みにおける隣接 (またはローカル) データのアクセスのみに限定することなどによって回避されなければならない。

## 4.3　転移学習

転移学習は、学習されたネットワークの入力特徴抽出機能を関連する未学習問題に再利用する方法である [15]。このネットワークは特徴抽出器として機能し、その出力は問題 (またはターゲットアプリケーション) に応じてカスタマイズできる分類器に送られる。たとえば、図 4.3 では、最初にデータセット A を使って特徴抽出器を構築します。次に、特徴抽出器を使って「データセット A」に関連する「データセット B」を分類する。このとき、ニューラルネットワーク B では、それに応じて分類器が再構築される。

エッジ AI システムのカスタマイズ性は実用上必須であるため、転移学習を採用するのが現実的である。本研究では、図 4.3 に示すように、特徴抽出器がハードワイヤード 3D 記憶装置に実装され、ユーザー定義の分類器がソフトウェアまたは再構成可能なハードウェアによってカスタマイズ可能であると仮定します。特徴抽出器内のワイヤの重み（コンダクタンス）は、学習によって事前設定、すなわち配線することができる。

AlexNet [16] や VGG16 [17] のように、転移学習のために実際に使用されている特徴抽出器は数多く存在するが、それらは複雑な構造を有するため 3D メモリスタデバイスでは実現できない。より具体的には、層間の大量の接続 (例えば、ReRAM 内のワイヤ) の重なりは、デバイスに実装された場合、短絡または重大な欠陥を引き起こす。次章では、3D メモリスタデバイスに実装するのに十分に単純で、前述の配線の問題を解決するための、一種の確率的ニューラルネットワーク構造を紹介する。

## 4.4　Locally-Connected Convolutional Deep Belief Network

本研究の目的は、組み込みシステム、いわゆるエッジ AI システムのための低電力 3D メモリスタでニューラルネットワークを実現することである。これらの装置における厳しい制約を考慮して、畳み込みニューラルネットワーク (CNN) と Deep Belief Network(DBN)

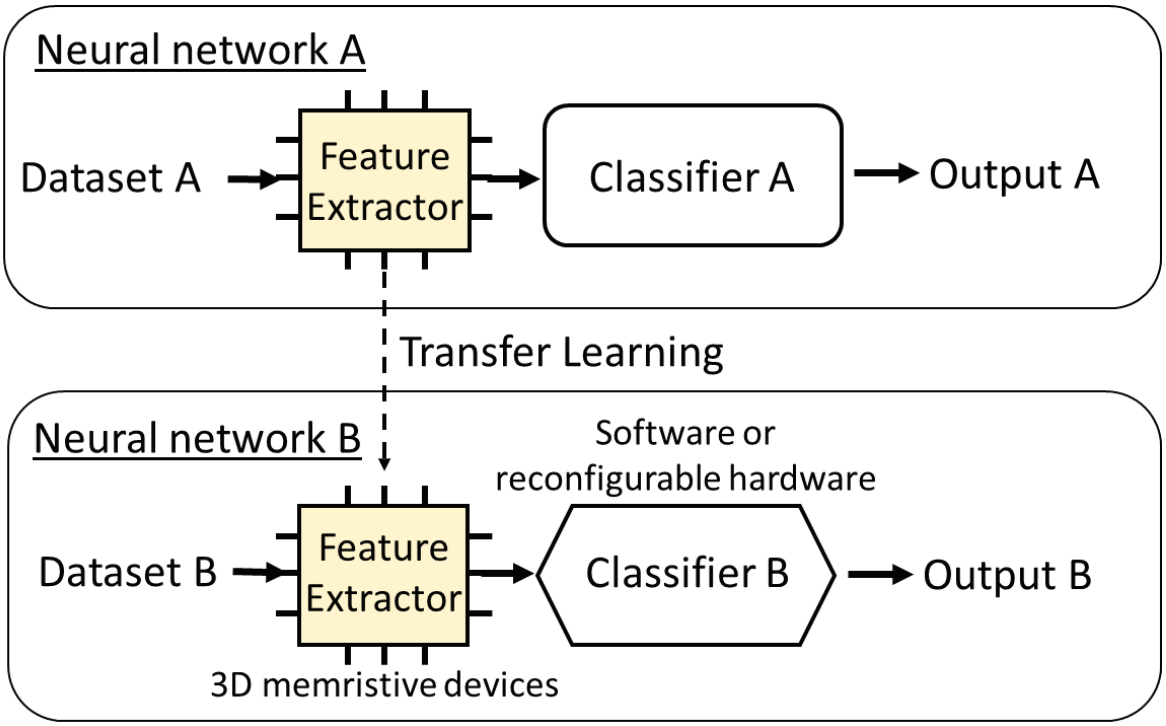



**図 4.3** 転移学習

の利点を統合することにより局所結合畳み込み Deep Belief Network(LCCDBN) と呼ばれる確率ニューラルネットワークを提案する。

3 次元メモリスタデバイスは製造においていくつかの厳しい制約を有する。その中でも、ワイヤの複雑さとコンダクタンス制御は特に厳しいものである。まず、CNN(AI のさまざまなアプリケーションで広く使用されているニューラルネットワーク) に注目して、レイヤー間の重み設定を学習する。CNN では、各レイヤは、特徴マップを作成するために、入力に適用される単一または複数の小さいサイズの重み付き行列を使用する (共有重み (図 4.4(a)))。しかし、共有重みは、全結合を含むネットワークでは、図 4.4(a) に示すように導線が重なり合ってしまうため、3D メモリスタデバイスでは実現できない。そのため、局所結合の非共有ウェイトを用いることで製造上の制限を考慮しながら、複数の特徴マップを作成できるように CNN を拡張する (図 4.5(b))。たとえば、28 × 28 ピクセルの入力画像の場合、共有重みは 1 つの畳み込みレイヤに 1 つの特徴マップを作成するが、3 × 3 重み行列を 1 つのゼロパディングで使用すると、拡張機能は 28 の特徴マップを作成する。非共有重みを使用することにより、製造の容易さと複数の特徴マップの両方が実現される。

次に、我々はその訓練方法による別の確率的ニューラルネットワーク、DBN に注目

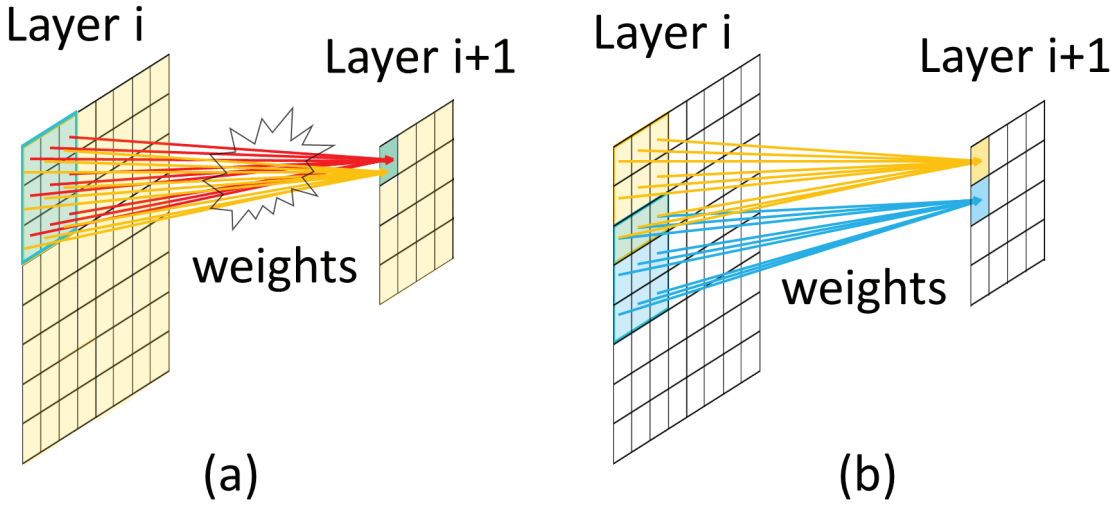



**図 4.4** 配線の複雑性 (a) 導線の重なり合い (b) 重なり合いのない配線.

する。学習方法は教師ありと教師なしの 2 つのアプローチに分類される。3D メモリスタデバイスの場合、教師ありトレーニングを採用することで、重みを wire by wire に調整する。これは、従来のフォンノイマンアーキテクチャで行われていたのと同様に、重みが算術演算ユニットから離れて (例えばオフチップメモリに) 格納され、オフチップコントローラの制御を要する。これは、分類性能を向上させる一方で、製造上の困難および計算速度の低下ももたらし得る。反対に、教師なし学習を採用すると、重みが layer by layer に調整される。つまり、重みはオフチップコントローラの介入なしに、局所結合によって調整される。これは、今日のアーキテクチャにおけるインメモリ・コンピューティング [13] と同等であり、コンピューティングおよびエネルギーの観点から、明らかに製造が容易で効率的である。そのため、3D メモリスタデバイスのトポロジを考慮すると、階層別のトレーニングがより適している。実際には、貪欲な階層別学習が十分なパフォーマンスを得て、DBN として機能することが示されている [15, 18, 19]。

DBN に注目するもう 1 つの理由は、その確率論的な性質である。最近の 2D メモリスタデバイスの研究動向を参照すると [13]、本研究では目標とする 3 D メモリスタデバイスはアナログ回路としても使用される可能性があり、それは環境雑音、素子ばらつきなどの動的擾乱に対して脆弱である。これらの効果を考慮するために、3D メモリスタデバイスによるニューラルネットワークは、決定論的ではなく確率論的にモデル化する必要がある。本研究では部分的に DBN を拡張することによって、確率変数の層で構成される DBN の確率的性質に暗黙のうちに対処する。

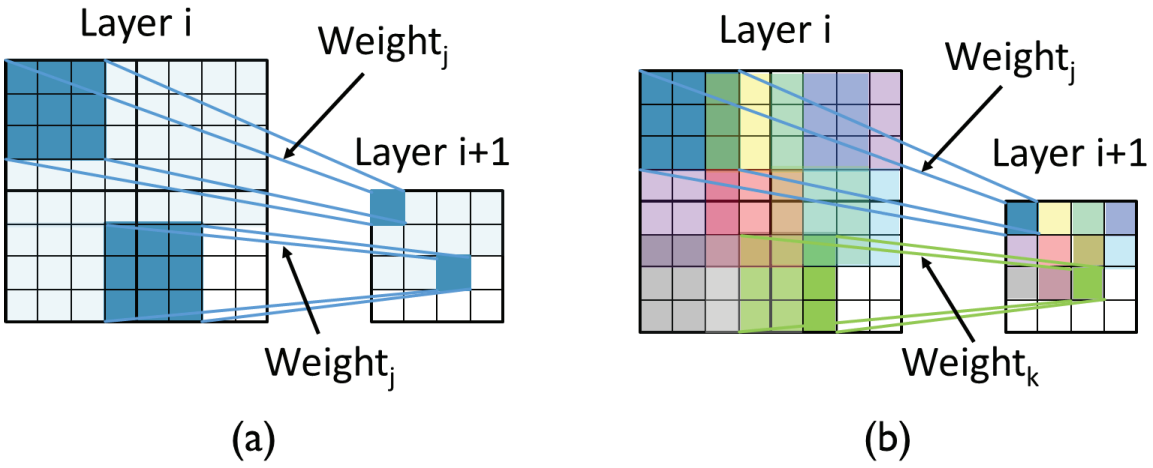



**図 4.5** (a) 重み共有畳み込み (b) 重み非共有畳み込み

要約すると、LCCDBN は局所結合のみを持つ確率ニューラルネットワークモデルで、CNN(非共有重み) と DBN(階層別学習と確率的性質) から開発された。この LCCDBN は、特徴抽出器を構築するために 3D メモリスタデバイスに組み込まれ、出力は全結合分類器 (再構成可能なハードウェアまたはソフトウェアとして実現される) に接続される。図 4.6 は、3 層特徴抽出器と 1 層分類器の例を説明している。本論文では、特徴抽出器は製造技術を考慮して小さな 3x3 フィルタを有する比較的少ない層 (すなわち 1 から 3 層) からなると仮定する。しかしながら、これらの制限は、デバイス製造技術の進歩と共に将来緩和される可能性がある。また、このような構成の簡素化は、ネットワーク構造を調査したり変更したりすることによる影響を理解する上で重要である。

## 4.5 評価

### 4.5.1 評価方法

本研究では MNIST と CIFAR-10 データセットを用いて評価を行なった。

- MNIST [20] (Fig. 4.7) は、0 から 9 の 10 種類の手書き数字で構成されるデータセットである。50000 枚の学習用データ、10000 枚の検証用データ、10000 枚のテスト用データが含まれる。各画像は 28x28 ピクセルのグレースケール画像である。MNIST は、その単純さから、画像分類タスクにおける最も初歩的なタスクとして広く用いられている。

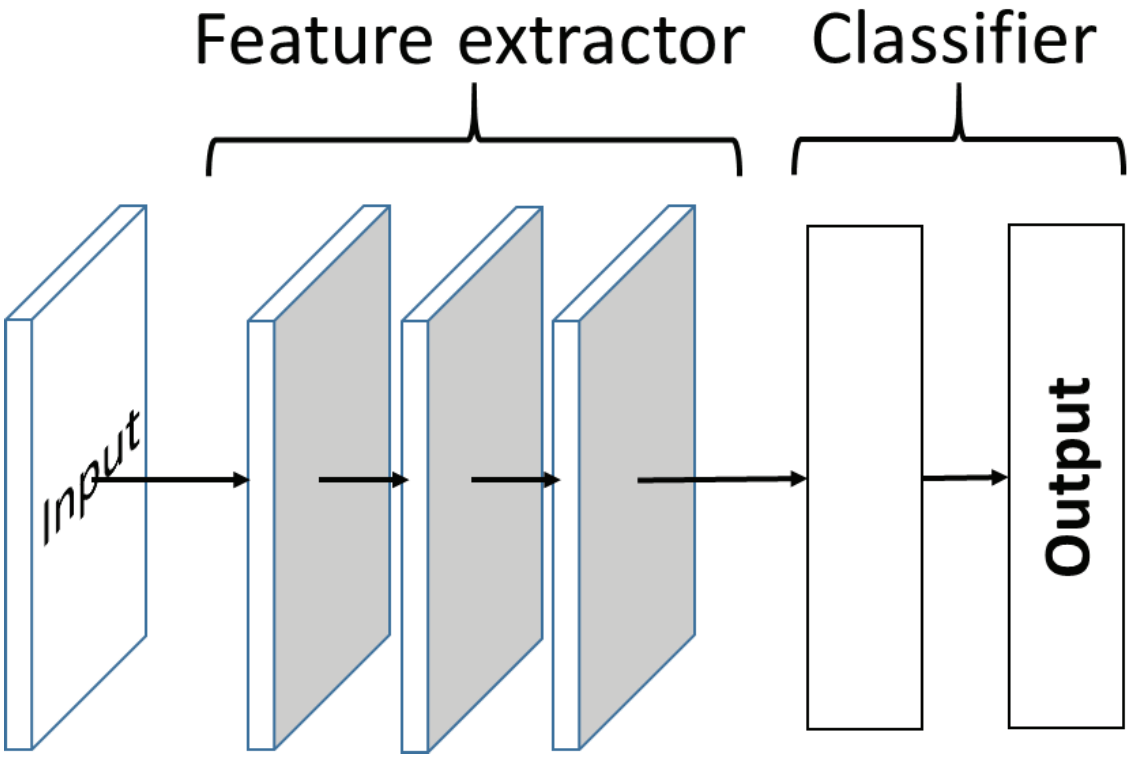



図 **4.6**　LCCDBN で構成されるネットワークの全体図

- CIFAR-10 [21] (Fig. 4.8) は、10 クラス (airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck) の画像から構成されるデータセットである。50000 枚の学習用データと 10000 枚のテストデータが含まれる。検証用データは含まれていないが、学習用データの一部を切り出すことで、容易に作成できる。各画像は 32x32 ピクセルのカラー画像であるが、本評価では簡単のためにグレースケールに変換して用いる。このデータセットも、MNIST よりも一段階現実的にデータセットとして、広く利用されている。

