博 士 論 文

大規模ニューラルネットワークの
ハードウェアアーキテクチャに関する研究

指導教官　堀口 進 教授

北陸先端科学技術大学院大学
情報科学研究科情報システム学専攻

菅原 英子

2004 年 9 月

要 旨

本論文では，パターン認識の分野で用いられる階層型ニューラルネットワーク (Neural Network: NN) に注目し，組み込みシステムへの応用を目的とした，自律再構成機能を有する階層型NNのハードウェアアーキテクチャについて議論する．組み込みシステム向けのNNには，故障に対して自律的にネットワーク構成を変更し，その構成に最適な結合重みを自動的に獲得するという，二つの機能を併せ持つ自律再構成能力が必要不可欠である．まず，故障補償能力を有する階層型NNのハードウェアアーキテクチャについて検討する．一般的な階層型NNはニューロン間接続のための配線領域が膨大になるという問題がある．また，予備ニューロンの追加には多くの配線領域を必要とする．そこで，配線領域の削減と故障補償容易性を考慮したバス結合型NNアーキテクチャを提案し，単純な操作で故障ニューロンを補償する機能シフト法を提案する．故障補償が可能な階層型NNのハードウェア実装を行い，故障数に応じて故障ニューロンの削除やネットワーク構成の変更が可能なことを示す．次に，結合重みの獲得手法について検討する．構成が確定したNNに対して，最適な結合重みを得る手法として，重み学習が知られているが，多くの手法は教師信号とニューロンの出力の誤差を小さくするように結合重みの調整を繰り返すため，多数存在する個々のニューロンに対して重み学習用の複雑な回路の追加が必要になる．本論文では，遺伝的アルゴリズム (Genetic Algorithm: GA) による重み獲得手法に注目する．GAを用いる利点は，NNと独立して動作するため，個々のニューロンに学習回路を追加する必要がない点である．GAを実行するプロセッサを実装し，ハードウェアによる重み獲得が可能なことを示す．さらに，予備ニューロンを用いた故障補償と重み学習による故障補償の両者の利点を併せ持つ自律再構成法を提案する．提案する再構成法では，故障ニューロン数が予備ニューロン数以下であれば，重み学習を必要とせず，機能シフト法のみで高速な故障補償が可能である．また，予備ニューロン数以上の故障が発生した場合でも，機能シフト法によりネットワーク構成を変更した後，それに対する結合重みをGAにより獲得して故障の影響を取り除く．提案手法の回路設計を行い，自律再構成機能を有する階層型NNシステムの実現可能性を示す．

目 次

図 目 次

表 目 次

第 1章

緒言

1.1 研究の背景と目的

現在，パターン認識技術を用いた産業応用が盛んに行われている．例えば，宇宙空間や災害地等における作業ロボットの遠隔操作のためのロボットビジョンや，情報システムやビル等のセキュリティを強化することを目的とした生体認証システムなどが挙げられる．ロボットビジョンへの応用はロボットの視覚情報から物体の動きや空間の奥行き等を認識するものである．また，生体認証システムは従来のパスワード等による認証に代わり，指紋や顔，虹彩，声紋等の生体情報を用いて認証を行うもので，登録された生体情報から個人を特定するシステムである．これらのシステムでは，処理の正確さのほかに，人間の脳のようなより柔軟なパターン認識能力が求められる．

ニューラルネットワーク (Neural Network: 以下，NN) は，現在主流となっている情報処理方式であるノイマン型アーキテクチャとは異なる情報処理方式の一つである．比較的単純な信号処理素子であるニューロンを多数結合してネットワークを形成し，並列分散型の処理を行う．ノイマン型コンピュータでは，メモリ内のプログラムとデータを逐次的に読み出して実行するため，処理速度の向上が難しい．これに対し，人間の脳の動作原理をモデル化した NN では，各ニューロンの演算処理を並列に実行できるため，非常に高い処理能力が実現できる．ノイマン型コンピュータでも並列処理が行われるが，プログラムを効率よく並列化するのは難しい．また，ノイマン型コンピュータでは，問題を解くためのアルゴリズ

ムが既知であり，それを正確に記述できることが前提となる．一方，NN は解法アルゴリズムが明らかではない問題でも扱うことができるという特徴を持つ．そのため，従来のコンピュータでは解決困難な問題にも適用できると期待されており，文字認識や画像認識といったパターン認識問題や最適化問題，ロボット等の制御などの分野で幅広く研究，応用が行われている．

従来，NN はコンピュータ上のソフトウェア・シミュレーションによって実行されてきた．ソフトウェア・シミュレーションによる実行は様々なネットワーク構造や学習アルゴリズムの評価を容易に実行できることや，応用技術の開発に適しているといった利点を持つが，同時に，ノイマン型コンピュータを用いた直列演算による NN の実行はネットワーク規模が大きくなるにつれて実行時間や学習に要する時間が急激に増加するという欠点を持つ．そのため，リアルタイム処理を必要とする分野において，ソフトウェア・シミュレーションにより実行される大規模 NN を利用することは非常に困難である．

そこで，近年の VLSI 技術の向上を背景に，NN そのものをハードウェア上に実装する研究が行われている [1–7]．NN のハードウェア実装の目的として，NN の実時間処理の実現や組み込みシステムへの応用などが挙げられる．特に，NN の組み込みシステムへの応用は大きな期待が持たれている．1 チップ実装された NN を既存の機器やシステムに組み込むことで，その NN が実現する機能を容易に追加できると考えられる．例えば，ロボットの視覚センサに NN チップを組み込むことで，前述のような物体の動き等を認識できるロボットビジョンを小型システムで実現できるようになる．

NN をハードウェア上に実装する際には，回路規模や配線領域の増加とハードウェア上に発生する故障の問題に対処する必要がある．多数のニューロンを結合した大規模 NN では，ネットワークの大規模化に伴い，ニューロン数はもちろん，ニューロン間結線も大幅に増加するため，回路規模や配線領域を考慮したハードウェア実装を行う必要がある．また，現在の技術では，ハードウェア製造時の欠陥や動作中の故障の発生を完全に避けることは不可能であり，高い信頼性を得るためには何らかの故障補償技術が必要不可欠である．組み込みシステムへの応用を想定した場合には，ホストコンピュータ等を用いることなく，自律的に短時間で故障補償できることが重要になる．

回路規模増加の問題に対しては，NN 回路の主要な構成要素がニューロンであることから，ニューロンの小型化に関する様々な研究が行われており，近似計算による処理の簡略化などの回路規模削減手法が提案されている [8–13]．また，この問題は今後の集積技術の発展によりある程度解消される問題であるともいえる．これに対し，配線領域の増加は断線等の故障の増加につながる問題であり，高集積化が進み多数のニューロンからなる大規模なハードウェア NN システムの構築が可能になるにつれ，ますますその対策が重要になると考えられる．配線領域の削減手法として，ニューロンの入出力にパルス信号を用いるパルスニューロンモデルが注目されている [10,11,13]．しかし，パルスニューロンモデルの実現にはいくつかの問題が指摘されており，現在も様々な研究が行われている．

NN の故障補償に関しては，様々な研究が行われている．ハードウェア NN システムにおける故障補償手法は，これまでの研究から，大きく二つのアプローチに分類できる．一つは，もとの NN 回路に予備回路を追加して故障の発生に備えるハードウェア的アプローチである [17–21]．この手法は回路に冗長性を持たせることから，冗長手法と呼ばれる．冗長手法では高速な故障補償が可能であるが，付加した予備回路以上の故障の発生には対処できないという欠点を持つ．また，予備回路を付加することによる回路規模の増加も考慮しなければならない．つまり，回路規模と故障補償能力の間にトレード・オフの関係が存在する．もう一方の手法は，再学習等によりニューロンの結合重みを更新することで NN から故障の影響を取り除くソフトウェア的アプローチである [22–28]．この手法は重みトレーニング手法と呼ばれる．NN は故障が発生しても学習を繰り返すことで，ある程度その機能を保持できるという能力を有しており，この NN が持つ耐故障性/故障補償能力はハードウェア化された NN においても有効であることが示されている [14–16]．一般に，階層型 NN の学習には誤差逆伝搬法 (Back Propagation: 以下，BP) が用いられるが，学習に時間がかかることから，BP の改良アルゴリズムや BP によらない高速な重み更新アルゴリズム，並列化による高速化手法などが提案されている [22–28]．ただし，アルゴリズムの複雑さや特別な計算機資源を必要とすることなどから，これまでに提案されているアルゴリズムのすべてがハードウェア NN システムにおける重みトレーニングとして使用できるとは限らない．学習アルゴリズム等を用いた重み更新手法のほかに，遺伝的アルゴリズム (Genetic Algorithm:

以下，GA) を用いて NN の結合重みを獲得する手法も研究されている [33–35]．一般に，重みトレーニング手法には，結合重みの更新に長時間を要するという問題のほかに，出力がランダムに変化するランダム故障には対処できないとった問題が指摘されている．このように，冗長手法と重みトレーニング手法にはそれぞれ利点と欠点があり，より効率の良い故障補償手法の実現が望まれる．

組み込みシステムへの応用を想定した NN のハードウェア実装ではより高度な信頼性が求められる．また，外部の制御用コンピュータ等を使用しない独立したシステムであることや，故障発生時には人間の介入なしに復旧できることが望まれるため，ホストコンピュータ等を用いずに，それ自身が自律的に短時間で故障補償を行えることが重要になる．すなわち，組み込みシステム向けの NN には，故障の発生に対して必要に応じて自律的にネットワーク構成を変更するとともに，新しいネットワーク構成に最適な結合重みを自動的に獲得するという自律再構成能力が必要不可欠である．これまでの研究では，GA を用いて NN の構造や結合重みを自動的に決定する手法の研究 [36–41] が行われているが，これらの研究はあくまでも NN のソフトウェア・シミュレーションにおける NN の構造や結合重み獲得のための一手法という位置づけであり，ハードウェア NN システムの自律再構成手法に関する研究は十分になされていない．

以上のことから，組み込みシステムへの応用を想定した階層型 NN のハードウェア・システムの実現には，(1) ハードウェア量，特に配線領域を少なく抑える階層型 NN のハードウェアアーキテクチャと，(2) 故障の回避やネットワーク構成の変更と結合重みの獲得を自律的に行うハードウェア・メカニズム，の二点を同時に考えなければならない．

そこで本論文では，パターン認識の分野で一般に用いられる階層型 NN に注目し，組み込みシステムへの応用を前提とした自律再構成機能を有する階層型 NN のハードウェアアーキテクチャを確立することを目的とする．ハードウェア NN システムにおける自律再構成とは，ハードウェア上に発生する故障に対して，外部のホストコンピュータ等を用いることなくそれ自身が自律的に故障補償を行う仕組みである．故障の発生に対して，必要に応じて自律的にネットワーク構成を変更するとともに新しいネットワーク構成に最適な結合重みを自動的に獲得することのできるハードウェア NN システムを再構成型 NN と呼ぶ．このような再構成

型 NN を実現するための故障補償手法とそのハードウェアアーキテクチャを提案し，そのプロトタイプシステムの回路設計およびハードウェア実装を通じて，自律再構成機能を有する再構成型 NN の実現可能性について論じる．

1.2 論文の構成

本論文は 6 章で構成される．第 2 章では，階層型 NN とそのハードウェア・システムについて概説し，ハードウェア実装の際に考慮すべき問題とその解決法について検討する．特に，組み込みシステムへの応用を想定した階層型 NN のハードウェア・システムの実現のために考慮すべき点を明らかにする．第 3 章では，ハードウェア実装向き階層型 NN のハードウェアアーキテクチャと，そのアーキテクチャ上での冗長手法による故障補償について論じる．まず，配線領域と故障補償容易性を考慮したバス結合型 NN アーキテクチャを提案する．そして，そのアーキテクチャ上での故障補償手法として，予備ニューロンを用いて単純な操作で故障ニューロンを補償する機能シフト法を提案し，冗長手法による故障補償能力を持つ階層型 NN システムを構築する．第 4 章では，結合重みの獲得手法について検討する．構成が確定した NN に対して最適な結合重みを獲得する手法として，重み学習が知られているが，ここでは，GA による NN の重みトレーニングに注目する．GA のハードウェア化を行い，ハードウェア GA による重みトレーニング・システムを構築する．第 5 章では，階層型 NN の自律再構成のためのハードウェア・メカニズムとして，冗長手法による故障補償と重みトレーニングによる故障補償の両者の利点を併せ持つ自律再構成法を提案する．これは，第 3 章で述べる予備ニューロンを用いた故障補償と第 4 章で述べる GA による重みトレーニングを組み合わせた手法である．また，提案する自律再構成法を実現した再構成型 NN のハードウェアアーキテクチャを提案する．提案手法の回路設計を行い，自律再構成機能を有する再構成型 NN システムを構築する．最後に，第 6 章において本論文の結論を述べる．

第 2章

階層型ニューラルネットワークのハードウェア・システム

2.1 はじめに

ニューラルネットワーク (Neural Network: 以下，NN) とは，比較的単純な信号処理素子であるニューロンが多数結合して信号をやりとりするネットワークである．個々のニューロンの動作は単純なものであるが，それらが多数結合されることによって全体として複雑な情報処理を可能とする．NN の主な応用分野として，パターン認識や最適化問題，ロボット等の制御などが挙げられる．ニューロンの結合方式によっていくつかのネットワーク構造があり，代表的な構造としては，層間結合によるモデルである階層型 NN と，リカレント結合によるモデルであるホップフィールドネットワークがある．階層型 NN は主にパターン認識装置や制御装置として広く研究されている．ホップフィールドネットワークは最適化問題の解法装置として注目されている．現在，パターン認識の産業応用が盛んに行われていることから，本論文では階層型 NN をターゲットとし，そのハードウェア・システムについて議論する．

認識・制御など，階層型 NN の応用が期待される分野にはリアルタイム処理を必要とするものが多い．しかし，これまで主流であったコンピュータ上でのソフトウェア・シミュレーションによる NN の実行は，ネットワーク規模が大きくなるにつれて実行時間や学習に要する時間が急激に増加するという欠点を持ち，これ

が実世界のアプリケーションへの応用を困難にしていた．そこで，近年のVLSI技術の発展に伴い，NNそのものをハードウェア上に実装する研究が行われている．NNのハードウェア化の目的として，高速処理の実現や組み込みシステムへの応用が挙げられる．特に，1チップ実装されたNNを既存の機器やシステムに組み込むことで，そのNNが実現する機能を容易に追加できるという点で，組み込みシステムへの応用に対する期待は今後ますます大きくなると考えられる．組み込みシステム向けNNの応用例として，物体認識などが可能な遠隔操作ロボットのためのロボットビジョンや，指紋や声紋等の生体情報を用いて認証を行う生体認証システムなどが挙げられる．

NNをハードウェア上に実装する際には，回路規模や配線領域の増加とハードウェア上に発生する故障の問題を考慮する必要がある．特に，組み込みシステムへの応用を想定した場合には，より高度な信頼性が求められる．また，組み込みシステムでは，外部の制御用コンピュータを使用しない独立したシステムであることや，故障発生時には人間の介入なしに復旧できることが望まれるため，ホストコンピュータ等を用いずに自律的に短時間で故障補償を行うことが要求される．すなわち，組み込みシステム向けのNNには，故障の発生に対して，必要に応じて自律的にネットワーク構成を変更するとともに，新しいネットワーク構成に最適な結合重みを自動的に獲得するという，二つの機能を併せ持つ自律再構成能力が必要不可欠である．

本章では，階層型NNの基本的な構造とそのハードウェア・システムについて概説し，階層型NNのハードウェア実装の際に考慮すべき問題を明らかにする．まず，NNの基本素子であるニューロンと，本論文で扱う階層型NNの構成について説明する．次に，従来のハードウェア化手法を示し，階層型NNのハードウェア実装における問題とその解決法について述べる．大規模NNのハードウェア実装の際の回路規模や配線領域の増加の問題に対しては，これまでに提案された回路規模削減手法および配線領域削減手法を挙げ，ハードウェアNNシステムのアーキテクチャについて考察する．ハードウェア上に発生する故障の問題に対しては，従来のハードウェアNNシステムにおける故障補償手法を分類し，それぞれの利点，欠点を明らかにするとともに，より効率の良い故障補償手法について検討する．さらに，組み込みシステムへの応用を想定したハードウェアNNシステムに

おける自律再構成手法について考察する．

2.2 ハードウェア・ニューラルネットワーク・システム

2.2.1 階層型ニューラルネットワークの概要

階層型NNの基本素子であるニューロンは多入力一出力の演算素子である．最も基本的なニューロンモデルはMcCullochとPittsによって提案された[29]．図2.1に基本的なニューロンモデルを示す．n個の入力とそれに対応する結合重みの積和演算を行うことで入力の重み付き総和を計算し，その重み付き総和に対して活性化関数によるしきい値処理を行う．