本評価では、3 種の異なる構造の LCCDBN を含む、4 種類のニューラルネットワークに対して評価を行なった。LCCDBN のコンセプトは、部分的に DBN [18] からの派生であるため、活性化関数としてシグモイド関数を、重みは正規分布で初期化した。各ネットワークの概要を以下、および表 4.1 に示す。

1. **DBN**: DBN は 1 から 3 個の全結合隠れ層から構成される。全結合層自体が特徴抽出器としても分類器としても機能するため、別途分類器を附置する必要がない。Hinton らのモデル [18] に基づき、ニューロンの数は第 1 隠れ層は 500 個、第 2 隠

れ層は 500 個、第 3 隠れ層は 2000 個と設定した。

2. **LCCDBN_2,000**: この LCCDBN は、教師なし階層別学習のみが可能であるような厳しいデバイス制約を仮定して構築した。エッジ AI システムにおいては、分類器は簡素な構造の方が好ましいため、分類器は 1 層構造のみとした。2000 は分類器が 2000 個のニューロンを含むことを表し、これはリソース・リッチな分類器であるといえる。

3. **LCCDBN_100**: 特徴抽出器は LCCDBN_2,000 と同一であるが、分類器は 100 個のニューロンのみを含み、すなわちメモリ容量が少ないようなリソースがより厳しく制限されたシステムを仮定した。これにより、LCCDBN そのものの性能をより明らかにすることができる。

4. **LCCDBN_supervised**: この LCCDBN はバックプロパゲーション (すなわち教師あり学習) を用いる。これは製造上の困難を克服し、制約が緩和された場合を仮定する。これらの制約緩和により、活性化関数として ReLU [22] を、重みの初期化手法として He らの手法 [23] を用いることができ、バックプロパゲーションをより効果的に活用することができる。

また、本研究では特徴抽出器としての性能を評価するために、各ネットワークに対し、次のようにして転移学習実験を行なった。

- **Only classifier**: 特徴抽出器を用いずに、分類器のみをバックプロパゲーションで学習させた。これは、特徴抽出機能評価のベース指標となる。

- **With feature extractor (FE)**: 4 種類の各ネットワークを特徴抽出器として用いる。最初に、各ネットワークを 10 クラスのデータセットを用いて学習させる。次に、学習された特徴抽出器を再利用し、分類器のみを 5 クラスのデータセットを用いて学習させる。ここで、5 クラスとは MNIST の場合は偶数のみを、CIFAR-10 の場合は airplane, bird, deer, frog, ship を選んだ。

本評価では、1 から 3 の隠れ層の数に対し、前述の 4 つの構造を比較するために、次にあげる 3 つの測定基準を利用した。

- **Wire complexity**: 全結合層と局所結合層を含む DBN [18] および LCCDBN の配線の複雑性を見積もった。

**表 4.1** 評価対象としたネットワークの概要

| | **DBN** | **LCCDBN _2,000** | **LCCDBN _100** | **LCCDBN _supervised** |
|---|---|---|---|---|
| Hidden layers # | 1, 2, 3 | | | |
| Hidden neurons # | $1^{st}$:500, $2^{nd}$:500, $3^{rd}$:2,000 | Same as input pixels | | |
| Classifier layers # | - | 1 Layer | | |
| Classifier neurons # | - | 2,000 | 100 | 2,000 |
| Receptive field size | - | $3 \times 3$ | | |
| Zero-padding size | - | 1 | | |
| Stride | - | 1 | | |
| Learning method | Greedy layer-wise training | | | Back prop. |
| Activation function | Sigmoid | | | ReLU |
| Weight init. | Normal distribution | | | He init. |

**図 4.7** MNIST データセット

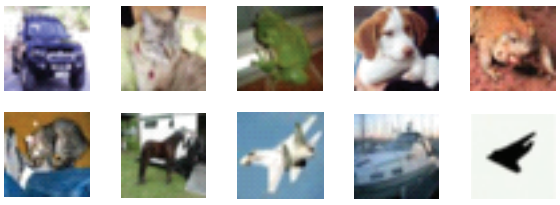


**図 4.8** CIFAR-10 データセット

表 4.2 Wire complexity.

**MNIST**

| | **DBN** | **LCCDBN** |
|---|---|---|
| 1 Hidden Layer | 28×28×500 = 392,000 | 1×(3×3×28×28) = 7,056 |
| 2 Hidden Layers | 392,000 + (500×500) = 642,000 | 2×(3×3×28×28) = 14,112 |
| 3 Hidden Layers | 642,000+ (500×2,000) = 1,642,000 | 3×(3×3×28×28) = 21,168 |

**CIFAR-10**

| | **DBN** | **LCCDBN** |
|---|---|---|
| 1 Hidden Layer | 32×32×500 = 512,000 | 1×(3×3×32×32) = 9,216 |
| 2 Hidden Layers | 512,000 + (500×500) = 762,000 | 2×(3×3×32×32) = 18,432 |
| 3 Hidden Layers | 762,000+ (500×2,000) = 1,762,000 | 3×(3×3×32×32) = 27,648 |

- **Image reconstructability**: 隠れ層の数を変えて、DBN および LCCDBN がどれだけ入力画像を復元するかを視覚的に評価した。
- **Performance of transfer learning**: 各ネットワークに対し、転移学習の性能を MNIST および CIFAR-10 を用いて定量的に評価した。

### 4.5.2 評価結果

まず、DBN と LCCDBN のワイヤ数を比較して、ワイヤの複雑さが軽減された効果を評価した。配線の複雑さは 3 つの異なる LCCDBN 構造で同じである。表 4.2 の上部と下部は、隠れ層の数を変えた時の、MNIST と CIFAR-10 のワイヤ数を表している。DBN では、第 1 の隠れ層内のワイヤ数は、入力画像サイズとニューロン数 500 の積によって決まる。層が増えるにつれて、前の層と次の層のニューロン数の積によって決まるワイヤが追加されます。一方、LCCDBN では、各層は、受容野サイズ 3x3 と入力画像サイズとの積によって決定される同じワイヤ数を有する。これらの結果から、LCCDBN による配線の削減がかなり大きいことが明らかになった。これは 3D メモリスタにおけるニューラルネットワークの実現を加速するであろう。

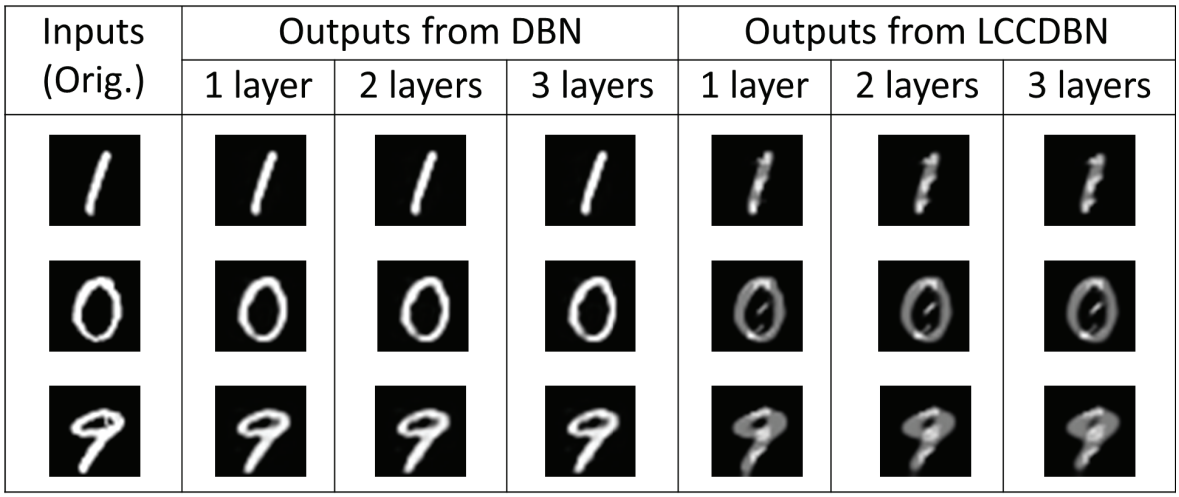



**図 4.9**　MNIST データセットに対する画像再構成

**表 4.3**　MNIST データセットに対する性能比較

| | **DBN** | **LCCDBN _2,000** | **LCCDBN _100** | **LCCDBN _supervised** |
|---|---|---|---|---|
| Only Classifier | 96.12% | 97.62% | 96.18% | 98.90% |
| With 1-layer FE | 93.62% | 95.47% | 90.17% | 99.00% |
| With 2-layer FE | 95.96% | 94.64% | 91.81% | 98.94% |
| With 3-layer FE | 93.58% | 95.12% | 90.13% | 99.02% |

**MNIST**

次に、LCCDBN が特徴抽出器として十分に機能することを確認するために、画像再構成性と定量的性能の観点から、MNIST について評価を行った。MNIST の中のいくつかの選択された画像について、図 4.9 は原画像と再構成画像を特徴抽出器のみに対して比較する。DBN と LCCDBN は両方とも階層別教師なし学習によって学習された。LCCDBN の出力にノイズが追加されているが、ワイヤの複雑さが大幅に減少しているにもかかわらず、画像は十分に再構成されていることがわかる (表 4.2)。

表 4.3 は、画像分類タスクに対する特徴抽出器の性能を比較している。ここでもやはり、LCCDBN は厳しいデバイス制約の下でも DBN と同等の性能を達成していることがわかる (LCCDBN 2,000 と DBN を最後の層で同じニューロン数を持つため)。しかしながら、DBN および教師なし LCCDBN は、分類器のみよりも性能が劣化した。これは、MNIST での画像分類タスクが十分に単純なためである。

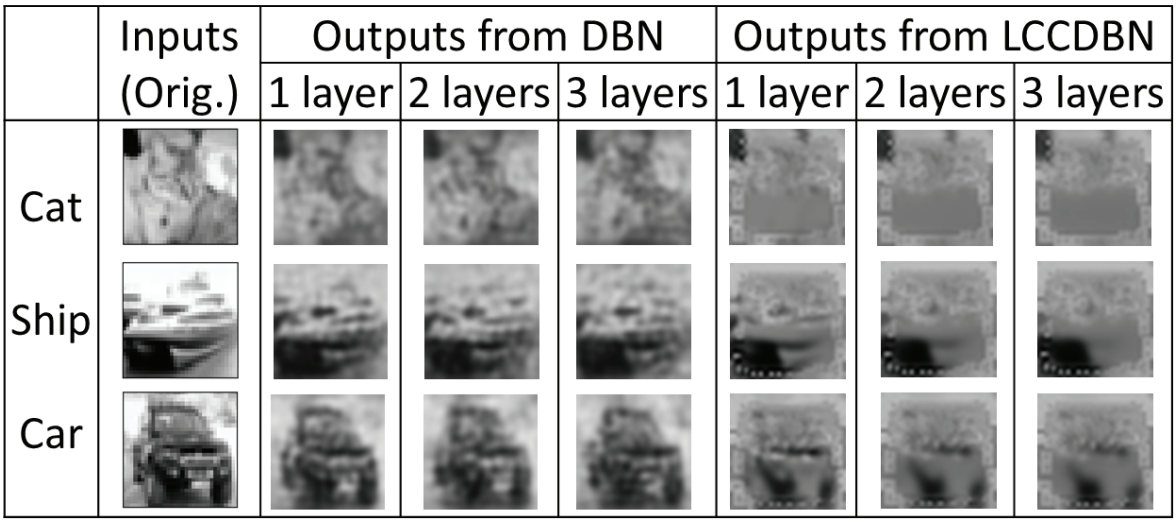



**図 4.10** CIFAR-10 データセットに対する画像再構成

**表 4.4** CIFAR-10 データセットに対する性能比較

| | **DBN** | **LCCDBN _2,000** | **LCCDBN _100** | **LCCDBN _supervised** |
|---|---|---|---|---|
| Only Classifier | 41.08% | 49.78% | 41.14% | 57.68% |
| With 1-layer FE | 40.96% | 47.74% | 41.56% | 65.06% |
| With 2-layer FE | 43.82% | 47.70% | 41.78% | 64.56% |
| With 3-layer FE | 40.94% | 47.54% | 41.92% | 62.96% |

**CIFAR-10**

MNIST データセットと同様に、CIFAR-10 データセットについても評価を行なった。図 4.10 のように画像は MNIST の画像よりも複雑になるため、DBN と LCCDBN の両方の再構成画像はわずかにぼやけてしまう。これは、特に大幅な配線削減が行われている LCCDBN において、教師なし学習の限界を示唆する。

表 4.4 は分類器のみの場合と特徴抽出器を用いた場合の性能結果を示している。LCCDBN からの画像再構成 (図 4.10) は DBN からの画像再構成より劣っていると考えられるかもしれないが、DBN と LCCDBN2,000 を比較すると LCCDBN がより良い性能が達成されている。これは、LCCDBN が複雑な画像上でも分類に必要な特徴を効率的に抽出できることを示している。LCCDBN ベースの特徴抽出器を使用すると、リソースが豊富な 2,000 個のニューロンを有する分類器のみ使用する場合よりも性能が低下す

るが、これらの結果は、分類器が 100 個のニューロンのみを有する場合は逆転する。また、MNIST データセットに対する結果に立ち返ると、分類タスクが複雑である現実的な状況では、LCCDBN ベースの特徴抽出は分類に役立づと考えられる。最後に、我々は、デバイス製造技術が成熟するにつれてオフチップコントローラの実装が実現可能になるかもしれないと仮定し、教師あり LCCDBN を評価した。表からわかるように、LCCDBN ベースの特徴抽出器は、この制約を緩和することによってパフォーマンスを大幅に向上させます。

これらの評価を通して、以下の発見を得た。

- ワイヤの複雑さが大幅に減少したにもかかわらず、LCCDBN は DBN と同等の性能と画像再構成を示した。LCCDBN の特徴抽出機能をさらに実証するために、分類器を除く特徴抽出器のみの性能を t-分散確率的近隣埋込み (t-SNE) [24] によって示す (図 4.11 および表 4.11 )。MNIST データセットに対する 1 つの隠れ層特徴抽出器の結果のみが示されている。これは、他の構造についても同様の結果が観察されたからである。図 4.11 は、特徴がどの程度うまく抽出され区別されているかを説明している。各クラスが密であるほど、特徴はより良く抽出される。特徴抽出器を用いない場合、クラスは部分的に重なるが (図 4.11(a))、特徴抽出器を用いるとより分離され (図 4.11 (b)-(d))、DBN と LCCDBN の有用性がわかる。特に図 4.11(b) と図 4.11(d) の結果に注目すると、層間配線が DBN よりはるかに少ないにもかかわらず、LCCDBN によって特徴がよく抽出されていることがわかる。さらに、特徴抽出の定量的測定基準として、t-SNE 座標におけるクラスターの中心から各クラスのすべてのベクトルの合計を使用して計算される角度を導入した (表 4.5)。表 4.5 からわかるように、各角度はほぼ均等に離れており、つまり、すべてのクラスは明確に区別されているといえる。これらの結果は、上記の評価に加えて、特徴抽出器としての LCCDBN の有効性を十分に示している。

- 一般的に、メモリスタデバイスは大きな変動性を持っている。2 つのタイプのメモリスタデバイスのうち、段階的なコンダクタンス制御タイプは、コンダクタンス制御プロセス中の変動性を考慮に入れるが、一回限りのコンダクタンス制御は、重みの読み取りおよび書き込みの両方における変動性によって影響を受ける。したがって、前者は後者よりも変動性は少ないがコンダクタンス制御においてより時間がかかる。コンダクタンス制御時間を考慮すると、後者のタイプのメモリスタデバイスに関する変動性を意識した学習が好ましい。したがって、事前定義された重みに依存しないという点で変動性があることを認識しているため、教師な

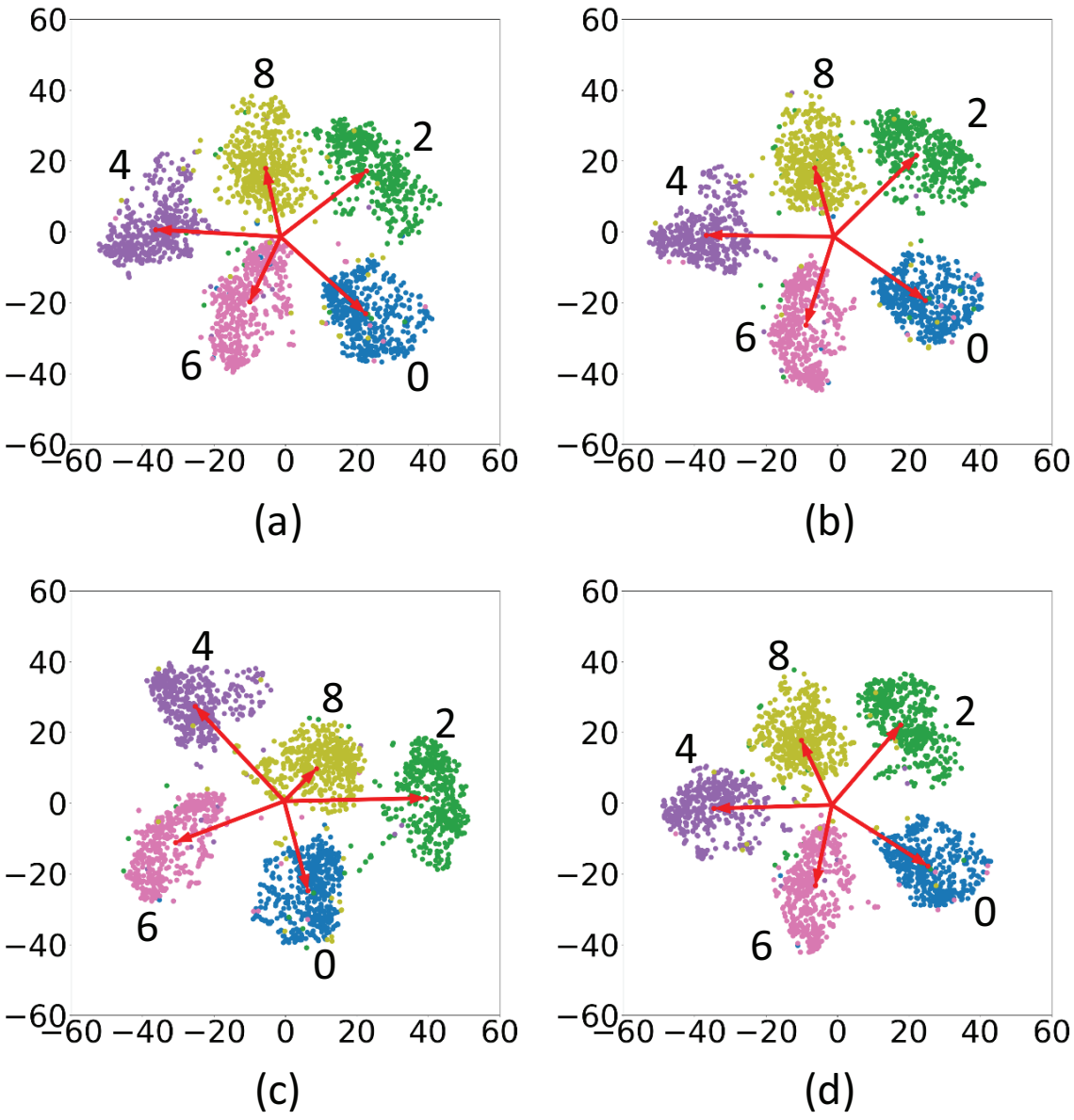



**図 4.11** t-SNE による MNIST の 2500 データ (500 データ x5 クラス) に対する特徴抽出の可視化 (a) 特徴抽出器を用いない場合 (b)DBN を用いた場合 (c)LCCDBN を用いた場合 (d) 教師あり学習された LCCDBN を用いた場合

**表 4.5** t-SNE による隣接クラスの角度

| Angle between two classes | **Original** | **DBN** | **LCCDBN_2,000 LCCDBN_100** | **LCCDBN _supervised** |
|---|---|---|---|---|
| Between Class 0 and Class 2 | 80.0 | 80.6 | 78.7 | 81.3 |
| Between Class 2 and Class 8 | 64.6 | 59.9 | 36.4 | 66.2 |
| Between Class 8 and Class 4 | 74.4 | 73.1 | 92.3 | 65.7 |
| Between Class 4 and Class 6 | 68.1 | 75.2 | 67.8 | 75.7 |
| Between Class 6 and Class 0 | 72.8 | 71.2 | 84.8 | 71.1 |

し学習が適している。上記の評価を通して、限られた層間配線に教師なし学習を使用するという厳しい制約の下でも、LCCDBN は入力画像の特徴を抽出することがわかった。分類タスクが複雑になればなるほど、LCCDBN ベースの特徴抽出器はその有用性を発揮する。

- 教師あり学習を用いると、パフォーマンスが大幅に向上し、特徴抽出器の有用性がより強調される。各隠れ層にコントローラを配置して重みを fine-tuning する、段階的なコンダクタンス制御タイプのメモリスタデバイスを利用する、または意図的なリフレッシュ技術を実現する [25] 、一回限りのコンダクタンス制御のばらつきを抑えるなど、いくつかの異なるアプローチで実現できる。これらは明らかに実装上いくつかの困難を強いるが、それらは取り組む価値があると考えられる。

## 4.6 結言

本性では、3D メモリスタデバイスを用いた低消費電力なエッジ AI システムを実現するために、局所結合畳込み Deep Belief Network(LCCDBN) と呼ばれる確率ニューラルネットワークのモデルを提示した。メモリスタデバイスの厳しい製造および実装上の制約を考慮して、LCCDBN は、それが転送学習のための特徴抽出器として実装できるように、ネットワーク内の接続 (例えば ReRAM 内のワイヤ) を大きく制限する。実験を通して、我々はいくつかの構成パラメータを変えて LCCDBN 構造を調査し、LCCDBN が大幅なワイヤ減少にもかかわらず従来の DBN に匹敵することを実証した。LCCDBN は複雑な分類作業のための特徴抽出器として有用であるが、さらなる改善が必要である。さらに、教師あり学習を採用するためにデバイスの制約を緩和することによって、

パフォーマンスが急速に向上する可能性が示唆された。このデバイスの制約を保持または緩和することによる定量的な調査を通じて、この制約を緩和することは将来のデバイステクノロジで取り組む価値があるはずであると結論付ける。全体として、本研究は、デバイス指向の LCCDBN 構造の有用性と限界だけでなく、実用的な 3D メモリスタデバイスを実現するために解決されるべき本質的な問題も明らかにした。

# 参考文献

[1] G. Brotman, Z. Z. Wang, and L. Xin and Zhong, "PANEL: Cloud-to-edge AI - Applications and User Experiences Driving the Next Generation of Smartphones and Edge Devices," in *4G/5G Summit*, 2017.

[2] J. J. Yang, M. D. Pickett, X. Li, D. A. A. Ohlberg, D. R. Stewart, and R. S. Williams, "Memristive switching mechanism for metal/oxide/metal nanodevices," *Nature Nanotechnology*, vol. 3, no. 7, pp. 429–433, jul 2008.

[3] J. J. Yang, D. B. Strukov, and D. R. Stewart, "Memristive devices for computing," *Nature Nanotechnology*, vol. 8, no. 1, pp. 13–24, jan 2013.

[4] L. Xia, B. Li, T. Tang, P. Gu, P.-Y. Chen, S. Yu, Y. Cao, Y. Wang, Y. Xie, and H. Yang, "MNSIM: Simulation Platform for Memristor-based Neuromorphic Computing System," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 5, pp. 1009–1022, 2018.

[5] H.-S. P. Wong, S. Raoux, S. Kim, J. Liang, J. P. Reifenberg, B. Rajendran, M. Asheghi, and K. E. Goodson, "Phase Change Memory," *Proceedings of the IEEE*, vol. 98, no. 12, pp. 2201–2227, dec 2010.

[6] Y. Huai, "Spin-Transfer Torque MRAM ( STT-MRAM ) : Challenges and Prospects," *AAPPS Bulletin*, vol. 18, no. 6, pp. 33–40, 2009.

[7] H.-S. P. Wong, H.-Y. Lee, S. Yu, Y.-S. Chen, Y. Wu, P.-S. Chen, B. Lee, F. T. Chen, and M.-J. Tsai, "Metal–Oxide RRAM," *Proceedings of the IEEE*, vol. 100, no. 6, pp. 1951–1970, jun 2012.

[8] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, "The missing memristor found," *Nature*, vol. 453, no. 7191, pp. 80–83, may 2008.

[9] G. S. Snider, "Spike-timing-dependent learning in memristive nanodevices," in *2008 IEEE International Symposium on Nanoscale Architectures*. IEEE, jun 2008, pp. 85–92.

[10] F. Alibart, L. Gao, B. D. Hoskins, and D. B. Strukov, "High precision tuning of state for memristive devices by adaptable variation-tolerant algorithm," *Nanotechnology*, vol. 23, no. 7, p. 075201, feb 2012.

[11] M. Prezioso, F. Merrikh-Bayat, B. D. Hoskins, G. C. Adam, K. K. Likharev, and D. B. Strukov, "Training and operation of an integrated neuromorphic network based on metal-oxide memristors," *Nature*, vol. 521, no. 7550, pp. 61–64, may 2015.

[12] A. Fumarola, S. Sidler, K. Moon, J. Jang, R. M. Shelby, P. Narayanan, Y. Leblebici, H. Hwang, and G. W. Burr, "Bidirectional Non-Filamentary RRAM as an Analog Neuromorphic Synapse, Part II: Impact of Al/Mo/Pr 0.7 Ca 0.3 MnO 3 Device Characteristics on Neural Network Training Accuracy," *IEEE Journal of the Electron Devices Society*, vol. 6, pp. 169–178, 2018.

[13] P. Chi, S. Li, C. Xu, T. Zhang, J. Zhao, Y. Liu, Y. Wang, and Y. Xie, "PRIME: A Novel Processing-in-Memory Architecture for Neural Network Computation in ReRAM-Based Main Memory," in *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*. IEEE, jun 2016, pp. 27–39.

[14] G. Pedretti, V. Milo, S. Ambrogio, R. Carboni, S. Bianchi, A. Calderoni, N. Ramaswamy, A. S. Spinelli, and D. Ielmini, "Memristive neural network for on-line learning and tracking with brain-inspired spike timing dependent plasticity," *Scientific Reports*, vol. 7, no. 1, p. 5288, dec 2017.

[15] S. J. Pan and Q. Yang, "A Survey on Transfer Learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, oct 2010.

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS'12 Proceedings of the 25th International Conference on Neural Information Processing Systems*, 2012, pp. 1097–1105.

[17] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *CoRR*, vol. abs/1409.1556, 2014.

[18] G. E. Hinton, S. Osindero, and Y.-W. Teh, “A Fast Learning Algorithm for Deep Belief Nets,” *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, jul 2006.

[19] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, “Greedy Layer-wise Training of Deep Networks,” in *Proceedings of the 19th International Conference on Neural Information Processing Systems*, ser. NIPS’06. Cambridge, MA, USA: MIT Press, 2006, pp. 153–160.

[20] Y. LeCun, C. Cortes, and C. J. Burges, “THE MNIST DATABASE of handwritten digits,” 1998.

[21] A. Krizhevsky, “The CIFAR-10 dataset,” 2014.

[22] V. Nair and G. E. Hinton, “Rectified Linear Units Improve Restricted Boltzmann Machines,” in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ser. ICML’10. USA: Omnipress, 2010, pp. 807–814.

[23] K. He, X. Zhang, S. Ren, and J. Sun, “Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification,” in *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ser. ICCV ’15. Washington, DC, USA: IEEE Computer Society, 2015, pp. 1026–1034.

[24] L. van der Maaten and G. Hinton, “Visualizing Data using t-SNE,” *Journal of Machine Learning Research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[25] K. Ohmori, A. Shinoda, K. Kawai, Z. Wei, T. Mikawa, and R. Hasunuma, “Reduction of cycle-to-cycle variability in ReRAM by filamentary refresh,” in *2017 Symposium on VLSI Technology*, jun 2017, pp. T90–T91.