ニューロンの入出力動作は比較的簡単な数式で表現できる．あるニューロンに対するn個の入力を$x_i(i = 1, \ldots, n)$とし，それぞれの入力に対する結合重みを$w_i(i = 1, \ldots, n)$とすると，ニューロンの出力zは

$$\begin{aligned} u &= \textstyle\sum_{i=1}^{n} w_i x_i - \theta, \\ z &= f(u). \end{aligned} \tag{2.1}$$

と表される．ここで，θはしきい値，関数$f(u)$は活性化関数と呼ばれるニューロンの出力関数である．McCullochとPittsのモデルでは，活性化関数として式2.2のステップ関数が用いられるが，一般には式2.3のシグモイド関数を用いることが多い．aは関数の広がりを決める正の定数であり，$a \to \infty$の極限でステップ関数となる．

$$f(u) = \begin{cases} 1, & x > 0 \\ 0, & x \leq 0 \end{cases} \tag{2.2}$$

$$f(u) = \{1 + \exp(-au)\}^{-1} \tag{2.3}$$

次に，階層型NNの概要について述べる．階層型NNはパターン認識や制御の分野で広く研究され利用されているネットワーク構成である．複数のニューロンを含んだ入力層，中間層，出力層から構成され，各ニューロンは隣接する層のすべてのニューロンと結合重みを通して完全結合される．図2.2に階層型NNの基本的

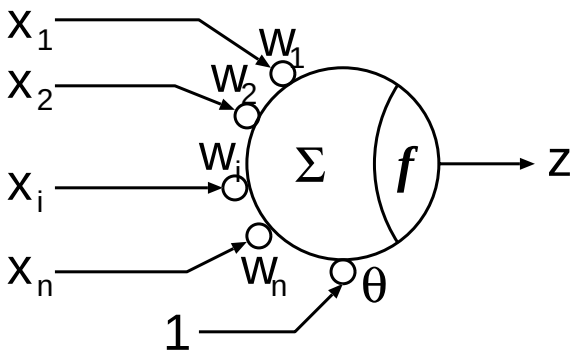


図 2.1: 基本ニューロンモデル

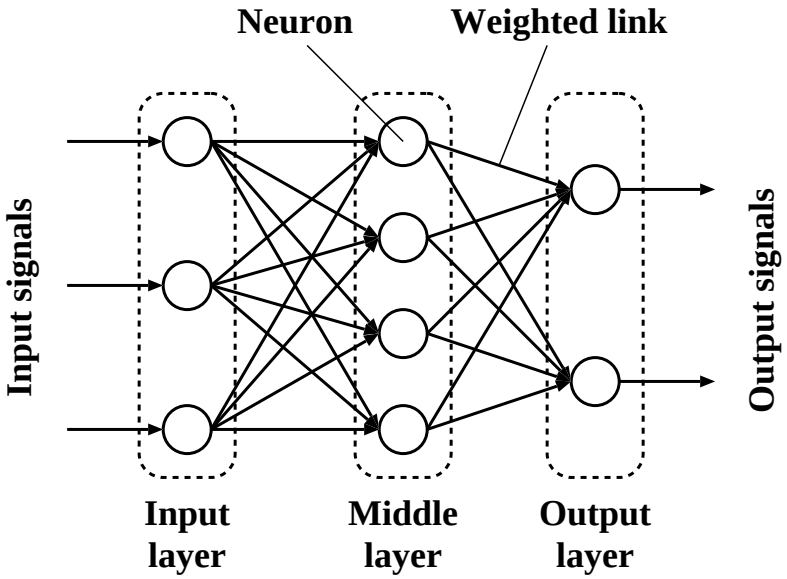


図 2.2: 階層型ニューラルネットワークの基本構成

な構造を示す．一般には図 2.2 に示した中間層を 1 層とした 3 層構造が多く用いられるが，中間層が複数存在する場合もありうる．適用する問題ごとに必要なネットワーク規模を見積もることは困難であるが，多くの用途において 10^3 個以上のニューロンを持つネットワークが必要であるといわれている．

階層型 NN では，入力層に渡された入力パターンが順次前進しながら各層で処理され，最終的な処理結果が出力パターンとして出力層から得られる．入力層のニューロンはネットワーク外部から入力データを受け取り，それを中間層のニューロンへと送信する．一方，中間層と出力層のニューロンは下位層のニューロンからデータを受け取り，式 2.1 に示した演算を行う．中間層のニューロンは演算結果を次層へ送信し，出力層のニューロンはネットワーク外部に出力データを送信する．層内の各ニューロンは互いに独立で，並列に演算を行うことができる．

2.2.2 ハードウェア・モデル

ハードウェア NN システムの実現方式として，アナログ回路で実現するアナログ方式とディジタル回路で実現するディジタル方式がある．アナログ方式の特徴を以下に示す．

- 完全並列アーキテクチャにより，高速処理が可能である．
- ディジタル回路に比べて回路構成素子が少なく，高集積化が可能である．
- よいアナログ記憶素子がない．
- ノイズに弱く，高精度化が難しい．

一方，ディジタル方式は以下のような特徴を持つ．

- ノイズに強く，演算語長に応じて高精度化できる．
- 複雑な処理が可能である．
- メモリ機能が容易に実現できる．
- 時分割多重処理により，等価的に多数のニューロンを構成すること (仮想ニューロン方式) ができる．

アナログ方式は高い処理速度を達成しうるが，回路のオフセット電圧等の精度的な問題があり，特に大規模 NN においてこの問題が顕著になると考えられる．また，大規模メモリの実現が困難であるという欠点もある．一方，ディジタル方式では，アナログ方式のような精度の問題はなく，メモリ LSI などを用いて大規模記憶を実現することも可能である．そのため，大規模 NN の実現にはディジタル方式が適しているといえる．

ディジタル方式の問題点としては，アナログ方式に比べ，回路規模が大きくなることが挙げられる．しかし，現在のディジタル技術を直接利用できることから，ディジタル方式による NN のハードウェア実装が主流となっている．本論文で扱うハードウェア NN システムもディジタル方式である．

最も単純な階層型 NN のハードウェア・モデルとして，複数のニューロンを配線で接続してネットワークを形成する方法が考えられる．これは図 2.2 に示した階層型 NN のネットワーク構成をそのまま実現する方法で，入力層に L 個，中間層に M 個，出力層に N 個のニューロンが存在する場合，入力層・中間層間に L 対 M の結合と中間層・出力層間に M 対 N の結合が存在することになる．この場合，ネットワークの大規模化に伴い，その主要な構成要素であるニューロンの数はもちろん，ニューロン間接続のための配線領域も大幅に増加する．多数のニューロンからなる大規模 NN をハードウェア上に実装するには，何らかの回路規模削減手法，特にニューロンの小型化と配線領域の削減のための工夫が必要となる．

一方，現在の技術ではハードウェア製造時の欠陥や動作中の故障を完全に避けることは不可能である．また，回路規模に比例して欠陥や故障の数が増大することから，高い信頼性を得るためには何らかの故障補償技術が必要不可欠である．特に，組み込みシステムへの応用を想定した場合，より高度な信頼性が求められる．また，システム構成として，外部の制御用コンピュータ等を使用しない独立したシステムであることや，故障発生時に人間の介入なしに復旧できる機能を有することが望まれる．そのため，ホストコンピュータ等を用いずに，自律的に短時間で故障を取り除く能力の有無が非常に重要になる．

2.3 回路規模を考慮したハードウェアNNシステム

ハードウェアNNシステムを構築する際の回路規模の増加の問題に対しては，階層型NNを構成する回路の大部分をニューロンが占めることから，ニューロンの小型化に関する研究が行われている．また，階層型NNは各ニューロンが隣接する層のすべてのニューロンと完全結合される構造であるため，ニューロン数の増加に伴い，ニューロン間結線のための配線領域も大幅に増加することから，配線領域を削減するための工夫が必要となる．

ニューロンの演算では入力と結合重みの積和演算が中心的な計算である．乗算器を使用した構成は回路規模が大きくなるという問題があるため，回路規模の大きい乗算器を用いずにNNを構成する手法に関する研究が行われている [6,8,9,12]．中條ら [9] は二分探索法に基づいたパーセプトロン用ニューロンの計算アルゴリズムを提案している．このアルゴリズムでは積和演算器を使用しないため，ニューロンの回路規模を小さくできる．肥川 [12] は活性化関数に3値関数を用いた多層NNを提案している．活性化関数を3値とすることで，入力と結合重みの乗算がシフトや論理積のような簡単な操作で実現でき，回路規模を低減できる．

また，パルスニューロンモデルを用いたNNも注目されている [10,11,13]．川島ら [11] は入出力にパルス信号を用いたパルスニューロンによる小規模回路で実現可能なNNのハードウェア化手法を提案している．パルスニューロンを用いることで入力と結合重みの乗算とシグモイド関数演算を確率的に計算することができ，乗算器とシグモイド関数用のメモリを必要とせず，回路規模を削減できる．肥川 [13] は多数決ニューロンを用いたパルス駆動型多層NNを提案しており，ニューロンの出力値の演算精度を上げるために多数決回路を用いている．パルスニューロンには，確率的な演算により回路規模を低減できるという利点のほかに，入出力にパルス信号を用いることで一つの入力を1本の信号線で構成できるため配線領域を削減できるという利点もある．しかし，パルスニューロンモデルにはパルス信号の生成法や演算精度の問題が指摘されており，現在，これらの問題を解決するための研究が行われている．

回路規模増加の問題に対しては様々な回路規模削減手法が提案されており，また，今後の集積技術の向上により，ある程度解消される問題であるともいえる．こ

れに対し，配線領域の増加は断線等の故障発生率の増加につながる問題であり，高集積化が進むにつれ，ますますこの問題への対策が重要になる．配線領域増加の問題に関してはパルスニューロンを用いることで配線領域を削減する手法が考えられているが，パルスニューロンの実装方式にはまだ議論の余地がある．以上のことから，本論文では配線領域の増加の問題解決に重点を置いた階層型 NN のハードウェアアーキテクチャについて議論する．

2.4 ハードウェア NN システムにおける故障補償手法

2.4.1 故障補償手法の分類

ハードウェア NN システムにおける故障補償手法は大きく二つのアプローチに分類することができる．一つは，もとの NN 回路に予備の回路を追加し，故障が発生した際に故障箇所を予備の回路で置き換えるハードウェア的な故障補償手法である．このアプローチは予備回路を付加することから冗長手法と呼ばれる．追加する予備回路の種類や置換アルゴリズムが異なるいくつかの手法が提案されている．

もう一方のアプローチは，ニューロンの結合重みを更新することで予備回路を付加することなく NN から故障の影響を取り除くソフトウェア的な故障補償手法である．このアプローチは重みトレーニング手法と呼ばれる．結合重みの更新のために様々な手法が提案されている．

これまでに提案された冗長手法および重みトレーニング手法の詳細は次節以降で述べる．

2.4.2 ハードウェア的故障補償手法

冗長手法には，追加する予備回路の種類や置換アルゴリズムの異なるいくつかの手法が提案されている．Emmerson ら [17] はパターン認識における多層パーセプトロンの隠れ層 (中間層) のすべてのニューロンの再生手法を提案した．Phatak ら [18] は結合重みの総和を利用した中間層ニューロンの再生法を提案した．当麻ら [19]，Fuhrman ら [20] はモジュールを多重化することで冗長性を持たせている．

冗長手法の利点は，故障の種類によらず効果的な故障補償が可能であるという点である．一般に，冗長手法による故障補償は重みトレーニング手法に比べ高速であるといわれている．しかし，故障箇所を置換するための予備回路をNN回路に組み込むため，付加した予備回路の分だけ回路規模が大きくなる．予備ニューロンを追加する場合，追加したニューロンを前後の層のニューロンと接続する必要があり，ニューロン間接続のための配線領域も増加することになる．また，NN回路に付加した予備回路以上の故障の発生には対処できないという欠点もある．この制限を取り除くためには構成ユニットの完全な多重化が必要になるが，これは明らかに通常の倍，もしくはそれ以上の回路規模を必要とする．これらのことから，冗長手法による故障補償を実現する場合，故障補償能力と回路規模のトレード・オフが大きな課題である．

2.4.3 ソフトウェア的故障補償手法

階層型NNには，故障が発生しても学習を繰り返すことで，ある程度その機能を保持できるという能力が備わっており，この耐故障性/故障補償能力に関して研究が行われている．丹ら [14] は耐故障性の実現方法および耐故障ネットワークの性質について明らかにした．NNの持つ自律欠陥救済能力はハードウェア実装されたNN上に発生する故障に対しても適用可能であることは安永ら [15] によって明らかにされた．また，高浪ら [16] は階層型NNに故障を注入して学習を行うことによって耐故障性が得られることを示した．

階層型NNの学習アルゴリズムとして，一般に誤差逆伝搬法 (Back Propagation: 以下，BP) が用いられる．しかし，BPによる階層型NNの学習には長時間を要することから，BPの改良アルゴリズムも含め，様々な学習アルゴリズム，結合重み更新アルゴリズムが提案されている．Simon [23] はネットワーク全体にわたって結合重みを平等に分配する手法を提案している．Hammadi ら [24, 25] はFTCAと呼ばれる新たな学習アルゴリズムを提案している．Khunasaraphan ら [22] はweight shiftingというBPやその改良アルゴリズムとは異なる新しい結合重み更新手法を提案している．これらの手法では，ネットワーク規模が大きくなるにつれて多数の結合重みを更新しなければならず，計算コストが著しく増加する．そこで，計

算コスト削減のために，Muller ら [26]，山森ら [27, 28] によって並列処理による結合重みのトレーニング手法が提案されている．

BP 等の学習アルゴリズムによる重みトレーニング手法には，計算コストの問題のほかに，NN における故障のうち，ある入力に対して出力がランダムに変化するランダム故障には対処できないという問題も指摘されている．また，これまでに提案されたアルゴリズムの多くはソフトウェア・シミュレーションによる NN のための重みトレーニング手法であり，アルゴリズムの複雑さなどから，これらのアルゴリズムのすべてがハードウェア NN システムにおける重みトレーニング手法として利用できるとは限らない．

BP 等の学習アルゴリズムによる重みトレーニングのほかに，遺伝的アルゴリズム (Genetic Algorithm: 以下，GA) による NN の結合重み獲得手法も提案されている [33–35]．GA による重みトレーニングは NN と GA の融合という観点で行われている研究の一例である．先に挙げた BP などの学習アルゴリズムは局所探索手法であるため，局所最適解に陥りやすく，問題が多峰性を帯びている場合にはうまく機能しないなどの欠点が指摘されている．これに対し，GA による重みトレーニングは，GA の持つ大域サンプリングの特徴を活かし，局所解に陥る可能性を低減できるという利点を持つ．これまでの研究では，ソフトウェア・シミュレーションとして実行される NN の結合重みを自動的に獲得するための手法として研究がなされている．GA による重みトレーニングは NN の結合重みを自動的に獲得することが目的であり，NN の故障補償のための結合重み更新とは厳密には異なるが，有効な重みトレーニング手法として注目されている．

重みトレーニング手法では，結合重みの更新のための計算コストの問題のほかに，補償可能な故障パターンの問題もある．ランダム故障のようなハードウェアによる故障補償を必要とする故障パターンも存在するため，重みトレーニング手法のみで完全に故障を回避・修復できるとは限らない．一方，冗長手法と異なり，階層型 NN を構成している回路の多重化を必要としないため，故障補償能力と回路規模との間のトレード・オフの問題はない．ただし，BP などの学習アルゴリズムを実装する場合，学習のための特別な回路を階層型 NN を構成する個々のニューロンに付加する必要がある．この追加回路のために故障の発生率が高くなったり，故障検出自体も複雑になる．これに対し，GA による重みトレーニングは NN とは

独立して動作するシステムとして実現できるため，個々のニューロンに学習回路を付加する必要がないという利点を持つ．

2.5 ニューラルネットワークの自律再構成

NN のハードウェア実装の目的の一つとして，組み込みシステムへの応用が挙げられる．組み込みシステムへの応用を想定した NN のハードウェア実装では，より高度な信頼性が求められるため，故障補償能力の有無はより重要になる．また，人間の介入や外部の制御用コンピュータ等の使用が困難な場面での利用も考えられるため，ホストコンピュータ等による制御を必要としない独立したシステムであることや，故障発生時に人間の介入なしに復旧できることが望まれる．そのため，組み込みシステム向け NN の故障補償能力には，外部のホストコンピュータ等を用いずに，自律的に短時間で故障補償を行う能力が要求される．すなわち，故障の発生に対して，必要に応じて自律的にネットワーク構成を変更し，新しいネットワーク構成に最適な結合重みを自動的に獲得するという，二つの機能を併せ持つ自律再構成能力が必要不可欠である．

関連研究として，NN のネットワーク構成や結合重みの自動決定に関する研究が挙げられる．これは NN と GA の融合という観点で行われている研究で，GA を用いたネットワーク構成の自動決定手法や結合重みの自動獲得手法が提案されている [36–41]．特に，北野 [40, 41] が提案しているニューロジェネティック・ラーニング理論 (Neurogenetic Learning: 以下，NGL) は NN の構造決定と結合重みのトレーニングを同時に行うことができる．これらの研究はあくまでもソフトウェア・シミュレーションにより実行される NN において，ネットワーク構造や結合重みを自動的に決定するための手法に関する研究である．本論文で定義しているハードウェア NN システムにおける自律再構成のための手法とは目的が異なる．そのため，ハードウェア NN システムにおける自律再構成のために，どのように GA を利用・応用すべきかといった点は明らかにされていない．GA の利用法やハードウェア NN システムへの組み込み方法などに関して検討する必要がある．