# 第5章　アナログ深層学習回路の素子数を半減する重み符号固定学習法

## 5.1　緒言

昨今の深層学習技術の多くは、豊富な計算資源を前提として運用されている。すなわち、エッジが収集したデータをネットワークを介してデータセンタへ集約し、高性能な計算機で集中的に学習推論処理を行う方式を基本形としている。これは、膨大なデータを高速処理することで深層学習の効力を最大限に発揮できる点で理に適っていることは疑うべくもないが、一方で、個人情報を始めとするデータの取扱に関するセキュリティ意識の高まりや、運転支援のようなリアルタイム性が要求される応用という観点から、エッジ端末自体に深層学習エンジンを搭載する必要性が高まりつつある。

エッジにおける深層学習においては、エッジ側端末は電源の制約上から計算性能が不十分であることも多く、電池を長寿命化しつつ十分な精度・速度で学習推論処理を実行するための専用処理回路が不可欠である。深層学習の実体であるニューラルネットワークによるデータ処理は、ほとんどがニューロン間の信号伝達を表す積和演算で構成される。したがって、そのデータと命令の局所性を利用すると、GPGPU 技術による並列計算 [1] や専用の並列積和演算回路 [2] による大幅な処理効率向上が可能である。このような技術は主にサーバ側で用いられるが、エッジ側でも同様のアプローチが適用でき、より低消費電力性に重きをおいた手法として、抵抗変化型メモリ (ReRAM) のようなメモリスタを積和演算器として活用する提案がなされている。

メモリスタ [3–6] は抵抗値で情報を記憶する素子であり、エッジ向け SoC に内蔵されるフラッシュメモリを置換する低消費電力な不揮発性メモリとして期待されている。その内部のクロスバー構造と電流・電圧に関する基本則を利用すると、ニューラルネットワークの積和計算を、一度のメモリ読み出し動作でアナログ回路的に演算することができる [7]。さらに、記憶素子の状態そのものがシナプス結合重みを表すために、重みを格納するためのレジスタやメモリを省略することができ、回路面積上の恩恵ももたらされる。以上のことから、本方式はエッジにおける深層学習専用回路として最適

な手法の一つであると考えられ、本章では本方式の回路構造上の欠点を克服する学習法を提案する。

## 5.2　シナプス結合荷重の符号固定学習法

### 5.2.1　アナログ・ニューラルネットワーク回路の基本構造

ニューラルネットワークの基本単位である人工ニューロンと、そのアナログ回路モデルを図 5.1 に示す。人工ニューロンは神経細胞間の信号伝達の最も単純なモデルで、複数のシナプス前ニューロンから信号を受け取り、次のニューロンへ信号を伝達する [8]。各人工ニューロン間の結合はもとの神経構造に従ってシナプスとよばれ、シナプス結合の強度により各信号の伝わりやすさが重みづけされる。実際の神経細胞では、シナプス前ニューロンからの信号によって生じた活動電位がある閾値を超えると「発火」し、次のニューロンへ信号を伝達するが、人工ニューロンにおいては、この発火機能は連続関数による写像で置き換えられることが多い。この活性化関数を $f$、シナプス前ニューロンからの信号を $x_i$、シナプスの結合重みを $w_i$、出力を $y$ とすると、人工ニューロンの神経伝達ダイナミクスは

$$y = f(\sum_i w_i x_i) \tag{5.1}$$

と表される。

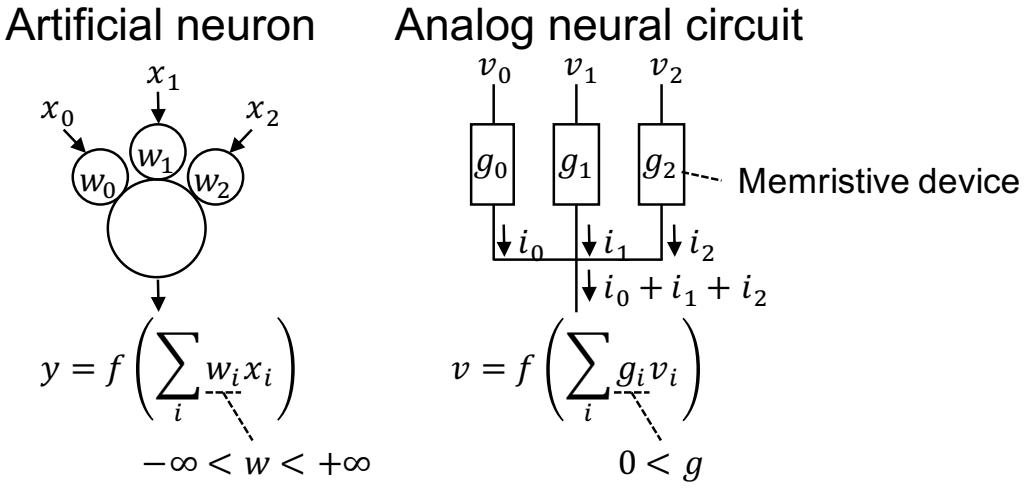



**図 5.1**　人工ニューロンの構造とそのアナログ回路モデル

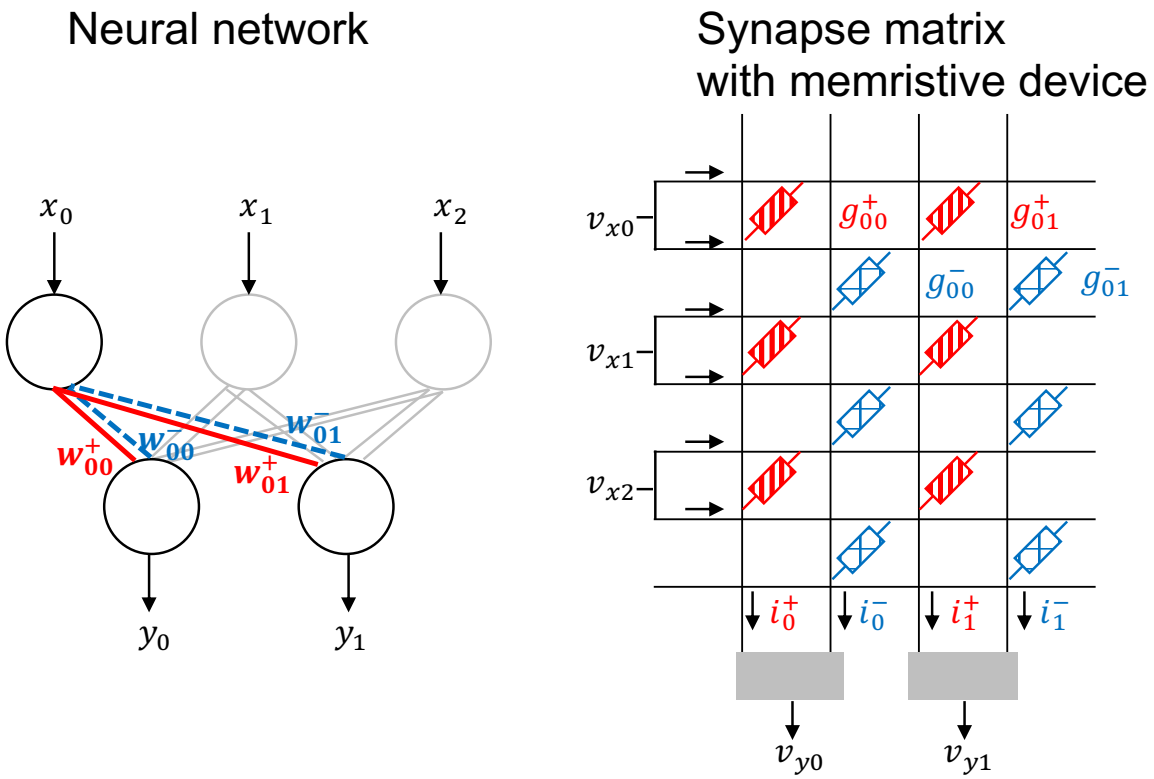



**図 5.2** ニューラルネットワークのアナログ回路モデル：1 個のシナプス結合を 2 個の素子で表現し、差動的に重みの正負を実現する。

人工ニューロンをアナログ回路で表現すると、シナプスはメモリスタに置き換えられる。メモリスタは、Chua によって提唱された受動素子であり、印加される電流や磁場によって素子の抵抗値が変化し、この抵抗値によって情報を記憶する [3–6, 9, 10]。回路上は抵抗器として機能するため、その両端に印加される電圧 $v$ と電流 $i$ およびコンダクタンス $g$ の間には、

$$i = gv \tag{5.2}$$

の関係が成り立つ。さらに、複数のメモリスタ素子を並列に接続すると、その接点における電流はキルヒホッフの法則により

$$i = \sum_i g_i v_i \tag{5.3}$$

と各電流の総和として求められる。この電流を関数 $h$ で表される電流・電圧変換回路

に入力すると、その出力電圧 $v$ は

$$v = h(\sum_i g_i v_i) \tag{5.4}$$

で与えられ、これは明らかに人工ニューロンのふるまいを示す式5.1と一致する。

このような原理に則りアナログ･ニューラルネットワークを作成する場合、人工ニューロンにおける重みの自由度が、メモリスタを用いた回路実装上の不都合をもたらす。ニューラルネットワークの重みは正の値と負の値の両方を取りうるのに対し、今のところ現実世界に負の電気抵抗は存在しないためである。しかも、重みの符号は学習の過程で0をまたいで頻繁に反転するため、十分な学習を終えたあとの各重みの符号を予測することは困難である。この問題を解消するためには、1個の重みを2個の素子で表現しなければならない [11–14](図5.2)。すなわち、2個の異なる素子に同電圧を印加し、各素子の出力電圧に対する差動方式を採用することで、素子対の抵抗値の大小関係により、重みの正負を表現することが可能となる。しかし、このような実装方法では、言うまでもなく電力および面積の両面から非効率的であり、メモリスタ・ニューラル回路の利点が大きく損なわれてしまう。

### 5.2.2 重み符号固定学習による素子数の削減

1個の重みを1個の素子で表現するためには、学習後の重みの符号があらかじめ判明している必要があり、学習過程における重みの符号に何らかの制約を課す方法が考えられる。ここで、人工ニューロンのもととなる神経細胞に立ち返ると、神経細胞間を結合する化学シナプスは、シナプス後ニューロンの発火を促進する興奮性シナプスと、逆に発火を抑制する抑制性シナプスの2種類に分類できる [15, 16](図5.3)。これら2種のシナプスは、シナプスが形成された時点で、どちらの化学的性質を有するかが決定され、神経活動の中で特性が反転することは起こり得ない。

人工ニューロンにおいては、興奮性シナプスが正の重みに、抑制性シナプスが負の重みに置き換えられる。本研究では、前述したシナプスの神経科学的特性に基づき、重みの符号を初期状態で固定する学習法を提案する (図5.4)。ニューラルネットワークの学習では、学習開始前に重みをランダムな分布で初期化する作業を必要とする。本手法では、学習過程で0をまたいで重みの符号が反転することを禁止するため、学習完了後の重みの符号は、初期化直後の符号が保持される。したがって、初期化時の符号がわかっていれば学習後の符号が一意に決定されるため、図に示すように1個の重みを1個の素子で表現することが可能となる。

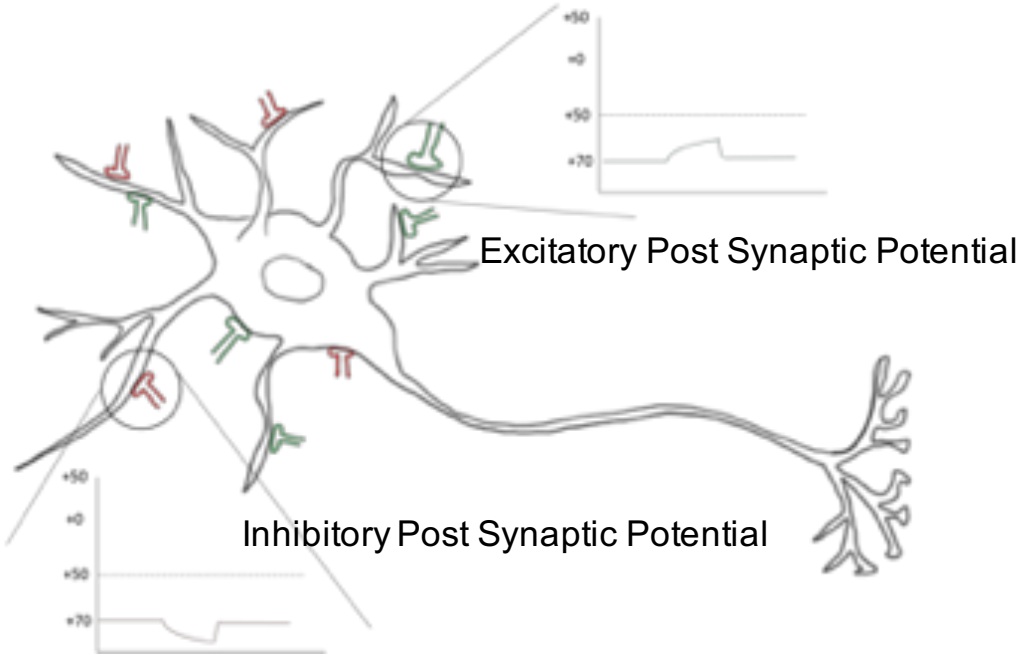



**図 5.3** ニューロン間に形成される化学シナプスは、発火を促進する興奮性シナプスと発火を抑制する抑制性シナプスの 2 種に分類される。

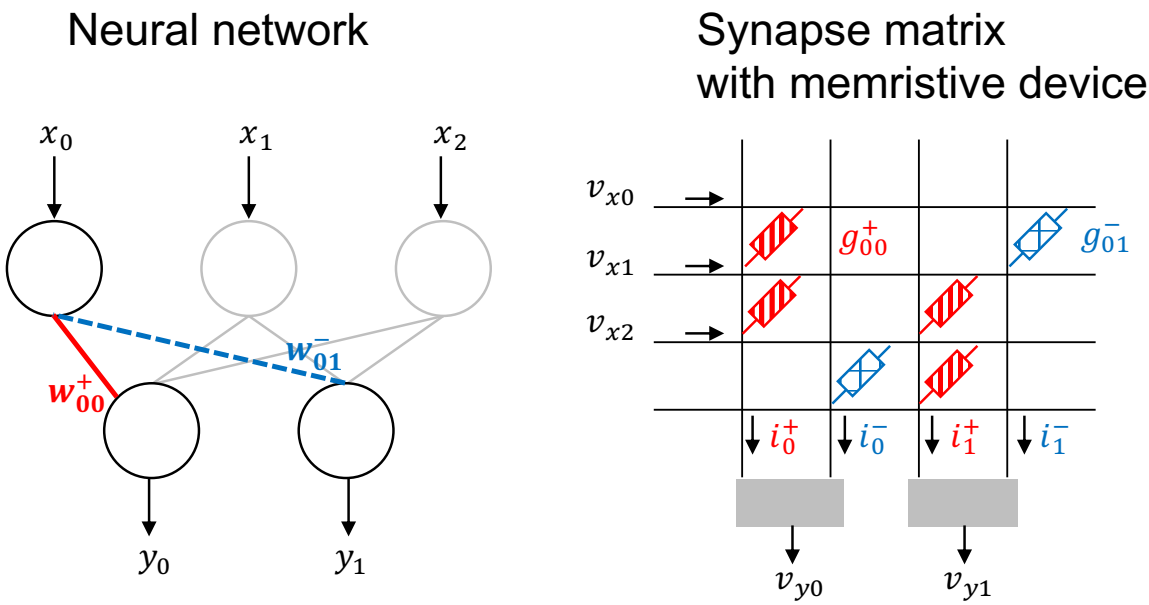



**図 5.4** 提案手法：あらかじめ重みの符号を決めてしまえば、1 個のシナプス結合を 1 個の素子で表現することができる。

## 5.3　全結合順伝播ニューラルネットワークへの適用

### 5.3.1　自己符号化器による層単位の教師なし事前学習

本研究で提案する重み符号固定学習法の有用性を検証するために、2 種の全結合型ニューラルネットワークに対して提案手法を適用し、獲得し得る性能を評価した。深層学習におけるニューラルネットワークは、ニューロンの階層構造で構成される。各層は複数個のニューロンを含み、同一層内のニューロンは結合しておらず、隣り合う層に属するニューロン同士のみでシナプス結合を介した情報伝達が行われる。1 個のニューロンが隣り合う層の全てのニューロンと結合した密なネットワークを全結合型ニューラルネットワークとよぶ。2 次元平面上に展開されたメモリスタの密なクロスバー構造を利用する場合、応用としては全結合ネットワークが最もふさわしい。層間ニューロン同士が疎に結合する畳み込みネットワークについては、メモリの 3 次元積層構造を利用する方が好ましく、これについては第 4 章を参照されたい。

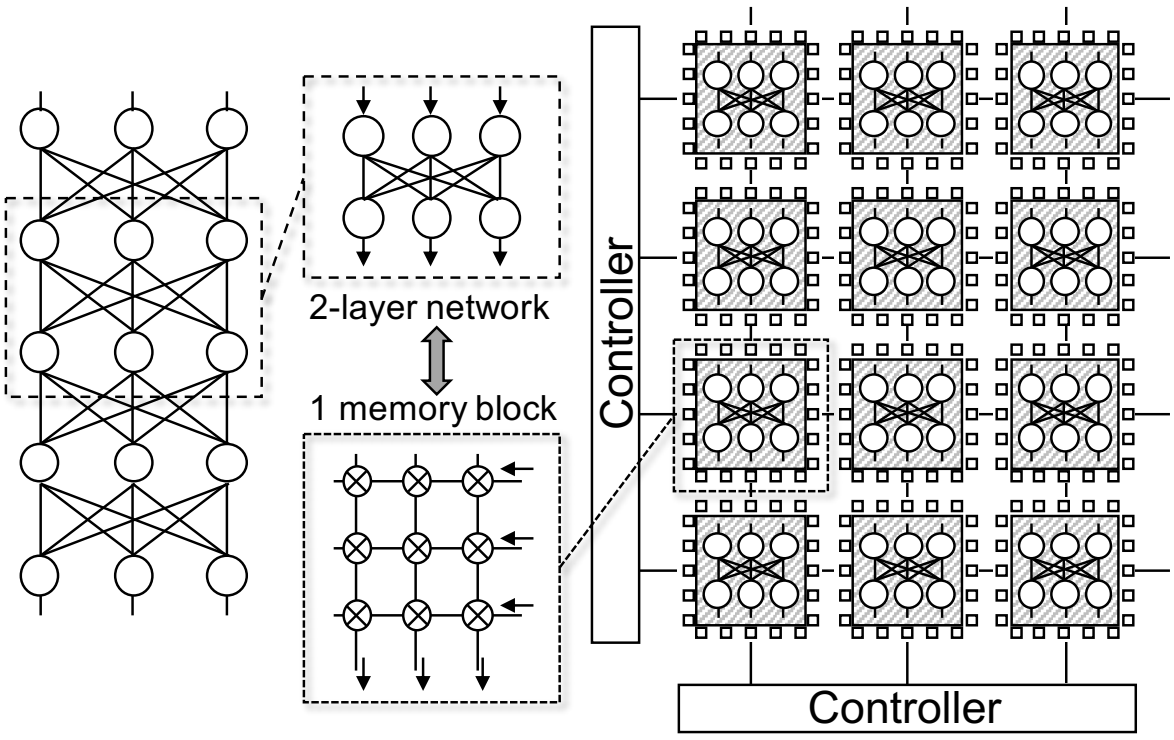



**図 5.5**　メモリを用いた深層ニューラルネットワークの構築

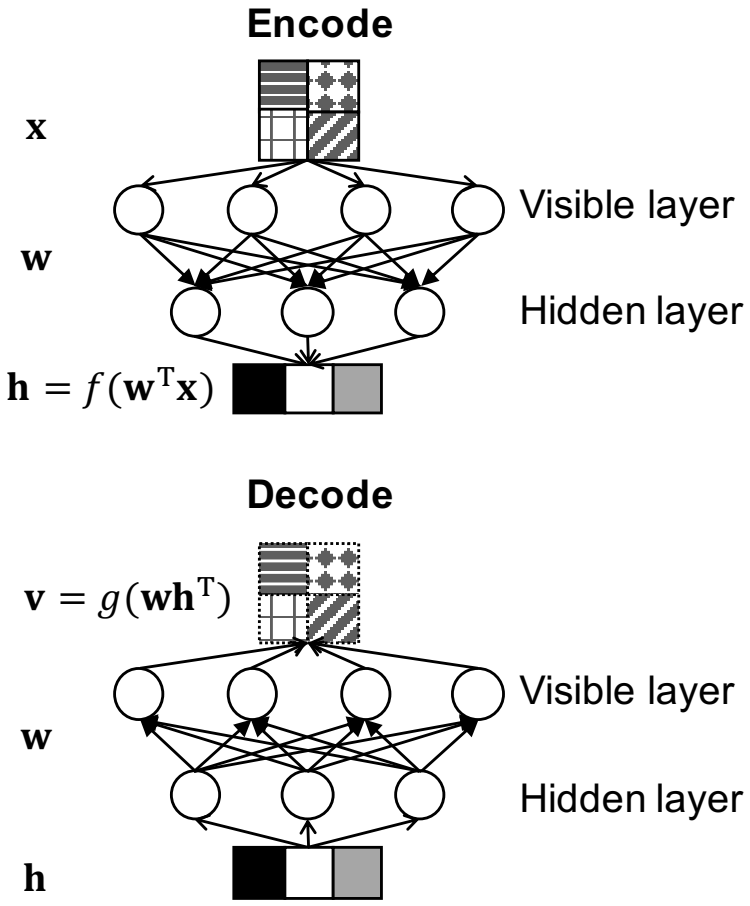



**図 5.6** 自己符号化器：入力された情報は隠れ層で符号化され、可視層で復号化される。

クロスバー構造を用いてシナプス結合を表現すると、1 個のメモリ回路の入出力は、2 層のニューラルネットワークの入出力と等価となる。したがって、このメモリ回路をさらに 2 次元平面上に展開し、各回路を接続することで、容易に深層学習へと拡張することができる (図 5.5)。そのように構成された深層ネットワークに対してある入力信号を与えると、1 回のメモリ読み出し動作でアナログ回路的に積和計算が演算され、推論結果が出力される [14]。しかし、学習処理もエッジに担わせたい場合には、この積和演算回路に加えて勾配計算および重み更新のための外部コントローラが必要となる。全層に対して誤差逆伝播法を適用しようとすると、活性化関数の導関数や勾配計算のための複雑な積和演算回路や、各層 (各メモリ回路) の入出力および計算によって求められる勾配を格納するための巨大なレジスタを外部コントローラに組み込まなければならず、これは消費電力上のオーバーヘッドになりかねない。

そこで本研究では、Hinton らによって考案された自己符号化器による学習を採用す

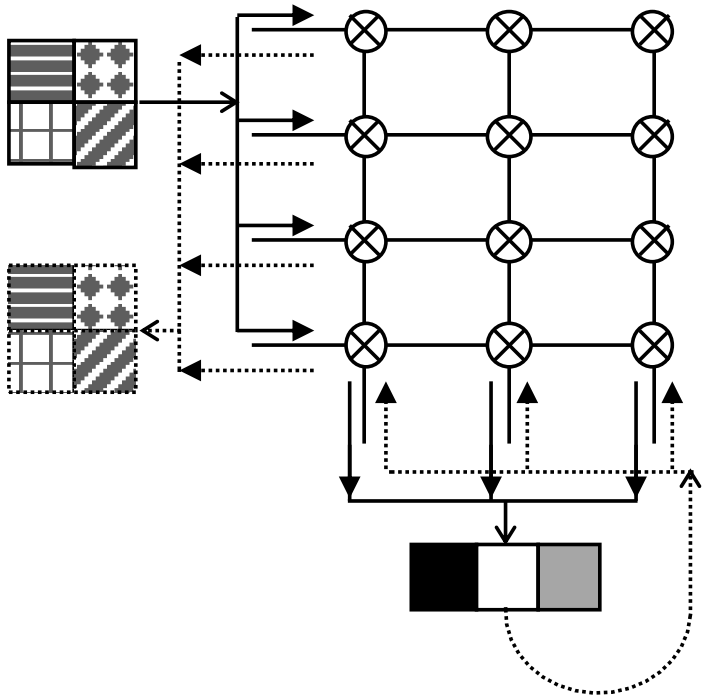


**図 5.7**　メモリを用いた自己符号化器：復号化に必要な重みの転置行列はクロスバー構造の対称性により単一のメモリで表現できる。

る。自己符号化器は本来データの次元圧縮法であるが、教師なしの layer-wise な事前学習法としても用いられる [17, 18] 可視層および隠れ層とよばれる 2 層の密なニューラルネットワークで構成され、通常のニューラルネットワークと同様に、可視層に入力された信号と可視層-隠れ層間の結合重みの積和演算で隠れ層の状態が決定される。さらに、隠れ層から出力された信号を可視層-隠れ層間シナプスの転置行列で表される重みと積和演算することで、可視層と同次元の新たな出力を得る。得られた出力信号が、可視層への入力信号と一致するように重みを更新する学習手法であり、入力信号に対する次元圧縮および復号化を行う自己符号化器 (AE: Autoencoder) としての機能を学習させることができる (図 5.6)。この重みの転置行列を利用する自己符号化演算は、メモリのクロスバー構造の対称性を利用すると、図 5.7 のように出力信号を再帰的にメモリに入力することで再現できるため、複雑な外部回路を用いずとも単一のメモリで自然に実装することができる。

Hinton らは、図 5.8 のように自己符号化器の隠れ層出力新たな入力として、次々と自己符号化器を積み重ねていくことで、深層学習における勾配消失問題が解決されることを示した [18]。メモリを用いたニューラルネットワークの場合は、前述したよう

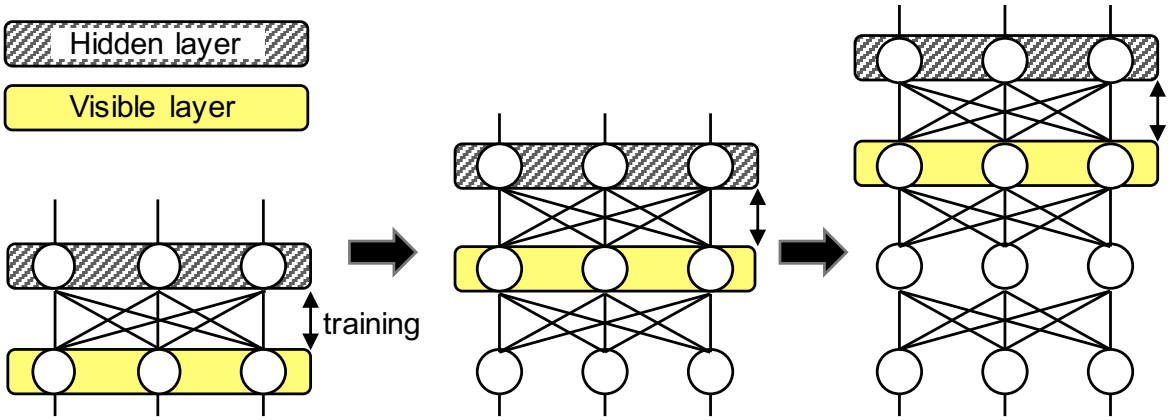



**図 5.8** 自己符号化器を積み重ねていくことで、勾配消失に対して頑強な深層ニューラルネットワークが得られる (Deep Belief Network)。

に単一メモリ内に 2 層のニューラルネットが埋め込まれる。したがって、オートエンコーダにより事前学習したメモリ回路を図 5.9 のように平面上に並べていくことで、容易にネットワークを拡張することができる。

### 5.3.2 MNIST データセットに対する推論能力評価

以上のことから、提案する重み符号固定学習をオートエンコーダおよび制限ボルツマンマシン (RBM: Restricted Boltzmann Machine) に適用し、layer-wise に構築した深層ニューラルネットワークの性能を評価した。RBM [19] はオートエンコーダと同様の教師なし事前学習法であるが、隠れ層の状態がシナプス前ニューロンからの入力にもとづく確率分布に従って決定される点で相違しており、オートエンコーダよりも高い汎化能力を学習させることができる。これら 2 種の教師なし事前学習法を用いて、まずは 2 層からなる最も単純な符号化-復号化機能を学習させ、提案手法の学習可能性を検討した。

学習させるデータとして、機械学習分野において画像認識の最も基本的なタスクとしてよく用いられる、アメリカ国立標準技術研究所が収集および編集した手書き数字データセット (MNIST: Modified National Institute of Standards and Technology database) を採用した [20]。MNIST は、0 9 の 1 桁のグレースケール手書き数字画像と、各画像が表す数字を示す教師ラベルのセットで構成される (図 5.10)。全部で 70000 個の画像・ラベルセットを有し、一般的にはそのうち 60000 個を学習用データセットとして、10000