組み込みシステムへの応用を想定したハードウェア・システムにおける再構成に関する研究としては村川らの GRD チップ [42] が挙げられる．GRD チップは NN

アプリケーションのために設計された進化型ハードウェアで，RISC プロセッサと二進木構成の 15 個の DSP で構成されており，実行するアプリケーションに応じて，チップ上にマッピングされた NN のネットワークトポロジと中間層のニューロンの活性化関数を GA によって動的に再構成することができる．この研究における NN の再構成とは，適用するアプリケーションに応じてニューロンの接続や活性化関数の種類といった NN の構成を変化させることであり，本論文で議論している故障補償のための NN の自律再構成とは異なる．しかし，ネットワーク構成や結合重みの獲得に GA を利用することが可能かつ有望な手法であることを示している．

現時点では，ハードウェア NN システムにおける自律再構成手法に関する研究は十分になされておらず，自律再構成能力を有する再構成型 NN のためのハードウェアアーキテクチャに関しても明らかにされていない．

2.6 まとめ

本章では，組み込みシステムへの応用を想定した階層型 NN のハードウェア実装の際に考慮すべき問題について論じた．まず，本論文で扱う階層型 NN と，階層型 NN の基本素子であるニューロンについて概説した．次に，ハードウェア実装の際の問題点として，回路規模と配線領域の増加と，ハードウェア上に発生する故障を挙げ，これらの問題に対する従来研究について述べた．また，ハードウェア NN システムにおける自律再構成手法について考察した．

回路規模増加の問題に対しては，ニューロンの小型化手法を中心に様々な回路規模削減手法が提案されている．また，今後の集積技術の向上によってある程度解消される問題ともいえる．配線領域増加の問題に対しては，パルスニューロンモデルを用いた配線領域の削減に関する研究が行われているが，パルス信号の生成法や演算精度の問題等が指摘されており，現在研究が進められている状況である．配線領域の問題への対策は高集積化が進むにつれ，ますます重要になると考えられる．そこで，本論文ではこの問題への対策に重点を置いた階層型 NN のハードウェアアーキテクチャについて議論する．

ハードウェア上に発生する故障の問題に対しては，冗長手法と重みトレーニン

グ手法の二つのアプローチが存在することを示した．冗長手法は故障の種類によらず効果的な故障補償が可能である．しかし，NN 回路に予備回路を付加することから，故障補償能力と回路規模との間にトレード・オフの関係があり，これが重要な問題となっている．重みトレーニング手法には冗長手法にみられるトレード・オフの問題はないが，結合重み更新のための計算コストの問題と，故障パターンによっては完全に取り除くことのできない故障が存在するという問題がある．また，BP などの学習アルゴリズムを実装する場合には，重み学習のための特別な回路を個々のニューロンに付加する必要があり，それによって故障の発生率が高くなったり，故障検出自体が複雑になるなどの問題も発生する．BP 等の学習アルゴリズムによる重みトレーニングのほかに，GA による結合重み獲得手法に関する研究も行われている．GA による重みトレーニングは NN とは独立して動作するシステムとして実現できるため，個々のニューロンに学習回路を付加する必要がなく，ハードウェア NN システムの重みトレーニング手法として有効であると考えられる．冗長手法と重みトレーニング手法の二つの手法にはそれぞれ利点と欠点があり，より効率の良い故障補償手法に関する研究の重要性が高まっている．

組み込みシステム向けのハードウェア NN には，故障の発生に対して，必要に応じて自律的にネットワーク構成を変更し，その構成に最適な結合重みを自動的に獲得するという自律再構成手法の実現が必要不可欠である．しかし，現時点では，自律再構成手法やそのハードウェアアーキテクチャに関する研究が十分になされているとは言えない．そのため，ハードウェア NN システムにおける自律再構成手法とそれを実現した再構成型 NN のためのハードウェアアーキテクチャを明らかにする必要がある．

以上のことから，組み込みシステムへの応用を想定した階層型 NN のハードウェア・システムの実現には，(1) ハードウェア量，特に配線領域を少なく抑える階層型 NN のハードウェアアーキテクチャと，(2) 故障の回避やネットワーク構成の変更と結合重みの獲得を自律的に行うハードウェア・メカニズム，の二点を同時に考えなければならない．本論文では，ハードウェア実装向き階層型 NN のアーキテクチャとして，配線領域と故障補償容易性を考慮したバス結合型 NN アーキテクチャを提案し，そのアーキテクチャ上での故障補償として，予備ニューロンを用いて単純な操作で故障ニューロンを補償する機能シフト法を提案する．また，結合重

みの獲得手法として GA による NN の重みトレーニングについて検討する．これは階層型 NN の結合重みを獲得するだけでなく，ハードウェア NN システムにおける自律再構成の一部となる．さらに，NN の自律再構成のためのハードウェア・メカニズムとして，冗長手法による故障補償と GA による重みトレーニングを組み合わせた自律再構成手法を提案し，自律再構成機能を有する再構成型 NN システムのハードウェアアーキテクチャを提案する．

第 3 章

故障補償能力を持つニューラルネットワークのハードウェア・システム

3.1 はじめに

階層型ニューラルネットワーク (Neural Network: 以下，NN) は各ニューロンが隣接する層のすべてのニューロンと完全結合される構造になっており，ネットワーク規模が大きくなるにつれて，ネットワークを構成するニューロンの数はもちろん，それらのニューロン同士を接続するための配線領域も膨大になるという問題がある．また，冗長手法による故障補償を実現するためには，予備ニューロンの追加に多くの配線領域を必要とする．このような構造は断線等の故障の増大につながる上に，故障が発生した際の故障補償が非常に困難になる．そのため，配線領域の削減は階層型 NN のハードウェア実装において重要な問題となっている．この問題に対して，パルスニューロンモデルを用いた NN [10,11,13] が注目されている．これは入出力にパルス信号を用いることで一つの入力を 1 本の信号線で構成し，配線領域を削減する手法である．しかし，パルスニューロンモデルにはパルス信号の生成法や演算精度の問題が指摘されており，現在も研究が進められているという状況である．

階層型NN のハードウェア実装に限らず，回路をシリコン上に実装する場合，チップ面積の増大に伴って欠陥や故障の数が指数関数的に増大するという問題がある．現在の技術では，ハードウェア製造時の欠陥や動作中の故障の発生を完全に避け

ることは不可能であり，大規模NNのハードウェア実装には，NNシステム自体に何らかの故障補償能力を持たせることが必要不可欠となっている．これまでに提案された階層型NNの故障補償手法は，NN回路に追加した予備回路を用いて故障を取り除く冗長手法 [17–21] と，結合重みの更新によってNNから故障の影響を取り除く重みトレーニング手法 [22–28] の二種類の手法に分類される．冗長手法はもとのNN回路に予備の回路を追加し，故障が発生した際に故障箇所を予備の回路で置き換えるハードウェア的な故障補償手法で，故障の種類によらず効果的な故障補償が可能である．しかし，追加した予備回路以上の故障の発生には対処できず，故障補償能力と回路規模との間にトレード・オフの関係が存在することになる．これに対し，ニューロンの結合重みを更新することで故障の影響を取り除く重みトレーニング手法では，冗長手法のように回路に冗長性を持たせる必要がない．しかし，ネットワーク規模が大きくなるにつれて結合重み更新のための計算コストが著しく増大することや，故障パターンによっては完全に補償できない場合があるといった問題が指摘されている．また，誤差逆伝搬法 (Back Propagation: 以下，BP) などの学習アルゴリズムを実装する場合には，重み学習のための特別な回路を個々のニューロンに付加する必要があり，それによって故障の発生率が高くなったり，故障検出自体が複雑になるといった問題も発生する．

本章では，階層型NNのハードウェア実装の際の問題点を考慮した階層型NNのハードウェアアーキテクチャと，そのアーキテクチャ上での冗長手法による故障補償について論じる．まず，配線領域と故障補償容易性を考慮したバス接続型NNアーキテクチャを提案し，そのアーキテクチャ上での故障補償手法として，予備ニューロンを用いて単純な操作で故障ニューロンを補償する機能シフト法を提案する．これらの提案手法の回路設計を行い，冗長手法による故障補償能力を持つ階層型NNシステムを構築する．ニューロン数の増加による回路の大規模化に関しては，様々なニューロンの小型化手法が提案されている [8–13]．また，今後の集積技術の発展によりある程度解消される問題であるといえることから，本論文ではニューロンの小型化等による回路規模の削減手法の提案は行わず，配線領域の問題の解決に重点を置く．

第2章で述べたように，階層型NNの中間層は1層以上と定義され，中間層が複数存在する場合もあり得るが，本研究では，より一般的な中間層を1層とした

3 層からなる階層型 NN について論じる．各層のニューロン数をそれぞれ L，M，N としたときの階層型 NN を L-M-N ネットワークと呼び，NN(L, M, N) と表す．

3.2 ハードウェア実装向き階層型ニューラルネットワークのアーキテクチャ

第 2 章で述べたように，階層型 NN は各ニューロンが隣接する層のすべてのニューロンと完全結合される構造になっており (図 2.2)，ネットワークの大規模化に伴い，ネットワークを構成するニューロンの数だけでなく，ニューロン間接続のための配線領域も大幅に増加する．また，予備ニューロンを用いた故障補償を実現するためには，予備ニューロンの追加に多くの配線領域を必要とする．このような構造は断線等の故障の増大につながる上に，ハードウェア実装そのものや，故障が発生した際の故障補償が非常に困難になる．そこで，本論文では各層間の接続方式として図 3.1 に示すようなバス型接続を提案する．バス型接続による階層型 NN をバス結合型 NN と呼ぶ．

バス結合型 NN 回路では，各層に配置されたニューロンは入力選択回路 (図 3.1 の Selector) を通じて接続される．入力層のニューロンの出力は入力選択回路によって一つずつ選択され，中間層に存在するすべてのニューロンに同時に入力される．同様に，中間層のニューロンの出力も入力選択回路を通じて出力層に存在するすべてのニューロンに同時に入力される．そのため，中間層と出力層に存在するニューロンは層内で並列に処理を行うことができる．入力層のニューロンにはそれぞれ外部からのデータを入力する入力ピンが割り当てられる．中間層と出力層に存在するニューロンは入力値と結合重みとの積和演算と活性化関数によるしきい値処理を行う回路である．各ニューロンは結合重みを格納するためのレジスタを持ち，重み獲得回路 (図 3.1 の Weight Controller) を通じて Weight Table から対応する結合重みを獲得する．Weight Table は結合重みを保持するテーブルである．この NN 回路は学習機能を持たないため，外部のコンピュータ上で重み学習のためのプログラムを実行し，その結果得られた結合重みを用いて Weight Table を作成するものとする．

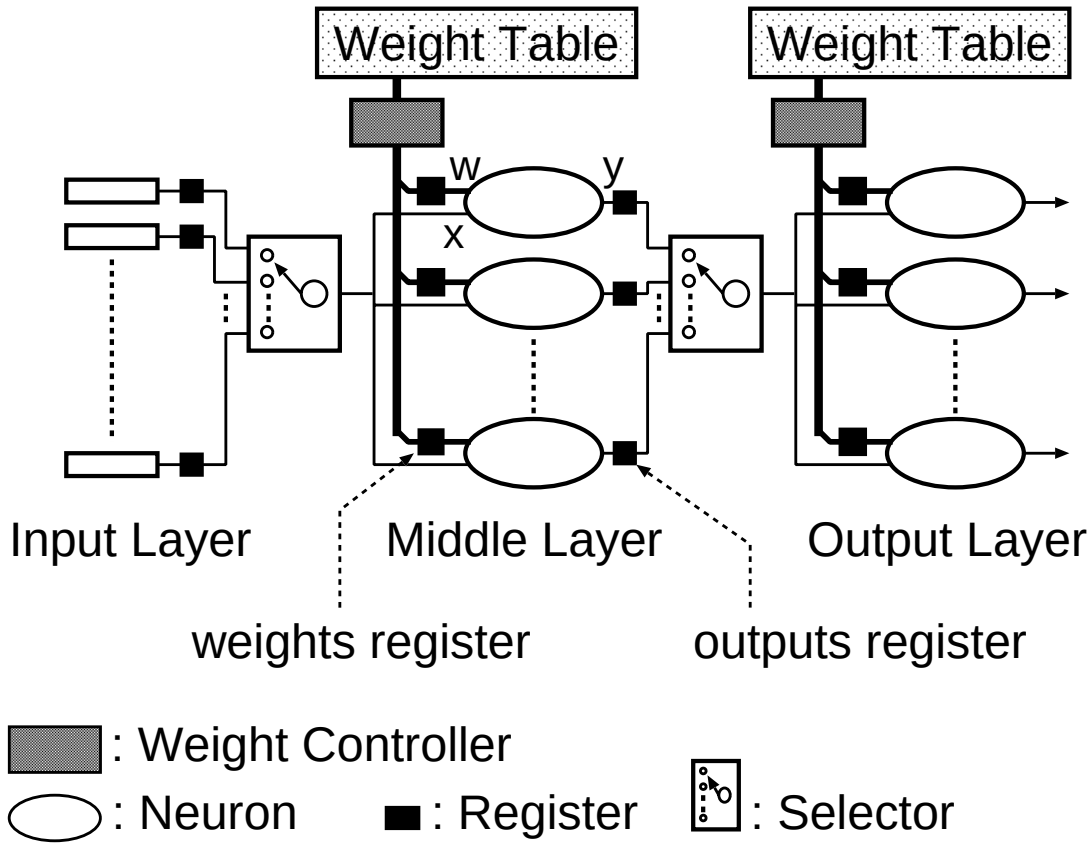


図 3.1: バス結合型 NN の回路構成

バス結合型 NN 回路の処理の流れは以下の通りである．

1. 重み獲得回路を通じて，中間層と出力層のニューロンに対して Weight Table から対応する結合重みを与える．
2. 入力層 (入力ピン) に入力データを与える．
3. 入力層・中間層間に配置された入力選択回路を通じて，クロックごとに選択されたデータを中間層の全ニューロンに同時に送る．中間層の各ニューロンは並列に演算を行う．
4. 中間層ニューロンの演算結果を中間層・出力層間に配置された入力選択回路を通じて，クロックごとに出力層の全ニューロンに送る．出力層の各ニューロンは並列に演算を行う．
5. 出力層の演算結果を出力する．

図 3.1 に示したバス結合型 NN では，各層間に下位層からの入力を選択するための入力選択回路を配置することで，1 ニューロンあたりの結線数を大幅に削減できる．ここで，各層に n 個のニューロンが存在する NN(n, n, n) において，各ニューロンが隣接する層のすべてのニューロンと完全結合される場合とバス型接続の場合の結線数を比較する．完全結合型 NN の場合，層間結線は n^2 となるので，NN 全体の結線数は $O(2n^2)$ となる．これに対して，バス結合型 NN の層間結線数は $2n$ となり，NN 全体で $O(4n)$ となる．また，中間層に 1 個のニューロンを追加する場合，完全結合型 NN では入力層，出力層のすべてのニューロンと追加ニューロンとの接続が必要になるため，$2n$ 本の結線を追加することになる．これに対し，バス結合型 NN では追加ニューロンの入出力分として 2 本追加するだけでよい．つまり，バス結合型 NN は完全結合型 NN に比べ，配線領域を大幅に削減でき，ニューロンの追加による結線数の増加も非常に少ない構造になっている．

次に，バス結合型 NN 回路で用いるニューロン回路について述べる．最も単純なニューロンの構成法は，n 個の入力とそれに対応する結合重みの乗算を行う n 個の乗算回路と，乗算結果の総和を求めるための $n-1$ 個の加算回路，しきい値処理を行う活性化関数用回路で構成する方法である．この場合のニューロンの構成を図 3.2 に示す．これに対し，バス結合型 NN では，ニューロンへの入力値入力選択回路を通じてクロックごとに一つずつ与えられ，n 個の入力を逐次的に処理する．そのため，複数の乗算器を必要とせず，図 3.3 のようにニューロンの構成を簡略化できる．図 3.4 にバス結合型 NN 回路で用いるニューロンの回路構成を示す．このニューロン回路では回路の簡略化のため，活性化関数として 2 値のステップ関数を使用している．ニューロンの出力は 0 か 1 となるので，入力 x とそれに対応する結合重み w の乗算を行う回路として論理積回路を用いている．図 3.4 に示した簡略化されたニューロンの演算精度は低いが，より高精度なニューロンを必要とする場合には，図 3.1 に示した階層型 NN の構成を変更することなく，乗算や活性化関数といったニューロンの演算回路を変更することが可能である．

ここで，図 3.4 に示したニューロンで構成されるバス結合型 NN 回路の処理時間について考察する．NN(L, M, N) において，中間層に存在するニューロンはセレクタを通して順次送られてくる L 個の入力に対して，下記のように入力と結合重みの乗算と，その加算を行う．

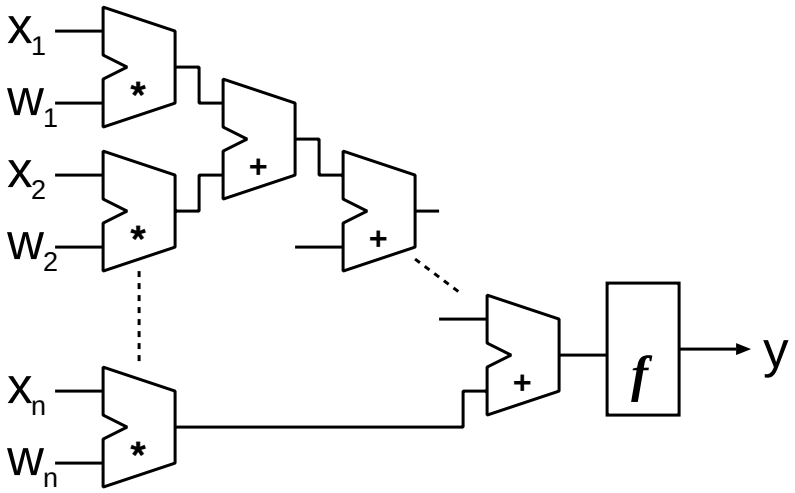


図 3.2: 単純なニューロンの構成

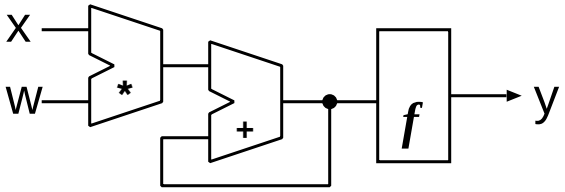


図 3.3: バス型接続のためのニューロンの構成

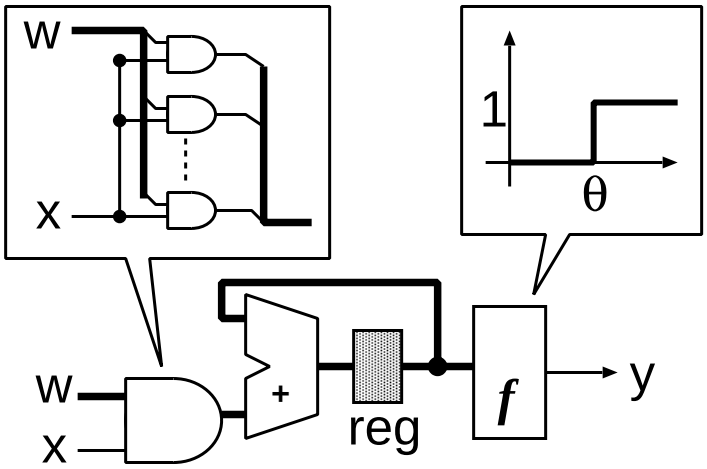


図 3.4: バス結合型 NN で用いるニューロンの回路構成

$sum = 0$
for $i = 0$ to $L - 1$ **do**
  $a = x_i \times w_i$
  $sum = sum + a$
end for

積和演算終了後，活性化関数によるしきい値処理を行う．出力層においても M 個の入力に対して同様の処理が行われるので，一つの入力データセット $(x_0, x_1, \ldots, x_{L-1})$ に対する処理時間 τ_p は，2 変数の乗算 1 回あたりの時間を t_m，2 変数の加算 1 回あたりの時間を t_a，しきい値処理に要する時間を t_{th} とすると，

$$\tau_p = (L + M)(t_m + t_a) + 2t_{th} \tag{3.1}$$

となる．これに対し，一つのニューロンに対する入力が一度に与えられ，それらを同時に処理できる完全結合型 NN 回路における処理時間 τ_c は

$$\tau_c = 2t_m + t_a(\log L + \log M) + 2t_{th} \tag{3.2}$$

となる．活性化関数によるしきい値処理に要する時間はどちらの回路においても等しい．このように，バス結合型 NN 回路では入力を逐次的に処理するため，完全結合型に比べ処理速度の点で多少劣る．しかし，配線領域を大幅に削減でき，ニューロンの追加による配線数の増加も非常に少ないという利点がある．

活性化関数として 2 値のステップ関数を使用したニューロンで構成される階層型 NN は，各ニューロンの出力値が 2 値となることから，NN に論理回路的な動作を期待する場合に用いられる．例えば，論理和や論理積といった論理演算やそれらを組み合わせた演算はステップ関数を使用した階層型 NN で実現可能である．しかし，パターン認識や画像処理といった，より実用的なアプリケーションに適用するためには，活性化関数としてシグモイド関数などの非線型関数を使用する必要がある．シグモイド関数を使用したモデルは現在の NN の主流となっている．このモデルでは，各ニューロンの出力に実数値をとるため，多くの応用が考えられる．しかし，ハードウェア NN システムではシグモイド関数の実装が容易ではなく，回路規模増大の原因となる．単純にシグモイド関数を実装する方法として，シグモイド関数へのマッピングを行うメモリを用いて実現する方法がある．しかし，多くのメモリ領域を使用することから，シグモイド関数の近似などの工夫が必要

である．以上のことから，図3.4に示したニューロンを用いた階層型NNでは，より実用的なアプリケーションの実行は難しいといえる．しかし，回路規模を考慮したニューロンの構成法に関する議論は本章の主な目的ではない．ニューロンの構成法に関してはこれまでに様々な研究が行われている．また，本章で提案したバス結合型NNではニューロンの構成を容易に変更できることから，回路規模を考慮したニューロンの構成法に関する議論はここでは行わない．

3.3 冗長手法による故障補償

3.3.1 故障モデル

階層型NNはニューロンと結合重み，ニューロン間の結合(リンク)から構成される．故障の発生する箇所として，一般に以下に示す3つが考えられる．

- ニューロンの故障
- 結合重みの故障
- リンクの故障

まず，ニューロンの故障について考える．ニューロンは第2章で述べたように一種のしきい値処理素子であり，その故障はニューロンの機能を実現している回路の一部あるいはすべての故障と考えられる．故障パターンには，出力値がある値に固定されるスタック故障と，同じ入力に対して出力値がランダムに変動するランダム故障が考えられる．ランダム故障はNNの学習によって除去することができず，ハードウェア的な故障補償が必要になる．

次に，結合重みとリンクの故障について考える．結合重みの故障もニューロンの故障と同様にスタック故障とランダム故障が考えられる．リンクの故障としては断線故障が挙げられる．これらの故障の発生は正常な入力データや結合重みをニューロンに与えることができず，出力値に影響を与える．その結果，正しい出力が得られないことになる．そのため，結合重みとリンクの故障もニューロンの故障とみなす．

したがって，本研究では故障箇所や故障の種類を区別せず，すべての故障をニューロンに関連した故障として扱う．ここで，故障ニューロンとは，故障がなく正常に動作しているときに得られる出力，すなわち，理論的に求まる出力とは異なる値を出力するニューロンと定義する．

図 3.1 に示したバス結合型 NN は，主にニューロン回路，Weight Table，重み獲得回路 (Weight Controller)，重み格納用レジスタ (weights register)，ニューロンの出力を保持するためのレジスタ (outputs register)，入力選択回路 (Selector) からなる．入力層のニューロンは外部からのデータを入力するための入力ピンであるため，入力層では故障は発生しないものとする．Weight Table はメモリで構成することができ，メモリに対しては様々な故障補償手法 [43] が研究されているため，本論文では Weight Table の故障については扱わない．また，重み獲得回路，入力選択回路は回路量が十分に小さいので，故障しないものとし，これらの故障についても考慮しない．

以上のことから，本論文で扱う故障補償の対象箇所はニューロン回路，重み格納用レジスタ，ニューロンの出力を保持するためのレジスタ，ニューロンの入出力のための結線とする．先に述べたように，これらの故障をニューロンの故障として扱う．ここで定義した故障はニューロンに何らかの故障検出回路を付加することで検出できるものとする．故障検出に関しては，これまでに様々な研究がなされており，一般的な故障検出法として，モジュールを二重化し，二つのモジュールの出力を比較して故障を検出する方法や，自己テスト回路 (BIST: Built In Self Test) を内部に埋め込むことで欠陥を検出する方法などが挙げられる．故障検出手法および回路構成に関する議論は本論文の範疇を越えるため，今回は故障検出回路そのものの設計および実装は行わない．

3.3.2 機能シフト法

図 3.1 に示したバス結合型 NN における故障補償手法として，予備ニューロンを用いて単純な操作で故障ニューロンを補償する機能シフト法を提案する．機能シフト法は冗長手法に分類される故障補償手法である．中間層と出力層にそれぞれ複数個の予備ニューロンを配置し，故障を含んだ階層型 NN から故障ニューロン

を取り除き，実行に必要な階層型NNを再構成する．機能シフト法では，同一層内に存在する故障ニューロンの機能を置き換えるために予備ニューロンを使用する．すなわち，ある層に存在する故障ニューロンの数が同一層内の予備ニューロン数以下である場合，予備ニューロンによる補償が可能である．

予備ニューロンを含む物理的なネットワーク構成 $\mathrm{NN}(L, M, N)$ を

$$NN(L, M, N) = NN(l, m + s, n + t)$$

と表す．ここで，L，M，N はハードウェア実装された入力層，中間層，出力層のニューロン数を表す．l，m，n は各層において実行に必要なニューロン数を表し，s，t はそれぞれ中間層，出力層で予備ニューロンとして使用されるニューロン数を表す．予備ニューロンの回路構成は図 3.4 に示したニューロンの構成と等しく，通常のニューロンと予備ニューロンは明確には区別されない．中間層，出力層において，実際にハードウェア実装されたニューロン数から実行に必要なニューロン数を引いた残りが予備ニューロンとして使用できるニューロン数ということになる．つまり，中間層と出力層における予備ニューロンの数は適用するアプリケーションの要求に応じて変化することになる．

ここで，機能シフト法の動作について述べる前に，機能シフト法で用いる変数を定義する．以下に示す変数は各ニューロンが保持し，機能シフト法を実行する際に使用する．

Definition 1 物理ID：同一層内に存在するすべてのニューロンには物理ID p が割り当てられる．中間層では，$0 \leq p \leq m + s - 1$ となり，出力層では $0 \leq p \leq n + t - 1$ となる．

Definition 2 論理ID：同一層内に存在する正常なニューロンには論理ID $l(p)$ が割り当てられる．中間層では，$0 \leq l(p) \leq m - 1$ であり，出力層では $0 \leq l(p) \leq n - 1$ である．

Definition 3 状態フラグ：状態フラグ $state(p)$ は p 番目のニューロンの状態を表し，以下のように定義される．

$$state(p) = \begin{cases} 0, & (\text{fault-free}) \\ 1, & (\text{faulty}) \end{cases}$$

Definition 4 故障数：ある層の最上位ニューロンから p 番目のニューロンまでに存在する故障ニューロンの数を故障数 $fn(p)$ とし，以下のように定義する．

$$fn(p) = \sum_{i=0}^{p} state(i)$$

$fn(p)$ は同一層内に存在する故障ニューロンの数を表すものではない．

次に機能シフト法の処理手順を示す．機能シフト法では，中間層と出力層においてそれぞれ故障補償のための処理を行う．ここでは中間層での処理手順を示すが，出力層での処理も以下に示す中間層での処理と同様である．

1. 各ニューロンの論理 ID を物理 ID に初期化する．

$$l(p) = p.$$

2. 各ニューロンの状態をチェックし，状態フラグをセットする．

$$state(p) = \begin{cases} 1, & \text{if it is faulty} \\ 0, & \text{if not} \end{cases}$$

3. 状態フラグを用いて各ニューロンの故障数を計算する．

$$fn(p) = \begin{cases} state(p), & \text{if } p = 0 \\ fn(p-1) + state(p), & \text{otherwise} \end{cases}$$

4. 故障数をもとに，各ニューロンの論理 ID を以下のように更新する．

$$l(p) = \begin{cases} p - fn(p), & \text{if } state(p) = 0 \\ 0, & \text{otherwise} \end{cases}$$

5. 更新された論理 ID に基づき，$l(p) < m$ を満たす正常なニューロンに対して Weight Table から対応する結合重みを与える．故障ニューロンには結合重みとしてすべて 0 を与える．

図 3.5 に機能シフト法による故障補償の例を示す．中間層に 5 個のニューロンが実装されており ($M = 5$)，実行に必要なニューロン数が 3 個である ($m = 3$) と仮定する．この場合，予備ニューロンとして使用できるニューロンは 2 個 ($s = 2$) と

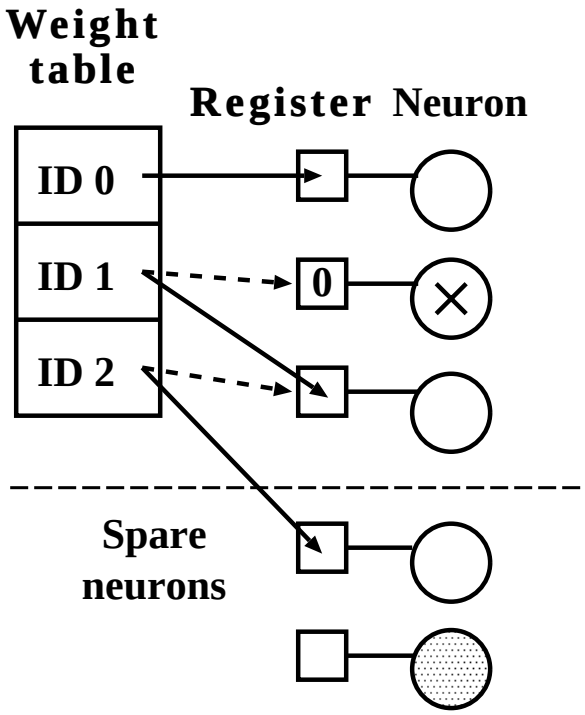


図 3.5: 機能シフト法のによる結合重みの獲得例

表 3.1: 機能シフト法で使用される変数 (図 3.5 の例)

p	$state(p)$	$fn(p)$	l(p)
0	0	0	0
1	1	1	0
2	0	1	1
3	0	1	2
4	0	1	3

いうことになる．物理 ID は最上位ニューロンから順に $p = 0, 1, 2, 3, 4$ である．また，$p = 1$ のニューロンが故障しているものとすると，各ニューロンの状態フラグは $p = 1$ のニューロンのみ 1 である $(state(1) = 1)$．このとき，故障数は $p = 0$ のニューロンのみ 0 となり $(fn(0) = 0)$，それ以外のニューロンの故障数は 1 となる．故障フラグと論理 ID から，実行に使用されるニューロンは $p = 0, 2, 3$ のニューロンとなる．これらのニューロンには Weight Table から対応する結合重みが割り当てられる．$p = 1$ の故障ニューロンには結合重みとして 0 が割り当てられる．表 3.1 に図 3.5 の例の各ニューロンの物理 ID，状態フラグ，故障数，更新後の論理 ID を示す．

3.4 故障補償能力を持つ階層型ニューラルネットワークの設計とその評価

図 3.1 に示したバス結合型 NN をもとに，機能シフト法による故障補償が可能な階層型 NN の回路設計を行った．設計した回路の構成を図 3.6 に示す．図中の `variable controller` は機能シフト法による故障補償を実現するための制御回路であり，NN の実行を開始する前に中間層と出力層の故障数の計算や，論理 ID の更新，各ニューロンへの結合重みの割り当て制御等を行う．入力層・中間層間に配置した入力選択回路 (`selector A`) は入力データを順番に中間層のニューロンへ送る．`selector A` とは異なり，中間層・出力層間に配置した入力選択回路 (`selector B`) では中間層に存在する故障ニューロンの出力を取り除くための制御も行っている．これらの入力選択回路により，ニューロンの出力は一つずつ次層のニューロンに送られるので，入力層の出力 (階層型 NN 回路への入力データ) と中間層の出力を一度レジスタに格納し，順番に次層へ送るものとした．この階層型 NN 回路には学習機能が含まれないため，外部のコンピュータ上で重み学習のためのプログラムを実行し，その結果得られた値を結合重みとして使用する．得られた結合重みは Weight Table として予め回路に与えておく．また，この回路は故障検出回路も実装していないため，ニューロンの状態を表す状態フラグも変数として予め回路に与えておく．