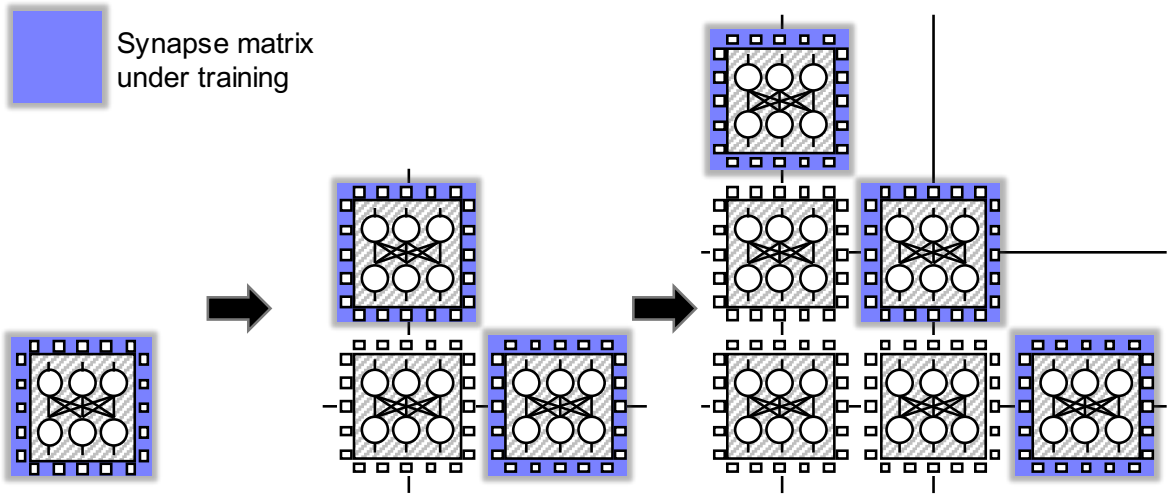



図 **5.9**　layer-wise な教師なし事前学習を用いたメモリ・ニューラルネットワークの拡張

図 **5.10**　MNIST データセットに含まれる手書き数字画像の一部

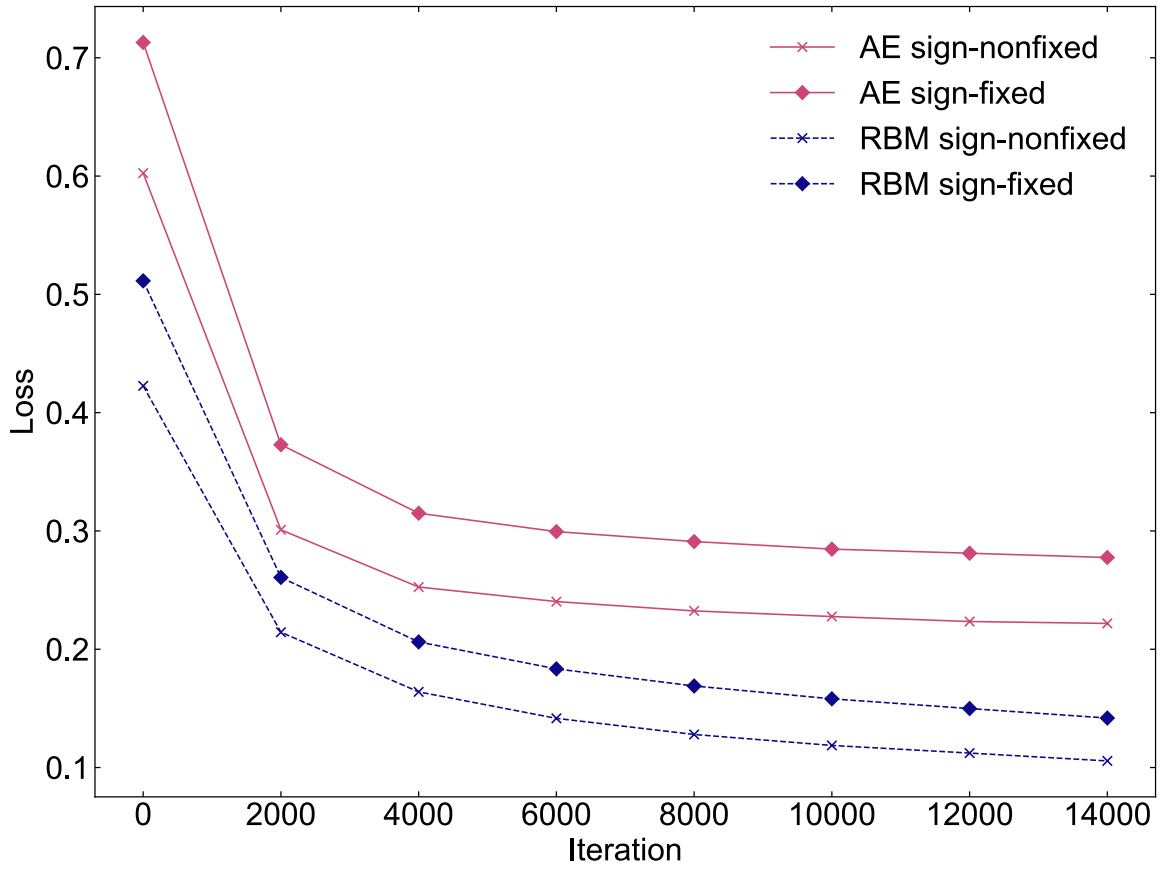



**図 5.11** MNIST 学習中の学習回数と損失値：sign-fixed が本研究で提案する重み符号固定学習を用いた場合を示す。

個を学習後の能力評価用データセットとして用いられる。画像を入力して得られた推論結果と教師ラベルの一致率を比較することで、ネットワークの汎化性能を評価することができる。事前学習を行う場合、可視層へ入力した画像が隠れ層で符号化され、隠れ層から可視層へ返ってきた信号が入力画像を復号するように、ネットワークを学習させる。

本研究では、Hinton らのモデルに従いネットワークの構造を決定した。入力層のニューロン数を MNIST の画像サイズ (28 × 28 pixels) と等しい 784 とし、500、500、2000 個のニューロンからなる 3 層の隠れ層を持ち、出力層として 0 から 9 の数字を one-hot 表現する 10 個のニューロンをもつ、784-500-500-2000-10 のネットワークを構築した [18]。まず 784-500 のネットワークをオートエンコーダまたは RBM で事前学習

**表 5.1**　MNIST の検証用データセットに対する分類精度 (fine-tuning 非実行)

| AE | AE (sign-fixed) | RBM | RBM (sign-fixed) |
|---|---|---|---|
| 92.97% | 92.3% | 96.32% | 95.01% |

させ、次に (784-500) からの出力を新たな入力として (784-500)-500 ネットワークに対して事前学習を行う。このとき、(784-500) の重みは最初のステップで学習済みであるため、新たな更新は行わない。このようにして次々に隠れ層を積み重ねていき、最後に (784-500-500-2000)-10 のネットワークに対して教師ラベルを用いた教師あり学習を行う。

この隠れ層と出力層間の学習における損失と学習回数をプロットすると、図 5.11 に示す結果を得た。図中の sign-nonfixed は重み符号を固定しない一般的な学習法を、sign-fixed は重み符号を初期状態で固定させたまま学習させた場合を表す。図から明らかなように、提案手法を用いると重み符号非固定化と比較して復号化精度は若干劣るものの、学習曲線は発散せずにある一定の数値へ収束しており、提案手法を用いた場合でも学習そのものは可能であることが示唆された。

次に、Hinton らのモデルに従い入力層のニューロン数を 784 とし、それぞれ 500、500、2000 個のニューロンからなる 3 層の隠れ層を持ち、出力層として 0 9 数字を one-hot 表現する 10 個のニューロンをもつ、784-500-500-2000-10 のネットワークを構築した。まず 784-500 のネットワークをオートエンコーダまたは RBM で教師なし学習し、次に (784-500) からの出力を新たな入力として (784-500)-500 ネットワークに対して同様に教師なし学習を行う。このとき、(784-500) の重みは最初のステップで学習済みであるため、新たな更新は行わない。このようにして次々に隠れ層を積み重ねていき、最後に (784-500-500-2000)-10 のネットワークに対して教師ラベルを用いた教師あり学習を行うと、手書き数字画像入力に対して描かれた数字を推論する深層ニューラルネットワークが得られる。

次に、学習で得られた手書き数字分類深層ニューラルネットワークに対して、学習では用いなかった検証用データを入力し、その推論精度を調べると、表 5.1 に示す結果を得た。表中 sign-fixed と書かれた列が重み符号固定事前学習を用いた場合の精度を示す。表からは、提案手法を用いた場合でも符号非固定化と同等の能力を獲得できたこ

とがわかる。前述した 2 層ネットワークにおいては復号化性能は符号非固定化よりも明らかに劣る結果であったが、とはいえ高精度な推論をするのに十分な特徴を学習していることが示された。本実験では、重み符号固定条件下での layer-wise な事前学習の有用性の検討を目的としたため、事前学習完了後の fine-tuning による学習は実行していない。したがって、さらに fine-tuning を取り入れることで、符号非固定化との性能差はより小さくなると期待される。

## 5.4 再帰型ニューラルネットワークへの適用

### 5.4.1 自己符号化 Recurrent Neural Network

次に、単純な順伝播型ネットワークとは異なる全結合型ネットワークとして、再帰型ニューラルネットワーク (RNN: Recurrent Neural Network) [21] への適用を試みた。RNN は、入力層、中間層、出力層の 3 層のニューロン層で構成される (図 5.12)。隣接する各層間のニューロン同士は密に結合しており、さらに中間層出力から中間層入力への帰還路を有する。この帰還路により、次式のように、ある時刻 $t$ の中間層ニューロンの状態 $\mathbf{h}(t)$ が、次の時刻 $t+1$ へ継承される。

$$\mathbf{h}(t+1) = f(\mathbf{w}_i^{\mathrm{T}}\mathbf{x}_{t+1}) + g(\mathbf{w}_h^{\mathrm{T}}\mathbf{h}_t) \tag{5.5}$$

すなわち、RNN は過去の情報を記憶することができ、したがって時系列データに対する推論能力に優れている。

また、帰還路を図 5.13 のように時間方向に展開すると、状態を継承する時刻の範囲 T と同数の中間層を持つ深層全結合ネットワークと見なせるため、前節で述べた順伝播型ネットワークと同様に推論および学習の演算を適用することができる [22]。誤差逆伝播法により学習させる場合には、入力とそれに対応する出力までの時刻幅 (タイムステップ) を決めて、時間展開された中間層を跨いで誤差計算を行う (BPTT: Backpropagation Throw Time)。

RNN に対しても、順伝播ニューラルネットワークのような layer-wise な教師なし事前学習が有効であることが報告されている [23, 24] この手法では、図 5.14 に示すように、時系列のデータが入力されると、ある時刻が経過した後に順序が保持された入力データが再生されて出力される。このとき、ある 1 時点の入力と、それに対応する再生された出力間の情報伝達経路のみを抽出すると、前節で述べた深層オートエンコーダの構造と一致することがわかる。したがって、抽出された構造に対してオートエンコーダ

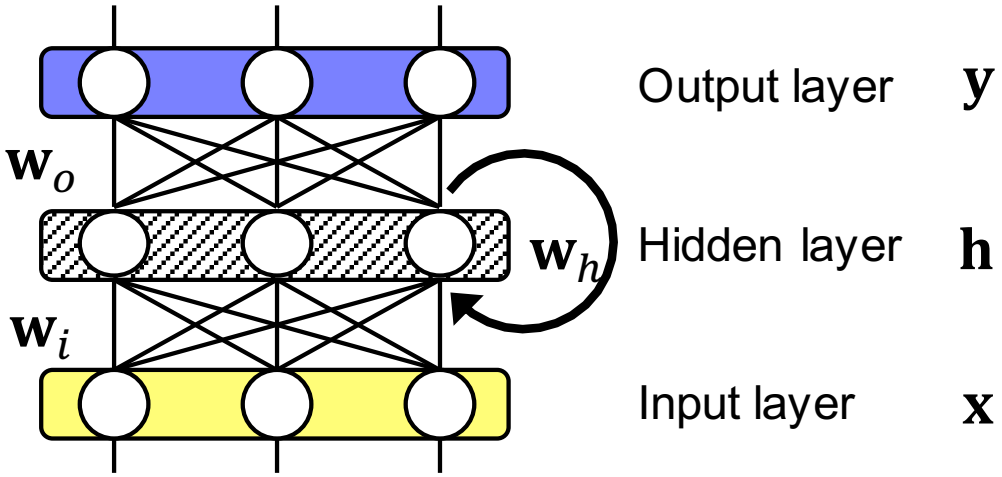



**図 5.12**　再帰型ニューラルネットワーク (RNN: Recurrent Neural Network)

による layer-wise な教師なし事前学習を行うことができ、本研究では、この RNN オートエンコーダに対して重み符号固定学習が可能かどうかを検証した。

### 5.4.2　Penn Tree Bank を用いた推論能力評価

学習させるデータセットとして Penn Treebank(PTB) を採用した。PTB は、ペンシルバニア大学が提供するコーパスである [25]。Wall Street Journal や Brown Corpus をもとに文章を構造化・集積したもので、品詞と統語構造をタグ付けした最も有名なコーパスの一つに数えられ、自然言語処理分野におけるベンチマークとして広く用いられている。PTB を用いた RNN の性能評価としては、Perplexity という指標が利用される。文章中に含まれる単語間の関係性には時間的な相関関係がある。例えば冠詞の次には必ず名詞が続くように、ある単語の次に出現する単語は、文脈からある程度予測される。したがって、RNN に対して単語を出現順に入力し、最後に入力された単語の次の単語を推論させることができる。このとき、Perplexity は次に出現する単語の候補数として定義される。たとえば、文章中の全単語の集合 bag-of-words から正解単語 $w$ を選び出す条件付き確率が $P(w|bag_of_words) = 0.01$ であった場合、これは 100 個の単語の中から 1 個の単語を選び出すことに等しく、$Perplexity = 100$ となる。このように、

$$Perplexity = \frac{1}{P(w|bag_of_words)} \tag{5.6}$$

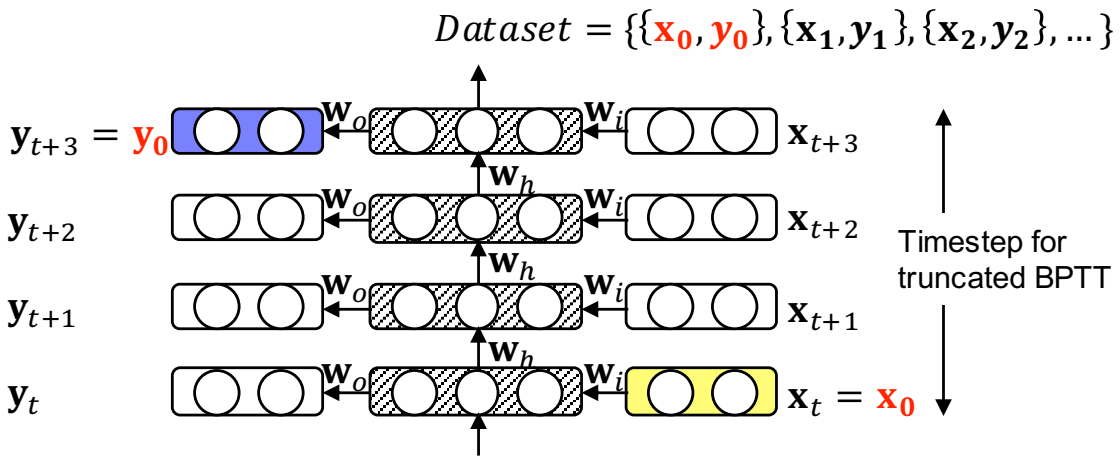



**図 5.13** RNN の帰還路を時間方向に展開すると深層順伝播型ネットワークと見なすことができる。

と表され、Perplexit の値が小さいほど候補を絞り込めたこと、すなわち性能が高いことを指す。より厳密には、推論された単語の生起確率を $p$ とおくと、損失関数は

$$\text{loss} = -\frac{1}{N}\sum_{i=1}^{N} \ln p_{\text{target}_i} \tag{5.7}$$

とクロスエントロピー誤差で定義され、Perplexity は

$$Perplexity = e^{\text{loss}} \tag{5.8}$$

で与えられる。

本検証で構築するネットワークは、入力層に 1 個、中間層 65 個、出力層に 1 個のニューロンを持ち、タイムステップは 25 時刻とした。コーパスをニューラルネットワークに入力する場合、元のデータを入力層のニューロン数に合わせてベクトル化する作業が必要となる [26, 27]。この作業は本検証と本質的な関係は無いため、簡略化のために機械学習フレームワーク Tensorflow [28] を用いてネットワークの構築、学習を行い、コーパスのベクトル化は tf.nn.embedding_lookup 関数により行った。また、RNN における layer-wise 事前学習は一般的によく使われる手法とはいえないため、事前学習を行わない BPTT に対しても重み符号固定学習法を試行した。