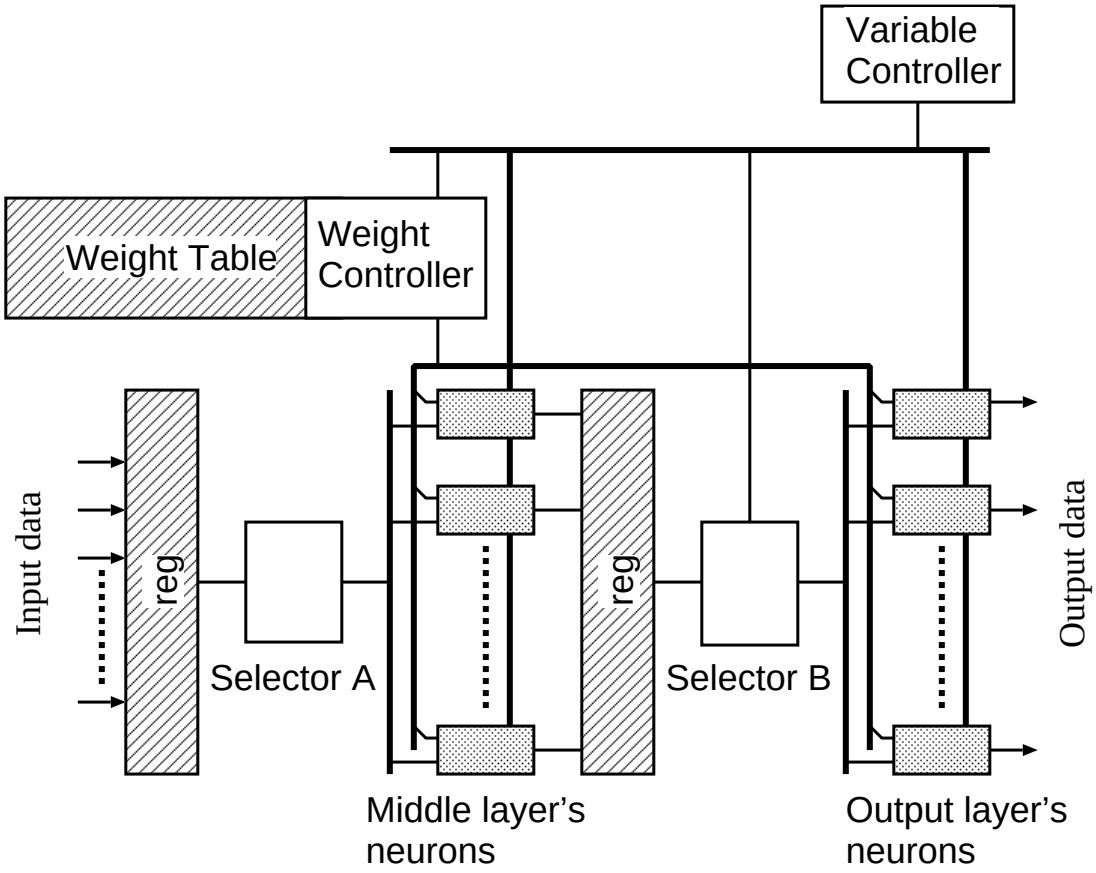


図 3.6: 故障補償能力を持つ階層型 NN の回路構成

表 3.2: 故障補償有/無の回路構成

故障補償機能なし NN 回路	故障補償機能付き NN 回路
ニューロン	ニューロン
入力層・中間層セレクタ	入力層・中間層セレクタ
中間層・出力層セレクタ	**中間層・出力層セレクタ**
Weight Table	Weight Table
—	**機能シフト制御回路**
重み獲得回路	**重み獲得回路**
レジスタ	**レジスタ**

バス結合型 NN 回路に機能シフト法による故障補償能力を追加することによるハードウェア・オーバーヘッドを評価するため，故障補償機能付き階層型 NN と故障補償機能なし階層型 NN の回路規模を比較する．設計した回路は ALTERA 社の回路合成ツールである MAX+Plus II を用いて FPGA 上にマッピングし，その回路規模を見積もった．比較する 2 種類の回路構成を表 3.2 にまとめる．大きな違いは，故障補償機能付き NN 回路に機能シフト制御回路と機能シフト法で用いる変数を保持するためのレジスタが追加されている点である．また，故障補償機能付き NN 回路の中間層・出力層間の入力選択回路は中間層に存在する故障ニューロンの出力を削除する機能を有する．故障補償機能なし NN 回路では，中間層・出力層間の入力選択回路は入力層・中間層間の入力選択回路と同じ構成となっている．さらに，重み獲得回路には，故障ニューロンに結合重みとしてすべて 0 の値を割り当てる機能を追加した．故障補償機能なし NN 回路の重み獲得回路では，Weight Table に格納されている結合重みを順番に各ニューロンに割り当てるだけである．ニューロンの回路構成とネットワークアーキテクチャは故障補償の有無に関わらず等しく，ニューロンは図 3.4 に示した回路を使用し，ネットワークアーキテクチャは図 3.1 に示したバス型接続である．

図 3.7 に設計した 2 つの階層型 NN 回路の回路規模を示す．縦軸は回路規模 (ゲート数) を表し，横軸はネットワークサイズを表す．結合重みは 8 ビットとした．機能シフト法に必要な回路の回路規模のみを評価するために，故障補償機能付き NN

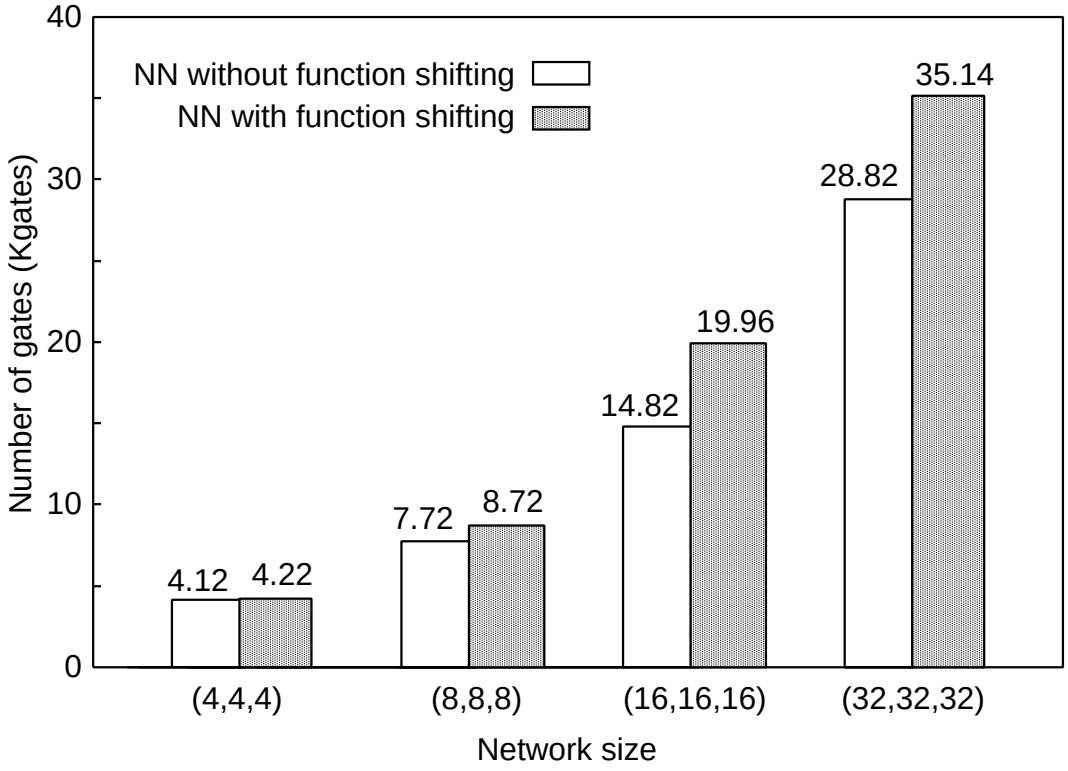


図 3.7: 階層型 NN の回路規模

回路のネットワークサイズは予備ニューロンも含んだサイズとした．つまり，ここで示した故障補償機能付き NN 回路は縮退型であり，中間層，出力層で使用できるニューロン数が予備ニューロンの分だけ少なくなる．$\mathrm{NN}(L, M, N)$ を実装した場合，故障補償機能なし NN 回路は最大で $\mathrm{NN}(L, M, N)$ の実行が可能であるのに対し，故障補償機能付き NN 回路では $\mathrm{NN}(L, M-s, N-t)$ の実行が可能であり，故障補償機能の有無で実行に使用できるニューロン数が異なることになる．

図 3.7 より，故障補償機能付き NN 回路の回路量は故障補償機能なし NN 回路に比べ，約 2%から 35%の増加がみられる．両方の回路に含まれるニューロン数は等しいので，この回路量の増加は機能シフト法のための追加回路によるものと考えられる．表 3.3 に $\mathrm{NN}(4,4,4)$，$\mathrm{NN}(8,8,8)$，$\mathrm{NN}(16,16,16)$，$\mathrm{NN}(32,32,32)$ の場合の故障補償の有無による入力選択回路と機能シフト制御回路の回路規模を示す．故障補償機能付き NN 回路の中間層・出力層間の入力選択回路には論理 ID と状態フラグ，故障数が制御用の信号として入力され，故障ニューロンの出力を取り除くた

表 3.3: セレクタと機能シフト法制御回路の回路規模 (Kgates)

	(4,4,4)		(8,8,8)	
	FT なし	FT あり	FT なし	FT あり
入力層・中間層セレクタ	0.08	0.08	0.14	0.14
中間層・出力層セレクタ	0.08	0.30	0.14	0.68
機能シフト制御回路	—	0.22	—	0.60
	(16,16,16)		(32,32,32)	
	FT なし	FT あり	FT なし	FT あり
入力層/中間層セレクタ	0.30	0.30	0.58	0.58
中間層/出力層セレクタ	0.30	1.66	0.58	2.38
機能シフト制御回路	—	1.76	—	4.52

めの制御を行っている．そのため，入力層・中間層間の入力選択回路よりも回路規模が大きくなっている．また，ネットワーク規模が大きくなると制御用のデータが増えるため，回路規模も増加する．同様に，機能シフト制御回路の回路規模もネットワーク規模が大きくなるにつれて増加する．表 3.3 に示した回路規模のデータは各ユニットをそれぞれ回路合成ツールでマッピングした結果得られた回路規模であり，階層型 NN 回路における各ユニットの実際の回路規模とは厳密には異なる．

図 3.7 より，故障補償機能なし NN 回路，故障補償機能付き NN 回路ともに，ネットワーク規模に比例した回路規模の増加がみられる．故障補償機能付き NN 回路では，ネットワーク規模が倍になると回路規模はほぼ倍になる．ネットワーク構成を $\mathrm{NN}(L, M, N)$ から $\mathrm{NN}(L', M', N')$ に変更した場合の回路の増加量は $\mathrm{NN}(L, M, N)$ の回路規模を c_{nn} とすると

$$c'_{nn} = c_{nn} \frac{L' + M' + N'}{L + M + N} \tag{3.3}$$

となる．ネットワークの大規模化に伴う回路の増加量を推定すると，各層に 128 個

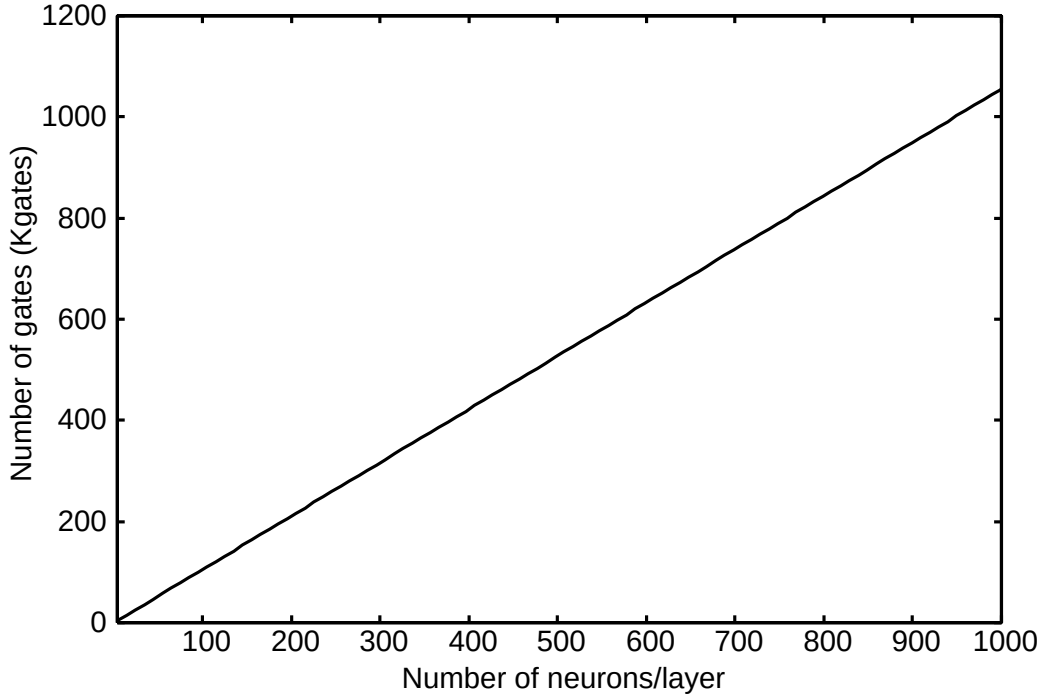


図 3.8: 階層型 NN の回路規模の変化

のニューロンを含む NN$(128, 128, 128)$ の回路規模は約 135K ゲート，各層に 256 個のニューロンを含む NN$(256, 256, 256)$ の回路規模は約 270K ゲートになるものと推定できる．図 3.8 にネットワークの大規模化に伴う回路規模の推移を示す．これは結合重みを 8 ビットとしたときの NN$(4, 4, 4)$ の回路規模を基準に推定した．縦軸は推定回路規模 (ゲート数) を表し，横軸は一層あたりのニューロン数を表す．単純化のため，各層のニューロン数を等しいものとした．つまり，$L = M = N$ とした．図 3.8 より，各層に 1,000 個のニューロンを配置した NN$(1,000, 1,000, 1,000)$ を実装する場合の回路規模は 1M ゲート程度になるものと推定できる．

次に，故障補償機能付き回路において予備ニューロンの増加に伴い，どの程度回路規模が増加するか評価する．ここでは，縮退型 NN 回路ではなく，実行に使用するニューロンのほかに複数個の予備ニューロンを追加した状態を想定する．図 3.9 に NN$(4, 4, 4)$ に予備ニューロンを追加したときの回路規模を示す．一番左の棒グラフは故障補償機能なし NN 回路の回路規模であり，次の棒グラフは図 3.7 に示したものと同じ縮退型の故障補償機能付き NN 回路の回路規模である．x 軸に $(4, 4 + s, 4 + t)$ と示した棒グラフは実行で使用できるニューロンのほかに中間層と

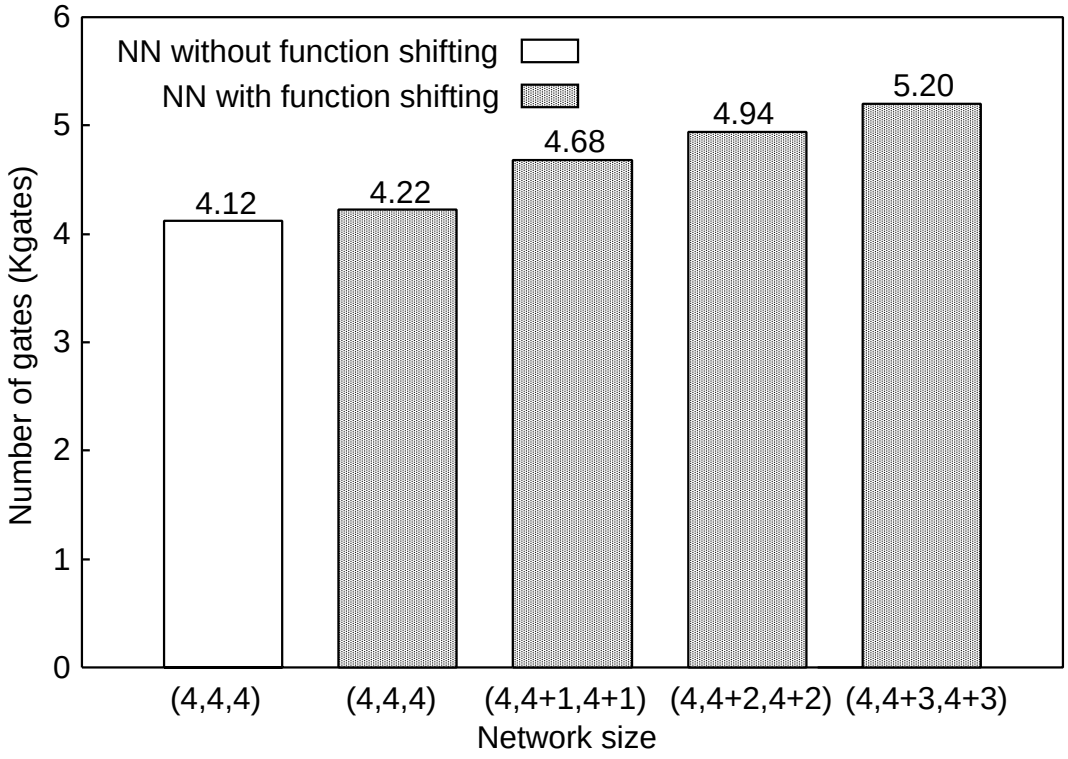


図 3.9: 予備ニューロンの追加によるハードウェア・オーバーヘッド

出力層にそれぞれ s 個，t 個の予備ニューロンを追加した故障補償機能付き NN 回路の回路規模である．これらの回路構成は縮退型故障補償機能付き NN 回路と等しい．図 3.9 から，予備ニューロンの追加に伴う回路の増加量はほぼ追加した予備ニューロン分のみで，機能シフト法を実現するための回路には影響を与えないことが分かる．

以上の結果から，縮退型の故障補償機能付き NN 回路では機能シフト法自体の回路追加のオーバーヘッドは比較的小さいといえる．実行に必要なニューロン以外に予備ニューロンとして使用できるニューロンを追加する場合，つまり，縮退型ではない階層型 NN 回路では追加ニューロン分の回路規模の増加があるため，全体の回路規模と予備ニューロン数とのトレードオフが問題となる．

3.5 故障補償能力を持つ階層型ニューラルネットワーク・システムのハードウェア実装

3.5.1 ハードウェア実装環境

第 3.4 節で述べた故障補償機能付き NN 回路を FPGA 上に実装し，その動作確認を行った．使用した FPGA は ALTERA FLEX シリーズの EPF10K10QC208-4(以下，FPGA1 と表記) と EPF10K130VGC599-3(以下，FPGA2 と表記) の 2 つである．FPGA1 は主に入出力用として使用し，設計した階層型 NN 回路は FPGA2 上に実装する．回路設計・論理合成ツールとして ALTERA 社の MAX+Plus II を使用する．表 3.4 に実装環境を示す．図 3.10 は使用した FPGA 等の外観である．

3.5.2 故障補償機能付き NN 回路の動作検証

第 3.4 節で設計した故障補償機能付き NN 回路を用いて，2 入力排他的論理和 (XOR) 関数を構成し，正しい出力が得られるか検証した．実験に用いる論理的な階層型 NN のネットワーク構成を図 3.11 に示す．各ニューロンの結合重みとしきい値は表 3.5 のように設定し，Weight Table として階層型 NN 回路に組み込む．この値は外部のコンピュータ上で C プログラムによる階層型 NN の重み学習を行った結果得られた値である．FPGA 上に実装した階層型 NN は NN$(4,4,4)$ の縮退型回路で，中間層の $p=3$ のニューロンと出力層の $p=1,2,3$ のニューロンが予備ニューロンとして扱われる．また，故障検出回路を付加することでニューロンの故障を検出できるものと仮定しているので，故障パターンはパラメータとしてあらかじめ階層型 NN 回路に与えておくものとする．

図 3.12 から図 3.15 に中間層および出力層のニューロンに故障を与えた場合の故障補償機能付き NN 回路の実行結果を示す．ここでは，中間層の $p=0$ のニューロンと出力層の $p=0$ のニューロンが故障しているものとした．図中の各信号はそれぞれ以下の通りである．

- x：階層型 NN への入力 (入力層への入力)
- y：階層型 NN からの出力 (出力層の出力)

表 3.4: 実装環境

FPGA ボード (1)	MU200-EA10 (三菱マイコン機器ソフトウェア)
FPGA チップ (1)	EPF10K10QC208-4 (ALTERA, 10Kgate)
FPGA ボード (2)	MEB200-A250 (三菱マイコン機器ソフトウェア)
FPGA チップ (2)	EPF10K130VGC599-3 (ALTERA, 130Kgate)
回路設計・論理合成ツール	MAX+Plus II (ALTERA)

図 3.10: 使用した FPGA ボード

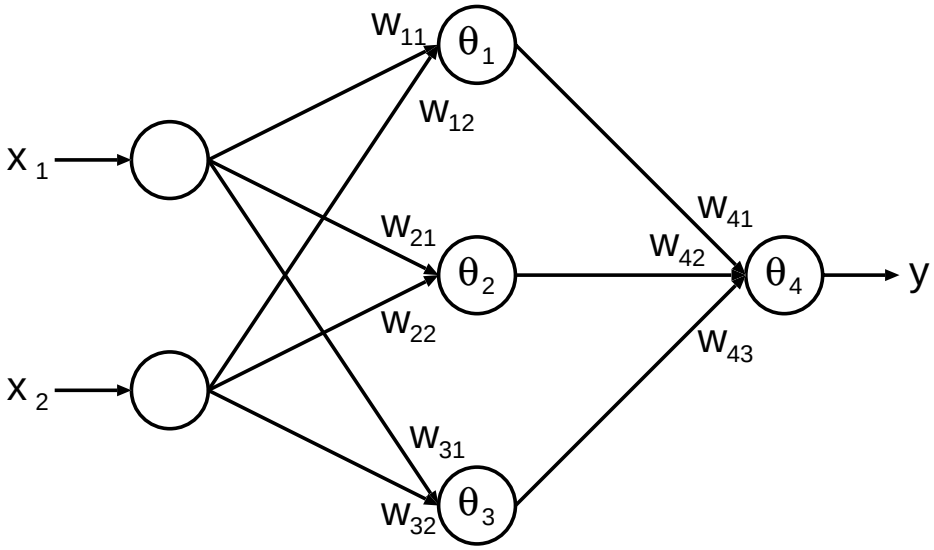


図 3.11: 動作検証に用いる論理的な階層型 NN の構成

表 3.5: 各ニューロンに与える結合重みおよびしきい値

中間層			出力層
$P_{id}=0$	$P_{id}=1$	$P_{id}=2$	$P_{id}=0$
$w_{11}:-2.500$	$w_{21}:+0.875$	$w_{31}:-1.250$	$w_{41}:-2.875$
$w_{12}:-2.500$	$w_{22}:-0.625$	$w_{32}:-1.375$	$w_{42}:-0.625$
			$w_{43}:+2.375$
$\theta_1:-1.750$	$\theta_2:-0.625$	$\theta_3:-1.875$	$\theta_4:+0.625$

- mni：中間層への入力
- oni：出力層への入力

mni，oni はそれぞれ 4 ビットの信号で，各ビットがニューロンの物理 ID に対応している．例えば mni=4 の場合，入力層ニューロンの出力が物理 ID 順に 0, 0, 1, 0 であることを表している．また，出力層の出力 y_p(p は物理 ID を示す) はニューロンの状態を表すビットと演算結果を表すビットの 2 ビットで構成される．

図 3.12 は入力として $(0, 0)$ を与えたときの実行結果である．出力層の $p = 0$ のニューロンが故障しているため，このニューロンの出力を表す信号は $y_0 = 2$ となっている．これはニューロンが故障していることを示す出力である．$p = 0$ のニューロンの代わりに $p = 1$ のニューロンが実行に使用される．この入力に対する期待出力は 0 であり，実際の出力も故障を示す y0 を除いて 0 となっている．図 3.15 に示した入力 $(1, 1)$ に対する結果も同様である．図 3.13 は入力として $(0, 1)$ を与えたときの実行結果である．出力層の $p = 0$ のニューロンの代りに $p = 1$ のニューロンが使用されており，演算の結果，その出力が 1 になっていることが分かる．図 3.14 に示したように，入力として $(1, 0)$ を与えたときもこれと同様の結果が得られている．これらの結果はある入力に対して期待出力と等しい出力結果が得られることを示しており，設計した故障補償機能付き NN 回路は正しく動作していることが分かる．

図 3.12 から図 3.15 に示した実行結果は論理合成ツールによるシミュレーション結果であるが，FPGA 上に実装した故障補償機能付き NN 回路の実行でも，これらのシミュレーション結果と同等の結果が得られることを確認した．また，この回路がクロック周波数 20MHz で動作することを確認した．

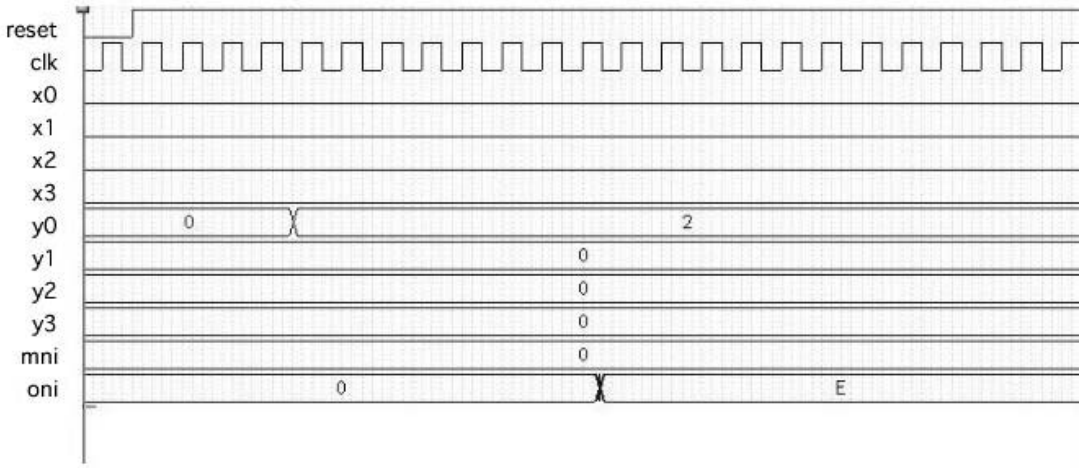


図 3.12: XOR の実行結果 1：入力=(0, 0) の場合

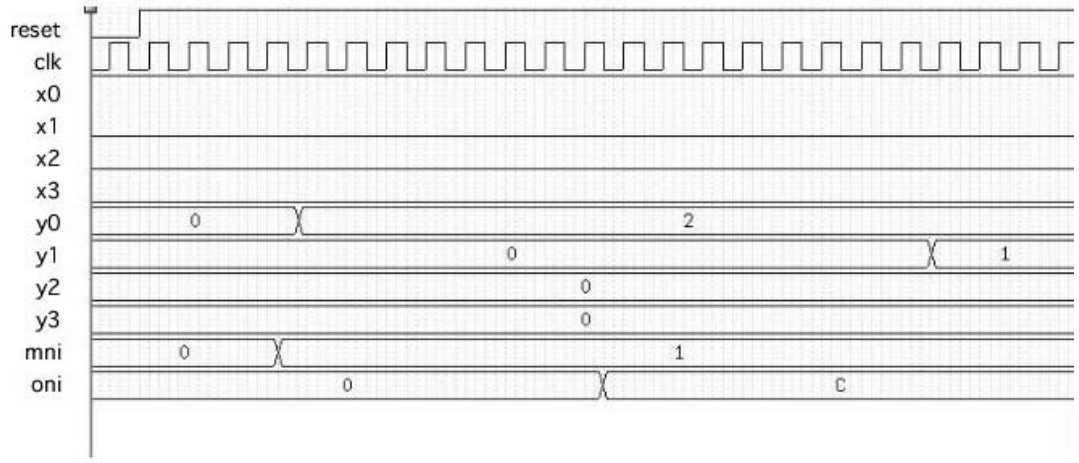


図 3.13: XOR の実行結果 2：入力=(0, 1) の場合

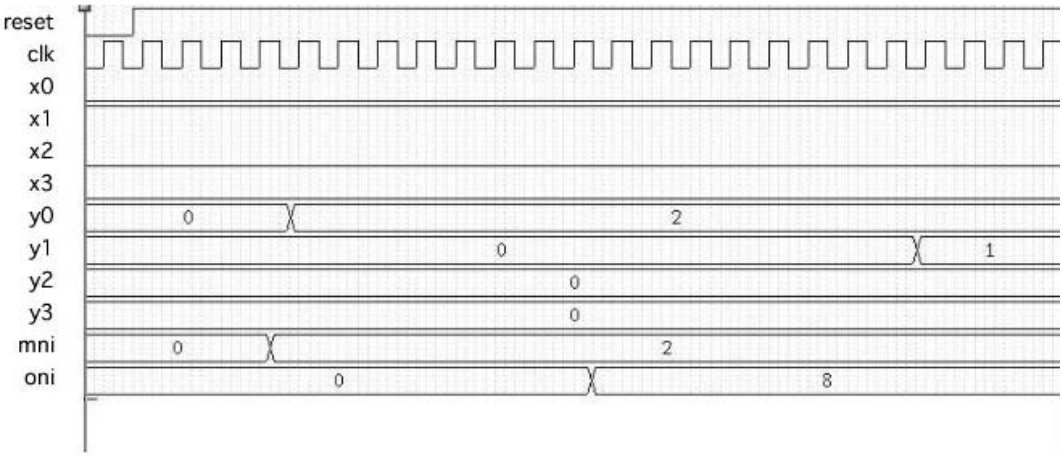


図 3.14: XOR の実行結果 3：入力=(1, 0) の場合

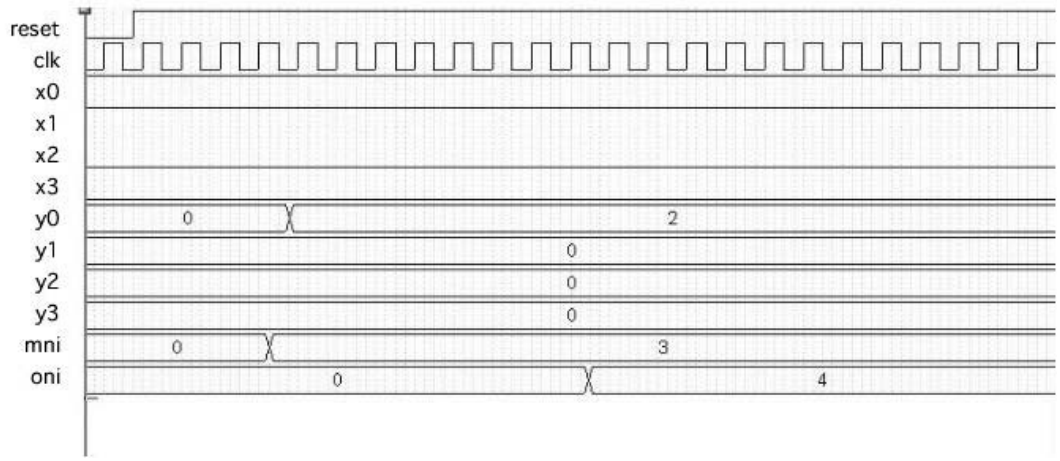


図 3.15: XOR の実行結果 4：入力=(1, 1) の場合

3.6 まとめ

本章では，階層型NNのハードウェア実装の問題点を考慮した階層型NNのハードウェアアーキテクチャと，そのアーキテクチャ上での故障補償手法について論じた．まず，ニューロン間接続のための配線領域と故障補償容易性を考慮した階層型NNのハードウェアアーキテクチャとしてバス結合型NNを提案た．また，そのアーキテクチャ上での故障補償手法として，予備ニューロンを用いて単純な操作で故障ニューロンを補償する機能シフト法を提案した．

バス結合型NNでは，層間に配置された入力選択回路によって，下位層からの入力が一つずつ選択され，層内の全ニューロンに同時に与えられる．入力選択回路を通じて順次与えられる入力は中間層，出力層で逐次的に処理されるため，完全結合型NNと比較すると，処理速度は劣るものの，ニューロン間接続のための配線領域が少なく，ネットワーク規模が大きくなっても配線領域が大幅に増加することがない．また，ニューロンの追加による結線数の増加も非常に少なく，故障補償容易性という観点からも優れており，大規模NNのハードウェア実装に適している．

機能シフト法は冗長手法に分類される故障補償手法であり，予備ニューロンを用いて，単純な操作で故障ニューロンを補償できる．故障ニューロンを物理的にネットワークから切り離すので，同一層内の故障ニューロン数が予備ニューロン数以下である場合には，故障の種類によらず故障補償が可能である．しかし，中間層と出力層にそれぞれ予備ニューロンを配置して故障ニューロンを取り除くため，予備ニューロン数以上の故障が発生した場合には，実行に必要な数のニューロンで構成された階層型NNを得ることができない．これは冗長手法による故障補償の欠点でもある．ただし，故障ニューロンをネットワークから完全に切り離すことで，ニューロン数の異なるネットワークを構成することは可能である．つまり，機能シフト法を用いてネットワーク構成を変更し，故障ニューロンを完全に取り除くことが可能である．この場合，新しいネットワーク構成のための結合重みを再度獲得する必要がある．この問題に対しては，重みトレーニング手法による故障補償を組み合わせることで，さらに効率の良い故障補償手法の実現が可能になる．また，結合重みを獲得する機能を持たせることはハードウェアNNシ

ステムの自律再構成の実現につながる．

機能シフト法による故障補償が可能なバス結合型 NN の回路設計を行い，その回路規模を評価した．その結果，機能シフト法のための回路追加のオーバーヘッドは比較的小さいことが分かった．縮退型ではなく実行に必要なニューロン以外に予備ニューロンを追加する場合には，全体の回路規模と追加する予備ニューロン数とのトレードオフの問題があり，適当な予備ニューロン数を考える必要がある．設計した故障補償機能付き NN 回路は FPGA 上に実装し，クロック周波数 20MHz で動作することを確認した．