学習用データに対する学習の様子をプロットすると図 5.15 に示す結果を得た。プロットした RNN モデルは、それぞれ

- RNN sign-nonfixed: 重み符号非固定下で事前学習をせずに BPTT により学習

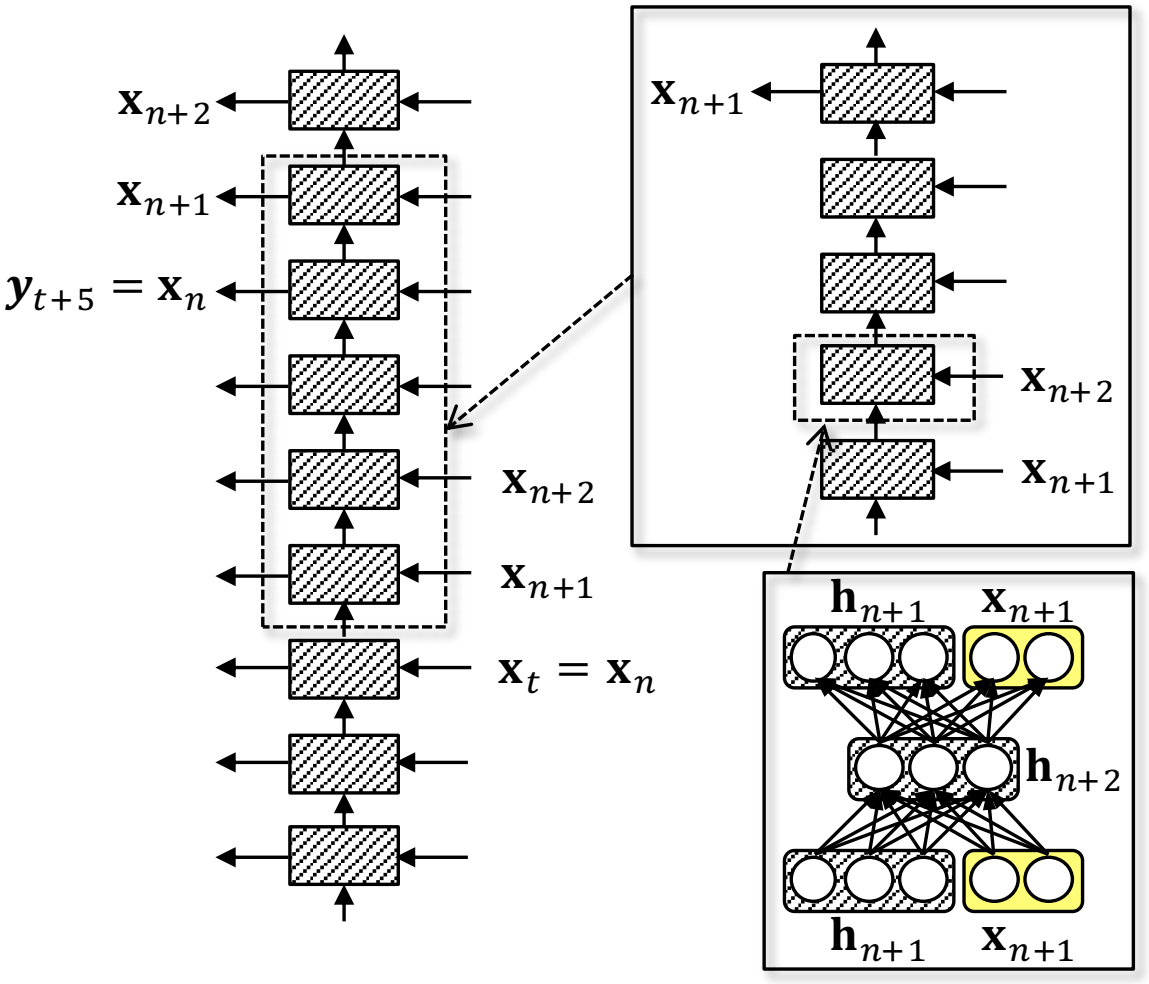



**図 5.14**　時間展開された RNN に対するオートエンコーダを用いた事前学習

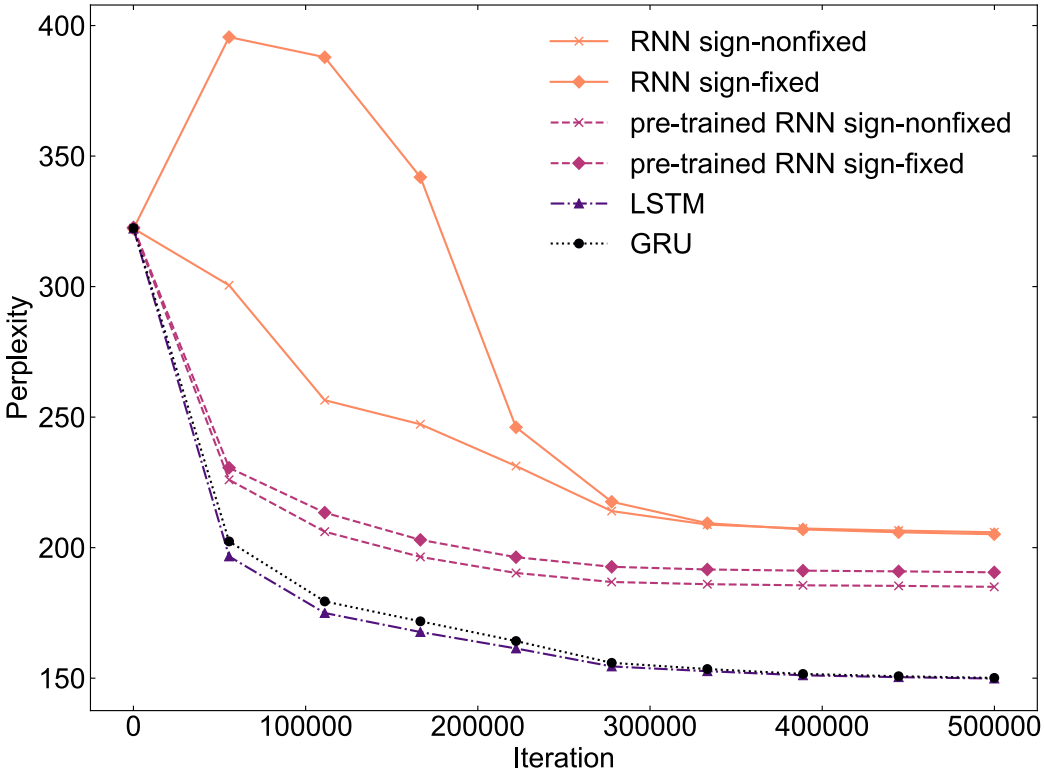



**図 5.15** PTB 学習中の学習回数と Perplexity

- RNN sign-fixed: 重み固定下で事前学習をせずに BPTT により学習
- pre-trained RNN sign-nonfixed: 重み符号非固定下でオートエンコーダによる事前学習をした後、BPTT により学習
- pre-trained RNN sign-fixed: 重み符号固定下でオートエンコーダによる事前学習をした後、BPTT により学習

を表す。LSTM(Long Short-term Memory) [29] は BPTT における勾配消失問題を解決するために提案された、忘却機構を取り入れた RNN であり、通常の RNN よりも高い性能を示すとして、現在の RNN の主流を担っている。GRU(Gated Recurrent Unit) [30] は LSTM の記憶/忘却機構をより簡素にしたモデルであり、LSTM よりも軽度な計算負荷で同等の性能を学習し得ることが知られている。図からは、以下の 2 点が読み取れる。

- 重み符号を固定しても十分な学習が可能である。

**表 5.2**　PTB の検証用データセットに対する Perplexity

| RNN | | | | LSTM | GRU |
|---|---|---|---|---|---|
| nonfixed | fixed | pre-trained, nonfixed | pre-trained, fixed | | |
| 394 | 496 | 268 | 286 | 247 | 214 |

- layer-wise な事前学習により汎化性能が向上し得る。

また、学習完了後に検証用データセットを用いて汎化性能を調べると、表 5.2 のようになった。以上の結果から、再帰型のニューラルネットワークにおいても、重み符号固定条件を課した学習が可能であることが示された。さらに、提案手法は RNN における layer-wise 事前学習に対しても有効であり、メモリスタによるニューラルネットワーク実装のスケーラビリティに適用できる展望を得た。

## 5.5　結言

本章では、メモリスタを用いたニューラルネットワークのための、素子数を半減する重み符号固定学習法について述べた。従来のメモリスタ・ニューラルネットワークでは、1 個の重みを 2 個の記憶素子を用いて正負の符号を表現していた。本研究では、重みの符号を初期状態で固定し、学習中に 0 をまたいで符号が反転するような更新を禁止する制限を課する学習法を提案した。この重み符号固定学習法を順伝播ネットワークと再帰型ネットワークに適用した。まず、深層順伝播ネットワークを自己符号化器により階層別に学習して Deep Belief Network を構築する手法に対し、重みの符号を固定しない場合と固定した場合とで性能を比較した。MNIST データセットに対する分類精度を比較すると、重みの符号を固定しても非固定学習と同等の精度を学習しうることを示した。また、再帰型ネットワークに対しても自己符号化器による事前学習法を採用し、かつ重み符号固定学習を適用すると、 Penn Tree Bank を用いた単語予測タスクに対し、重み符号固定下でも十分な性能を学習しうる結果を得た。提案手法を用いると、入力ラインおよび出力ラインを圧縮することにより実装面積を 1/4 に抑えることができるため、メモリスタ・ニューラルネットワークのさらなる低電力化を期待することができる。

# 参考文献

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS'12 Proceedings of the 25th International Conference on Neural Information Processing Systems*, 2012, pp. 1097–1105.

[2] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers *et al.*, "In-Datacenter Performance Analysis of a Tensor Processing Unit," in *Proceedings of the 44th Annual International Symposium on Computer Architecture*, ser. ISCA '17. New York, NY, USA: ACM, 2017, pp. 1–12.

[3] L. Chua, "Memristor-The missing circuit element," *IEEE Transactions on Circuit Theory*, vol. 18, no. 5, pp. 507–519, 1971.

[4] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, "The missing memristor found," *Nature*, vol. 453, no. 7191, pp. 80–83, may 2008.

[5] J. J. Yang, M. D. Pickett, X. Li, D. A. A. Ohlberg, D. R. Stewart, and R. S. Williams, "Memristive switching mechanism for metal/oxide/metal nanodevices," *Nature Nanotechnology*, vol. 3, no. 7, pp. 429–433, jul 2008.

[6] J. J. Yang, D. B. Strukov, and D. R. Stewart, "Memristive devices for computing," *Nature Nanotechnology*, vol. 8, no. 1, pp. 13–24, jan 2013.

[7] G. Pedretti, V. Milo, S. Ambrogio, R. Carboni, S. Bianchi, A. Calderoni, N. Ramaswamy, A. S. Spinelli, and D. Ielmini, "Memristive neural network for on-line learning and tracking with brain-inspired spike timing dependent plasticity," *Scientific Reports*, vol. 7, no. 1, p. 5288, dec 2017.

[8] F. Rosenblatt, "The Perceptron: A Probabilistic Model for Information Storage and Organization in The Brain," *Psychological Review*, pp. 65–386, 1958.

[9] P. H. Nielsen and N. M. Bashara, "The reversible voltage-induced initial resistance in the negative resistance sandwich structure," *IEEE Transactions on Electron Devices*, vol. 11, no. 5, pp. 243–244, 1964.

[10] J. Åkerman, "Toward a Universal Memory," *Science*, vol. 308, no. 5721, pp. 508 LP – 510, apr 2005.

[11] P. Chi, S. Li, C. Xu, T. Zhang, J. Zhao, Y. Liu, Y. Wang, and Y. Xie, "PRIME: A Novel Processing-in-Memory Architecture for Neural Network Computation in ReRAM-Based Main Memory," in *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*. IEEE, jun 2016, pp. 27–39.

[12] L. Xia, B. Li, T. Tang, P. Gu, P.-Y. Chen, S. Yu, Y. Cao, Y. Wang, Y. Xie, and H. Yang, "MNSIM: Simulation Platform for Memristor-based Neuromorphic Computing System," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 5, pp. 1009–1022, 2018.

[13] L. Song, X. Qian, H. Li, and Y. Chen, "PipeLayer: A Pipelined ReRAM-Based Accelerator for Deep Learning," in *2017 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, feb 2017, pp. 541–552.

[14] R. Mochida, K. Kouno, Y. Hayata, M. Nakayama, T. Ono, H. Suwa, R. Yasuhara, K. Katayama, T. Mikawa, and Y. Gohou, "A 4M Synapses integrated Analog ReRAM based 66.5 TOPS/W Neural-Network Processor with Cell Current Controlled Writing and Flexible Network Architecture," in *2018 IEEE Symposium on VLSI Technology*, jun 2018, pp. 175–176.

[15] J. Chavas and A. Marty, "Coexistence of excitatory and inhibitory GABA synapses in the cerebellar interneuron network." *The Journal of neuroscience : the official journal of the Society for Neuroscience*, vol. 23 6, pp. 2019–2031, 2003.

[16] S. M. Thompson and B. H. Gahwiler, "Activity-dependent disinhibition. I. Repetitive stimulation reduces IPSP driving force and conductance in the hippocampus in vitro," *Journal of Neurophysiology*, vol. 61, no. 3, pp. 501–511, 1989.

[17] G. E. Hinton and R. R. Salakhutdinov, "Reducing the Dimensionality of Data with Neural Networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[18] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A Fast Learning Algorithm for Deep Belief Nets," *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, jul 2006.

[19] P. Smolensky, "Chapter 6: Information Processing in Dynamical Systems: Foundations of Harmony Theory," in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, J. L. M. David E. Rumelhart, Ed. MIT Press, 1986, vol. 1, pp. 194–281.

[20] Y. LeCun, C. Cortes, and C. J. Burges, "THE MNIST DATABASE of handwritten digits," 1998.

[21] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, p. 533, oct 1986.

[22] P. J. Werbos, "Generalization of backpropagation with application to a recurrent gas market model," *Neural Networks*, vol. 1, no. 4, pp. 339–356, 1988.

[23] N. Srivastava, E. Mansimov, and R. Salakhutdinov, "Unsupervised Learning of Video Representations Using LSTMs," in *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ser. ICML'15. JMLR.org, 2015, pp. 843–852.

[24] A. M. Dai and Q. V. Le, "Semi-supervised Sequence Learning," in *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds. Curran Associates, Inc., 2015, pp. 3079–3087.

[25] U. of Pennsylvania, "The Penn Treebank Project."

[26] Y. Goldberg and O. Levy, "word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method," *CoRR*, vol. abs/1402.3, 2014.

[27] X. Huang, F. Alleva, H.-w. Hon, M.-y. Hwang, and R. Rosenfeld, "The SPHINX-II Speech Recognition System: An Overview," *Computer, Speech and Language*, vol. 7, pp. 137–148, 1992.

[28] Google, "TensorFlow," https://www.tensorflow.org/.

[29] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[30] K. Cho, B. van Merrienboer, Ç. Gülçehre, F. Bougares, H. Schwenk, and Y. Bengio, “Learning Phrase Representations using {RNN} Encoder-Decoder for Statistical Machine Translation,” *CoRR*, vol. abs/1406.1, 2014.

# 第6章　総括

現在の高度な人工知能情報処理は、そのほとんどが広範囲なネットワークと潤沢な計算資源の存在を前提としている。これは大量のデータを効率よく処理できる点では優れたシステムであるが、情報セキュリティ意識の高まりや集積されるデータの巨大化に伴い、社会的・技術的課題が明らかになってきた。本研究では、エッジに人工知能が埋め込まれたよりスマートな情報化社会の実現に向けて、エッジ環境における高電力効率なプロセッサの開発を目的とした。

現在の人工知能技術の基本形であるニューラルネットワークは、演算に要するデータと命令の局所性から、メモリの内部で回路的に積和演算を処理することで低電力化が可能であることが知られている。一方で、ニューラルネットワークに入力するデータの前処理や、推論結果を利用した後処理などはCPUが担うため、CPU自体の高電力効率化も不可欠である。そこで、まずはCPUの演算性能を保持したまま低消費電力化を実現させるアクセラレータおよび分岐予測器の開発に取り組んだ。

第2章では、プログラムのコントロール・データフローに着目し、命令間のオペランド依存性が変化するデータパスのみが動的に再構成されるアクセラレータについて述べた。本アクセラレータは、二次元アレイ状に配置された数十から数百個のプロセッシング・エレメント(PE)と、演算の途中結果を保存するためのレジスタ・アレイで構成され、CPUの代わりにプログラムの一部を並列処理する。各PE間のダイレクトな接続はプログラム実行開始時に固定される。一方で、レジスタを介した間接的な接続は、条件分岐命令が実行されて命令間のデータ依存性が変化した場合に、動的に再構成される。このように、動的再構成されるデータパスを最小限の範囲に収めることで、多様なプログラムに対する汎用性を保持したまま、電力効率をおよそ10倍向上させることに成功した。

第3章では、単純ベイズ分類器を応用した学習型分岐予測器による低消費電力化について述べた。動的な分岐予測器は、過去の分岐パターンを学習し、それに基づいて次の分岐方向を予測する。この学習および予測に機械学習の手法を取り入れる手法が、分岐予測コンテストで高い結果を示し続けている。しかし、機械学習による特徴の十

分な学習には大量の情報が必要であり、これらを保存するためのレジスタが消費電力を増加させてしまう。そこで、単純ベイズ法を用いた統計的機械学習による分岐予測器を、ソフトコア CPU にレジスタ転送レベルで組み込み、クロック・サイクル・ベースのシミュレーションにより消費電力を評価した。最も単純な静的分岐予測器と比較して回路規模は巨大になるが、予測精度の向上による消費電力削減がこの電力オーバーヘッドを上回り、ベンチマークプログラムを 3 秒間実行し続けた場合に 300 mW 低電力化される見積もりを得た。

第 4 章では、3 次元積層されたメモリスタを深層畳み込みニューラルネットワークの積和演算器として利用するための学習手法について述べた。3 次元積層されたメモリにおいては、ニューラルネットワークのシナプス結合は各層を結合する記憶素子配線として配置される。しかし層間を跨いだ誤差逆伝播法は制御回路の複雑化を招くため、各層を順番に学習させる layer-wize な教師なし事前学習法を用いるほうが好ましい。そこで、層間配線を簡単化するためにフィルタ間の重みを非共有化しつつ、layer-wize に事前学習する手法を開発し、本手法を用いると転移学習にも有用な特徴抽出器が得られることを示した。

第 5 章では、従来の半分の記憶素子数でメモリスタ・ニューラルネットワークを実現する学習手法について述べた。2 次元メモリのクロスバー構造を利用したニューラルネットワークにおいては、各記憶素子がシナプス結合を表現する。しかし、ニューラルネットワークにおけるシナプス結合荷重は正の値も負の値も取りうるのに対し、負の抵抗値は実世界に存在しない。したがって、従来はは 1 個の重みに対して 2 個の素子を用い、各素子の電流差分をとることで仮想的に重みの正負符号を表現する必要があった。本研究では、各重みの符号を初期状態で固定し、学習中に 0 を跨ぐ重み符号の反転を禁止する学習手法を開発した。順伝播型深層ニューラルネットワークに対する手書き数字認識タスク、および再帰型ニューラルネットワークに対する単語予測タスクに対して、提案手法を用いた layer-wise 教師なし事前学習を適用し、非制約下と同等の能力を学習し得ることを示した。

以上に挙げた研究は、エッジに搭載されるプロセッサへの応用を見込んだものである。今後、エッジ向けプロセッサ広く組み込まれるであろうメモリスタを深層学習回路として用い、さらにその他の広範な処理を担う CPU 自体の電力効率も向上させる。これらの相乗効果により、エッジ環境への人工知能の埋め込みに寄与する、知的な情報処理を取り入れたプロセッサの開発を目指した。コンピュータの歴史に鑑みると、かつては大型の計算機であったものがハードウェアの技術的進歩により小型化・高性能

化が進むことで、様々な情報技術が加速度的な進化を遂げてきた。人工知能技術もまた、現在は大電力で高性能なコンピュータに頼っているが、より小型で身近な環境に普遍的に人工知能が埋め込まれることで、さらなる高みが開かれるものと考えられる。本研究が、情報化社会を次のステージへ押し上げるための一つの貢献となることを期待する。

# 謝辞

本研究は、北海道大学大学院情報科学研究科情報エレクトロニクス専攻において浅井 哲也教授の御指導の下に行われたものであり、本研究を遂行するにあたり、終始懇切な御指導を賜りましたことに慎んで感謝の意を表します。また、常日頃より貴重な御意見および御助言を賜りました北海道大学大学院情報科学研究科情報エレクトロニクス専攻 本村 真人教授に厚く御礼申し上げます。

本研究を進めるにあたり、様々な御討論、御協力を頂いた北海道大学大学院情報科学研究科情報エレクトロニクス専攻 池辺 将之教授、ならびに北海道大学大学院情報科学研究科情報エレクトロニクス専攻 高前田 伸也准教授に厚く御礼申し上げます。

本論文の作成にあたり、有益な御討論をして頂いた北海道大学大学院情報科学研究科情報エレクトロニクス専攻高橋 康夫教授に厚く御礼申し上げます。

本研究を通して御協力、御討論を頂いた東京工業大学 原 祐子准教授ならびに東京工業大学 Paniti ACHARARIT 氏に心より感謝を申し上げます。

本研究を遂行するにあたり、様々な御協力を頂いた植吉 晃大氏、山本 佳生氏、安藤 洸太氏をはじめとする集積アーキテクチャ研究室および集積ナノシステム研究室の諸氏に感謝いたします。また、日頃の研究生活を御支援いただいた集積ナノシステム研究室学術研究員 百瀬 啓氏、工学系技術センター技術職員 西田 浩平氏、集積アーキテクチャ研究室秘書 三浦 由貴氏、集積ナノシステム研究室秘書横川 厚子氏に厚く御礼申し上げます。

最後に、研究活動を進めるに際して物心両面にわたり支え励ましあたたかく見守ってくれた両親に感謝いたします。

# 本研究に関する発表論文

## 1. 学術論文

1. Achararit P., Hida I., Marukame T., Asai T., and Hara-Azumi Y.,“Structural exploration of stochastic neural networks for severely-constrained 3D memristive devices,” Nonlinear Theory and Its Applications, vol. E9-N, no. 4, pp. 466-478 (2018).

2. Hida I., Takamaeda-Yamazaki S., Ikebe M., Motomura M., and Asai T.,“An energy-efficient dynamic branch predictor with a two-clock-cycle naive Bayes classifier for pipelined RISC microprocessors,” Nonlinear Theory and Its Applications, vol. E8-N, no. 3, pp. 235-245 (2017).

3. Hida I., Takamaeda-Yamazaki S., Ikebe M., Motomura M., and Asai T.,“A high performance and energy efficient microprocessor with a novel restricted dynamically reconfigurable accelerator,” Circuits and Systems, vol. 8, no. 5, pp. 134-147 (2017).

## 2. 国際会議発表論文

1. Hida I.,“Embedding a Naive Bayes Classifier as a Dynamic Branch Predictor into a Pipelined Microprocessor,” The 2nd GI-CoRE GSQ, GSB, & IGM Joint Symposium -Quantum, Informatics, Biology & Medicine-, Hokkaido University, Sapporo, Japan (Aug. 7-8, 2018).

2. Hida I., Ueyoshi K., Takamaeda-Yamazaki S., Ikebe M., Motomura M., and Asai T.,“Sign-invariant unsupervised learning facilitates weighted-sum computation in analog neural-network devices,” 2017 International Symposium on Nonlinear

Theory and Its Applications, Cancun International Convention Center, Cancun, Mexico (Dec. 4-7, 2017).

3. Hida I., Takamaeda-Yamazaki S., Ikebe M., Motomura M., and Asai T.,"A versatile and energy-efficient reconfigurable accelerator for embedded microprocessors," GI-CoRE GSQ, GSB, & IGM Joint Symposium -Quantum, Informatics, Biology, & Medicine -, Hokkaido University, Sapporo, Japan (Jul. 10-11, 2017).

4. Hida I., Ikebe M., Asai T., and Motomura M.,"A two-clock-cycle naive Bayes classifier for dynamic branch prediction in pipelined RISC microprocessors," 2016 IEEE Asia Pacific Conference on Circuits and Systems, Ramada Plaza Jeju Hotel, Jeju, Korea (Oct. 25-38, 2016).

5. Hida I., Kim D., Asai T., and Motomura M.,"A 4.5 to 13 times energy-efficient embedded microprocessor with mainly-static/partially-dynamic reconfigurable array accelerator," Proceedings of the Asian Solid-State Circuits Conference 2014, pp. 37-40, 85 Sky Tower Hotel, KaoHsiung, Taiwan (Nov. 10-12, 2014).

6. Hirao T., Kim D., Hida I., Asai T., and Motomura M.,"A restricted dynamically reconfigurable architecture for low power processors," 2013 International Conference on ReConFigurable Computing and FPGAs, Hotel Iberostar Cancun, Cancun, Mexico (Dec. 9-11, 2013).

7. Hirao T., Kim D., Hida I., Asai T., and Motomura M.,"A restricted dynamically reconfigurable architecture for low power processors," Proceedings of the 18th Workshop on Synthesis And System Integration of Mixed Information Technologies, pp. 267-268, Hotel Sapporo Garden Palace, Sapporo, Japan (Oct. 21-22, 2013).

## 3. 研究会発表論文

1. Achararit Paniti, 肥田 格, 丸亀 孝生, 浅井 哲也, 原 祐子,"On the neuromorphic 3D devices for locally-connected convolutional neural network," 日本神経回路学会第 28 回全国大会, OIST カンファレンスセンター, (那覇), 2018 年 10 月 24-27 日.

2. 肥田 格, 植吉 晃大, 高前田 伸也, 池辺 将之, 本村 真人, 浅井 哲也,“不揮発アナログシナプスデバイスの素子数を半減する重み符号固定事前学習法とその深層学習への適用,” 日本神経回路学会第 27 回全国大会, 北九州国際会議場, (福岡), 2017 年 9 月 20-22 日.

3. 肥田 格, 池辺 将之, 浅井 哲也, 本村 真人,“高エネルギー効率プロセッサの実現に向けたナイーブベイズ分類器による動的分岐予測,” 2016 年電子情報通信ソサイエティ大会, 北海道大学, (札幌), 2016 年 9 月 20-23 日.

4. 平尾 岳志, 金 多厚, 肥田 格, 浅井 哲也, 本村 真人,“低消費電力プロセッサのための限定的動的再構成アーキテクチャ,” 電子情報通信学会 リコンフィギャラブルシステム研究会, 北陸先端科学技術大学院大学, (能美), 2013 年 9 月 18-19 日.

## 4. 学会発表論文

1. 肥田 格, 高前田 伸也, 池辺 将之, 本村 真人, 浅井 哲也,“ナイーブベイズ分類器を用いた動的分岐予測器の設計と評価,” 電子情報通信学会 ICD/CPSY 学生・若手研究会, 東京工業大学, (東京), 2016 年 12 月 15-16 日.

2. 肥田 格, 平尾 岳志, 金 多厚, 浅井 哲也, 本村 真人,“組み込みプロセッサの低電力化に向けた限定的動的再構成アクセラレータの設計と評価,” LSI とシステムのワークショップ, 北九州国際会議場, (北九州市), 2014 年 5 月 26-28 日.