第 4章

遺伝的アルゴリズムによるニューラルネットワークの重みトレーニングシステム

4.1 はじめに

ネットワーク構成が確定したニューラルネットワーク (Neural Network: 以下, NN) に対して最適な結合重みを得る手法として，重み学習が知られている．一般には，階層型NNの結合重みの学習アルゴリズムとして誤差逆伝搬法 (Back Propagation: 以下，BP) が用いられる．しかし，BPによる階層型NNの学習ではネットワーク規模が大きくなるにつれて計算コストが著しく増加する，実行に長時間を要するといった問題が指摘されている．そこで，BPの改良アルゴリズムも含め，様々な学習アルゴリズム，結合重み更新アルゴリズムが提案されている [22–25]．また，並列処理による高速化手法も提案されている [26–28]．ただし，アルゴリズムの複雑さなどから，これまでに提案されているアルゴリズムのすべてがハードウェアNNシステムにおける重み学習として利用できるとは限らない．ネットワークの大規模化による計算コストの増加の問題のほかに，学習アルゴリズムや結合重み更新アルゴリズムは局所探索手法であるため，局所最適解に陥りやすく，問題が多峰性を帯びている場合にはうまく機能しないといった欠点も指摘されている．

BP等の学習アルゴリズムによる重み学習のほかに，遺伝的アルゴリズム (Genetic

Algorithm: 以下，GA) による NN の結合重み獲得手法が提案されている [33–35]. NN と GA はともに自然界の生物に関わる原理のモデル化であり，この二つの研究には多くの共通点があることから，NN と GA の融合という観点で様々な研究が行われている．GA による NN の重みトレーニングはその一例である．NN の重みトレーニングを GA で行う理由の一つとして，GA が持つ大域サンプリングの特徴を活かし，効率よく最適解の近傍までトレーニングして局所解に陥る可能性を低減することが挙げられる．

ハードウェア NN システムにおいて結合重みの獲得を実現する場合，BP などの学習アルゴリズムによる結合重みの更新では，教師信号とニューロンの出力の誤差を小さくするように結合重みの調整を繰り返すため，階層型 NN 回路を構成している多数のニューロンに対して重み学習用の複雑な回路の追加が必要になる．個々のニューロンに対して重み学習用回路を追加することは，故障の発生率の上昇や故障検出の複雑化の原因となりうる．これに対し，GA による重みトレーニングは階層型 NN 回路とは別の独立して動作するシステムとして実装でき，個々のニューロンに学習回路を追加する必要がないという利点がある．

本章では，ネットワーク構成の確定した NN に対する結合重みの獲得手法として，GA による NN の重みトレーニングについて検討する．まず，GA の概要と GA による重みトレーニングの概要について述べる．次に，ソフトウェア・シミュレーションによるパターン認識実験を行い，階層型 NN から故障ニューロンをすべて取り除いた縮退ネットワーク上での GA による重みトレーニングを評価する．これは，予備ニューロン数以上の故障ニューロンが存在し，それらを完全に取り除くために実行に必要とされるニューロン数よりも少ない数のニューロンで構成された階層型 NN において，GA による重みトレーニングによって動作可能な結合重みが得られるかどうかを評価する実験である．さらに，NN の重みトレーニングを行うハードウェア GA の回路設計およびハードウェア実装を行い，ハードウェア GA による重みトレーニング・システムを構築する．

第 3 章で述べた故障補償能力を持つ階層型 NN システムでは，機能シフト法を用いてネットワーク構成を変更し，故障ニューロンを完全に取り除くことが可能であった．機能シフト法によりネットワーク構成が変更された場合，新しいネットワーク構成のための結合重みを再度獲得する必要がある．GA による重みトレー

ニングは階層型NNの結合重みを獲得するためだけでなく，ハードウェアNNシステムにおける自律再構成の一部として利用されることになる．

4.2 遺伝的アルゴリズムによるニューラルネットワークの重みトレーニング

4.2.1 遺伝的アルゴリズムの概要

GAはダーウィン的進化論の発想に基づいて生物進化の過程を抽象化したアルゴリズムであり，確率的探索，学習，最適化の一手法として広く研究がなされている．

GAの基本的な操作を以下に示す．

初期化 (initialization) 決められた数の個体をランダムに生成する．ここで生成された個体群を初期集団とする．

適応度評価 (fitness evaluation) 解こうとする問題に応じて定められた評価関数によって各個体を評価し，適応度を求める．適応度とは，個体の優秀さの度合いを示す値である．

選択 (selection) 各個体の適応度を基に，一定の規則に従って親となる個体を選択する．選択手法にはルーレット選択，期待値選択，ランク選択，トーナメント選択，エリート保存選択などがある．

交叉 (crossover) 選択された親個体を交叉させ，子個体を生成する．交叉は個体間のビット列の交換によって実現される．代表的な交叉手法として，1点交叉，多点交叉，一様交叉が挙げられる．

突然変異 (mutation) 一定確率 (突然変異率) によって個体の一部を変化させる．多様性を維持することを目的とした操作であるが，突然変異率が大きすぎるとランダムサーチに陥ってしまう．

選択，交叉，突然変異は遺伝的操作と呼ばれ，これらの操作が終了すると新しい世代の個体集団が生成されたことになる．新たに生成された個体集団内の各個

体は評価関数によって評価され，各々の適応度を得る．このサイクルを 1 世代とし，何世代にもわたり個体集団に対して遺伝的操作，適応度評価を行うことによって優良個体を作り出す．

図 4.1 に GA の基本的なアルゴリズムを示す．まず，初期集団を生成し，その適応度を評価する．生成された集団が終了条件を満たしていれば GA の実行は終了するが，そうでない場合は，遺伝的操作，適応度評価を終了条件を満たすまで繰り返し，新たな世代の集団を生成していく．一般には，終了条件として実行時間 (世代数) や適応度が用いられる．

4.2.2 重みトレーニングへの遺伝的アルゴリズムの適用

GA による NN の重みトレーニングでは，NN の結合重みをどのように 1 次元の染色体にマッピングするかが重要な問題となる．これまでの研究では，図 4.2 に示すように，染色体上に直接結合重みやしきい値を表現する手法を使用している．図中の w は結合重みを，θ はしきい値を表す．結合重みの表現方法には実値表現と二値表現があり，実値表現ではそれぞれの結合重み，しきい値を実数値で表現する．二値表現では，結合重み，しきい値はビット列で表現される．例えば，8 ビットを -127 から $+127$ の範囲を持つ一つの数値として解釈する．各個体はそれぞれ異なる結合重みを持つ NN を表す．

GA を用いて NN の重みトレーニングを行う際の適応度評価では，まず，各個体が表現する結合重みをネットワークにマッピングし，NN を構成する．その NN に対し，トレーニングパターンを与え，NN を実行する．そのときの実際の出力とトレーニングパターンとして与えられた期待出力との誤差 (自乗和誤差) 等を計算し，得られた値を適応度として個体を評価する．遺伝的操作の適用，適応度評価を繰り返すことでネットワークの出力誤差は小さくなり，NN の重みがトレーニングされることになる．

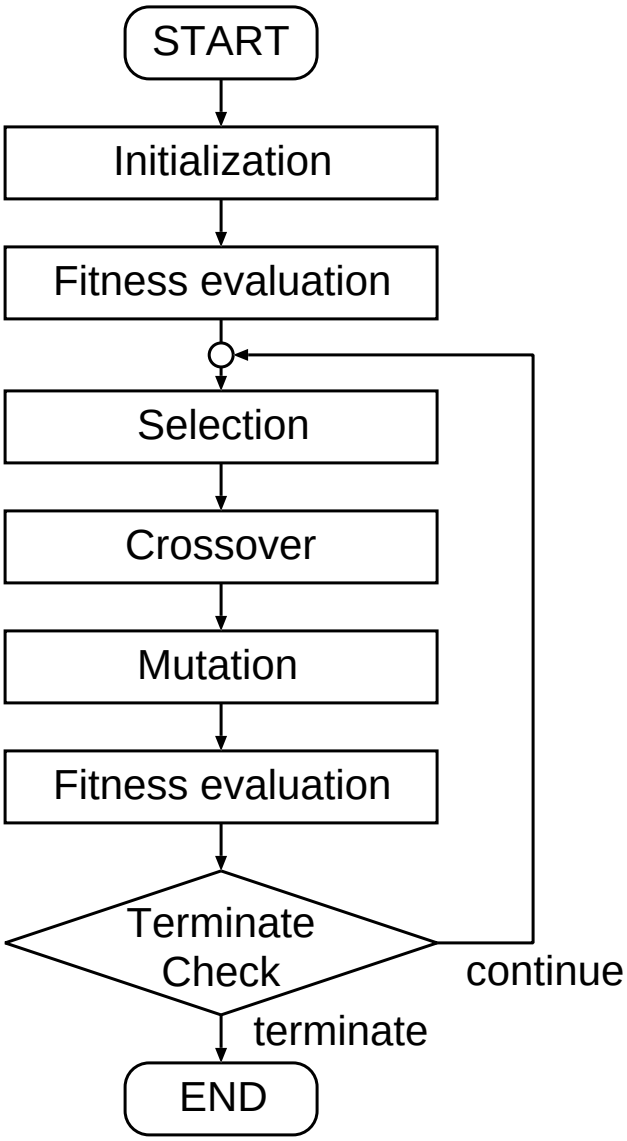


図 4.1: 基本的な GA アルゴリズム

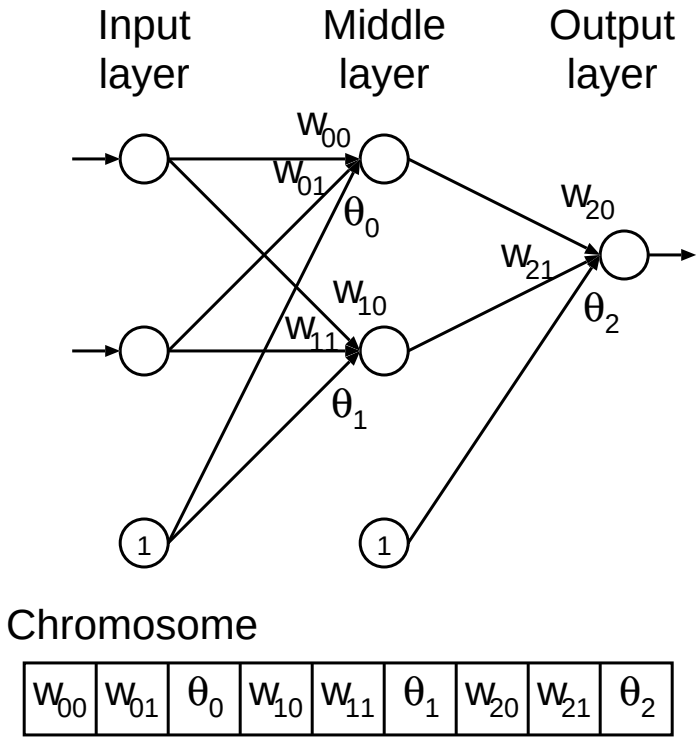


図 4.2: 結合重みのマッピング方法

4.3 ソフトウェア・シミュレーションによる重みトレーニングの評価

GA を用いた NN の重みトレーニングが可能なことはこれまでの研究で示されている．ここでは，階層型 NN に予備ニューロン数以上の故障が発生し，それらを完全に取り除くためにネットワーク構成が変更された状態を想定する．そして，中間層ニューロンの数が実行に必要なニューロン数よりも少なくなった状態でも，GA による重みトレーニングによって正常に動作するための結合重みが得られるか調べる．

縮退ネットワーク上での GA を用いた階層型 NN の重みトレーニングを評価するため，ソフトウェア・シミュレーションにより簡単なパターン認識実験を行った．適用する問題は図 4.3 に示す 4 つのパターンの認識である．ある入力パターンに対して，それに対応する出力ニューロンのみが 1 を出力し，それ以外の出力ニューロンが 0 を出力するとき，この階層型 NN は 4 パターンの認識に成功したものとす

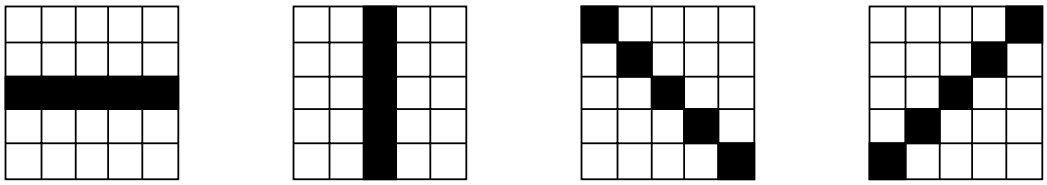


図 4.3: トレーニングパターン

る．このようなパターン認識を行う階層型 NN の結合重みを GA により獲得する．入力パターンのパターンサイズは 5×5 である．パターンサイズ，パターン数から，入力層には 25 個，出力層には 4 個のニューロンが必要になる．中間層ニューロンの個数は実験等により最適に設定すべきであるが，ここでは簡単に 15 個とする．つまり，4 つのパターンの認識に必要なネットワーク構成を $\mathrm{NN}(25, 15, 4)$ とする．実験では，故障の発生により使用できる中間層ニューロンの個数が減少したものと仮定する．実験に用いる NN のネットワーク構成は $\mathrm{NN}(25, M, 4)$ とする．ここで，$M \leq 15$ である．

実験に使用した GA パラメータ等は以下の通りである．

- 個体数：10，20，30，40，50
- 突然変異率：0.001
- 選択方式：ルーレット選択
- 交叉方式：一様交叉

評価関数には平均二乗誤差を使用する．出力層ニューロン数を n とし，トレーニングセットを $\{x^{pt}, t^{pt} | pt = 1, \ldots, P\}$ とする．ここで，P はパターン数，x^{pt}，t^{pt} はパターン pt に対する入力セットと期待出力セットである．x^{pt} をパターン pt を入力したときの実際の出力セットとすると，パターン pt における二乗誤差は

$$E_p = \sum_{i=1}^{n} (t_i^{pt} - z_i^{pt})^2. \tag{4.1}$$

と表され，全パターンに対する二乗誤差は

$$E_a = \frac{1}{P}\sum_{pt=1}^{P} E_p. \tag{4.2}$$

と表すことができる．式 4.2 を GA の評価関数とし，E_a を各個体の適応度とする．この実験では，適応度が小さいものほど良い個体ということになる．

図 4.4，4.5 に，GA の実行によって得られた最良個体 (階層型 NN の結合重み) を階層型 NN にマッピングし，パターン認識を行った結果を示す．図 4.4 は図 4.3 に示した 4 つのパターンの認識を行ったときの平均正答率である．中間層のニューロン数が 5 個にまで減少しても個体数によらず 100%の正答率が得られ，$M = 4$ の場合，個体数が 40 以上のときに正答率が 100%となる．これは GA による重みトレーニングで適切な結合重みが得られていることを示す．また，中間層のニューロン数が 3 個以下に減少すると適切な結合重みが得られず，正しく認識できなくなる．図 4.5 はエラーを含むパターンの認識を行ったときの平均正答率である．図 4.3 の 4 つのパターンにランダムに 1 ビットのエラーを与えた 48 パターンを使用し，入力されたパターンがもとの 4 つのパターンのうちのどれかを判定する．正答率にばらつきがあるものの，個体数が 40 以上で $M \geq 6$ のとき，90% 以上の正答率が得られている．

次に，GA 実行時の適応度の推移を示す．図 4.6 から図 4.9 は中間層のニューロン数をそれぞれ $M = 15, 10, 5, 2$ としたときの各世代における最良個体の適応度の推移を示している．トレーニングパターンには図 4.3 に示した 4 つのパターンを使用した．

図 4.6 は故障によるネットワークの縮退が起こっていない状態での GA による重みトレーニングの様子である．この場合，個体数によらず約 1,000 世代で適応度がほぼ 0 になる．図 4.7 は故障により中間層のニューロンが実行に必要な個数よりも 5 個少なくなった場合であり，$M = 15$ と同様に，約 1,000 世代で適応度がほぼ 0 になる．ただし，個体数が 10 のときには解の収束に若干時間がかかる．適応度がほぼ 0 であるということは，実際の出力と期待出力との間に誤差がほとんどなく，GA による重みトレーニングによって適切な結合重みが得られたことを示す．図 4.4 からも $M \geq 10$ のときには正答率が 100%となり，適切な結合重みが得られていることが分かる．$M = 5$ では，10,000 世代の実行終了時でも $M = 15, 10$ の場

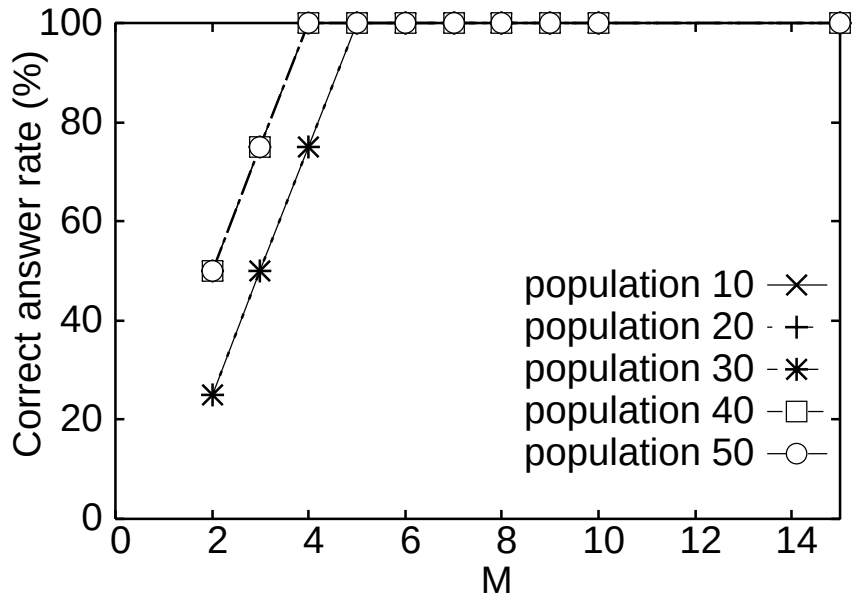


図 4.4: 4 パターンの認識実験

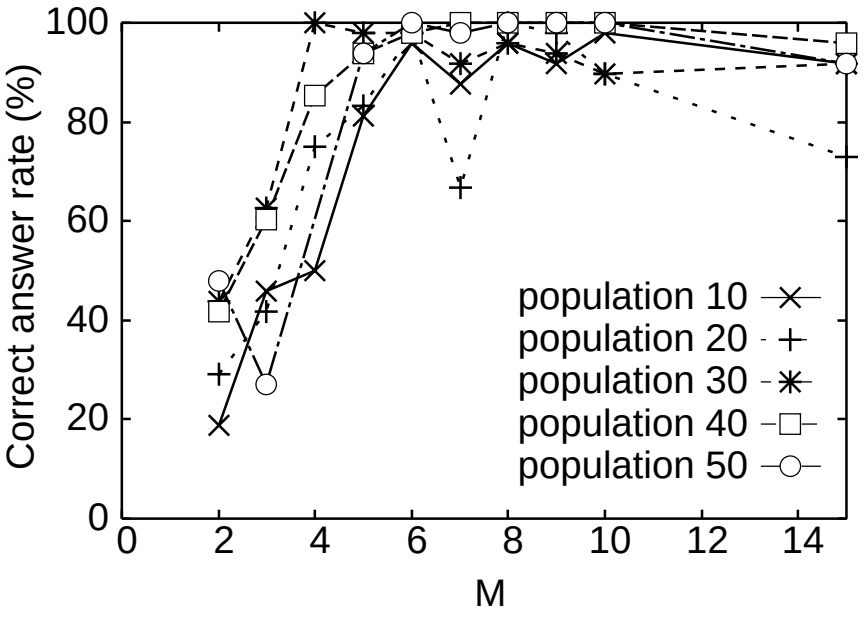


図 4.5: エラーを含む 48 パターンの認識実験

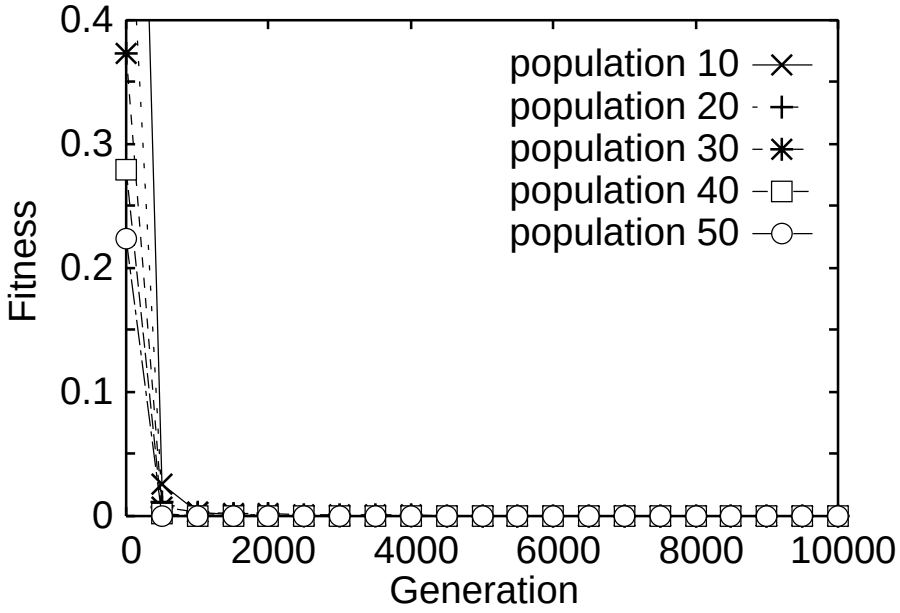


図 4.6: 適応度の推移 ($M = 15$)

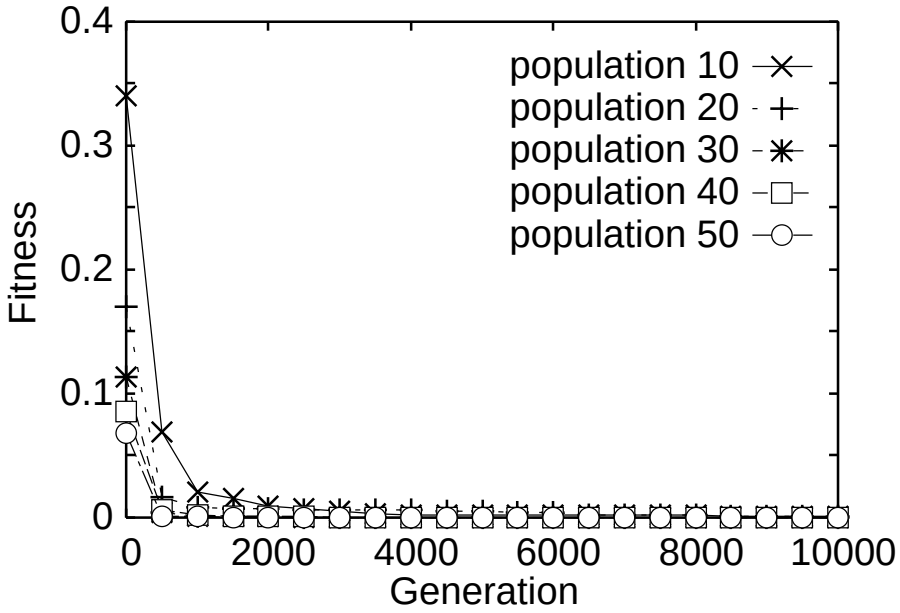


図 4.7: 適応度の推移 ($M = 10$)

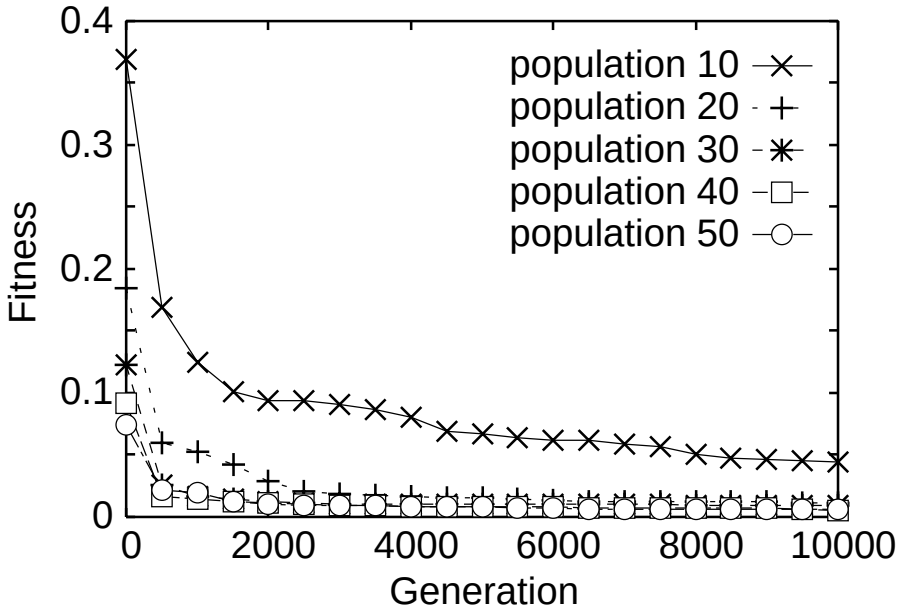


図 4.8: 適応度の推移 ($M = 5$)

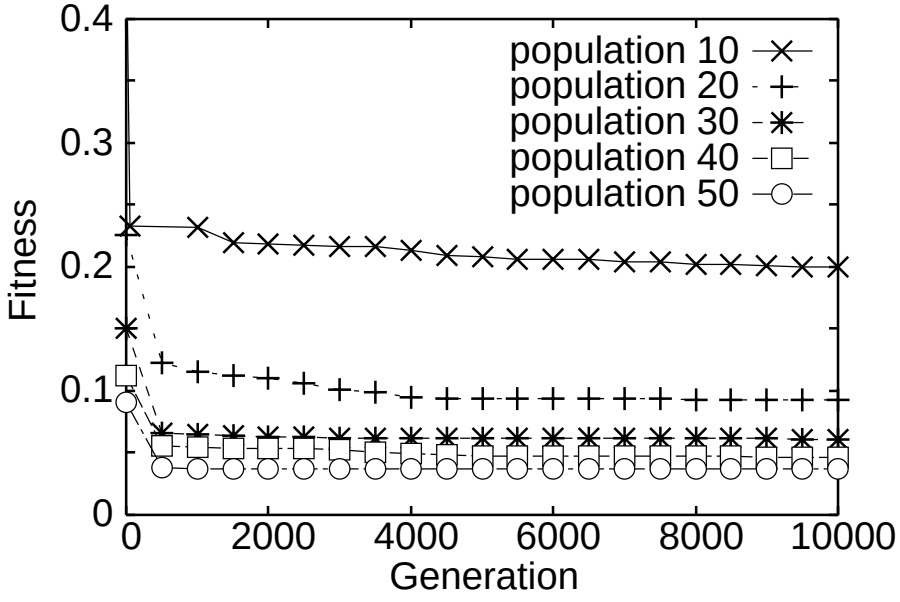


図 4.9: 適応度の推移 ($M = 2$)

合に比べて適応度が大きく，個体数が 10 の場合には約 0.05 となっている．$M = 2$ になると個体数が 50 の場合でも 10,000 世代で 0.03 程度の適応度しか得られない．図 4.4 からも $M = 2$ のときには適切な結合重みが得られず，パターンの認識に失敗することが分かる．

4.4 遺伝的アルゴリズムのハードウェア化とその評価

4.4.1 遺伝的アルゴリズムのハードウェア化

図 4.1 に示した GA アルゴリズムのハードウェア化を行う．GA は適用する問題によって適した選択，交叉の方法があり，また各個体の適応度を評価する評価関数が異なる．そこで，ユニットを交換することで様々な問題に適用できるように GA アルゴリズムの各ステップをそれぞれ別々のユニットとして設計する．

まず，基本的な GA の動作を実現する回路として，One-Max 問題を解くための GA 回路の設計を行った．次に，One-Max 問題を解くための GA 回路をもとに，NN の重みトレーニングを行う GA 回路の設計を行った．

各回路の構成や動作，回路規模の評価等に関しては次節以降で述べる．

4.4.2 One-Max 問題のための GA 回路とその評価

基本的な GA の動作を実現する回路として，One-Max 問題を解くための GA 回路を設計する．One-Max 問題とは，個体を構成するすべてのビットが”1”になったときを最適解とするという問題である．

図 4.10 に One-Max 問題を解くための GA 回路の構成を示す．GA 回路を構成する各ユニットは下記の通りである．

- `RNG`：乱数生成器
  初期集団の生成や遺伝的操作で用いられる乱数を生成する．8 ビットの LFSR (Linear Feedback Shift Register) を使用している．

- `controller`：制御回路
  世代数のカウント，実行終了等の制御を行う．

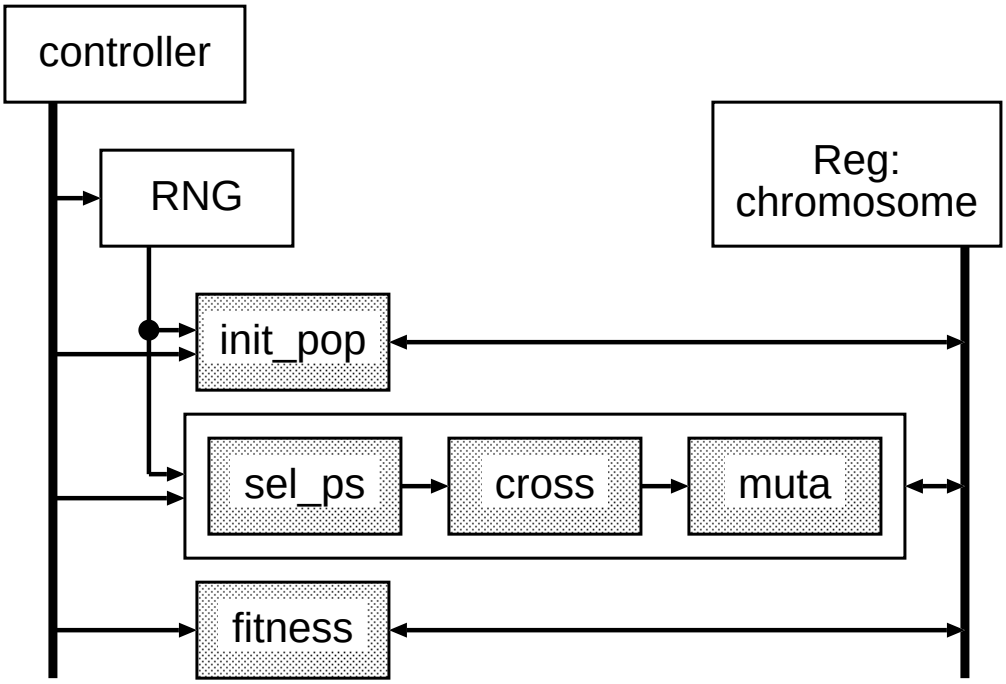


図 4.10: One-Max 問題を解くための GA 回路の構成

- `init_pop`：初期集団生成回路
  `RNG` ユニットにより得られる乱数を用いて初期集団を生成する．

- `sel_ps`：選択回路
  適応度に応じて，交叉の対象となる 2 つの親個体を選択する．ここではルーレット選択に基づき，親個体の選択を行うものとする．

- `cross`：交叉回路
  選択された 2 個体に対して交叉を行い，2 つの子個体を生成する．ここでは一点交叉を行うものとする．

- `muta`：突然変異回路
  新たに生成された 2 個体に対して突然変異を適用し，次世代集団を生成する．

- `fitness`：適応度評価回路
  集団内の各個体の適応度を評価する．ここでは個体のビットの総和を計算する．

初期集団の生成はランダムに行うものとし，選択にはルーレット選択を用いる．交叉はOne-Max問題を解く際に最も一般的だと思われる一点交叉を用いる．突然変異はすべてのビットを突然変異率を用いて判定し，ビット反転を行う．One-Max問題ではすべての遺伝子が1となる個体が最適解となるので，適応度はビットの総和とした．終了条件は最大世代数とし，1,000世代までの処理が終わったところで回路の実行を終える．最大世代数はパラメータとして回路に組み込んでおり，今回は1,000世代としたが，これは変更可能である．

ここで，One-Max問題を解くためのGA回路の動作について述べる．GA回路の動作が開始されると，はじめに`init_pop`ユニットにおいてランダムに初期集団が生成され，各個体の適応度が`fitness`ユニットで評価される．次に生成された各個体に対して`sel_ps`ユニット，`cross`ユニット，`muta`ユニットにおいて選択，交叉，突然変異が適用され，次世代集団を生成する．`sel_ps`ユニットでは適応度に応じて2個体を選択し，`cross`ユニットに送る．`cross`ユニットでは`sel_ps`ユニットから送られた2個体に対して交叉を行い，新たに2個体を生成する．交叉後の2個体は`muta`ユニットに送られ，決められた突然変異率にしたがって突然変異を適用する．この3つのユニットは決められた個体数を満たすまでパイプライン式に動作し，次世代集団を生成する．次世代集団が生成されると，`fitness`ユニットにおいて生成された次世代集団の各個体の評価が行われる．`init_pop`ユニットはGA回路の実行が開始されたときに一度だけ動作し，`sel_ps`ユニットから`fitness`ユニットまでの実行は終了条件が満たされるまで繰り返し行われる．終了判定は`controller`ユニットで行われる．

次に，One-Max問題を解くためのGA回路の回路規模を評価する．図4.11にGA回路の回路規模を示す．横軸は集団サイズを表し，縦軸は回路規模を表す．図中のCLは個体長を表し，$CL = 4$は各個体が4ビットであることを示している．個体数が倍になると回路規模もほぼ倍になるが，これは個体を格納しているレジスタの量が倍になるからと考えられる．CLが倍になるときも同様の理由で回路規模が倍になる．

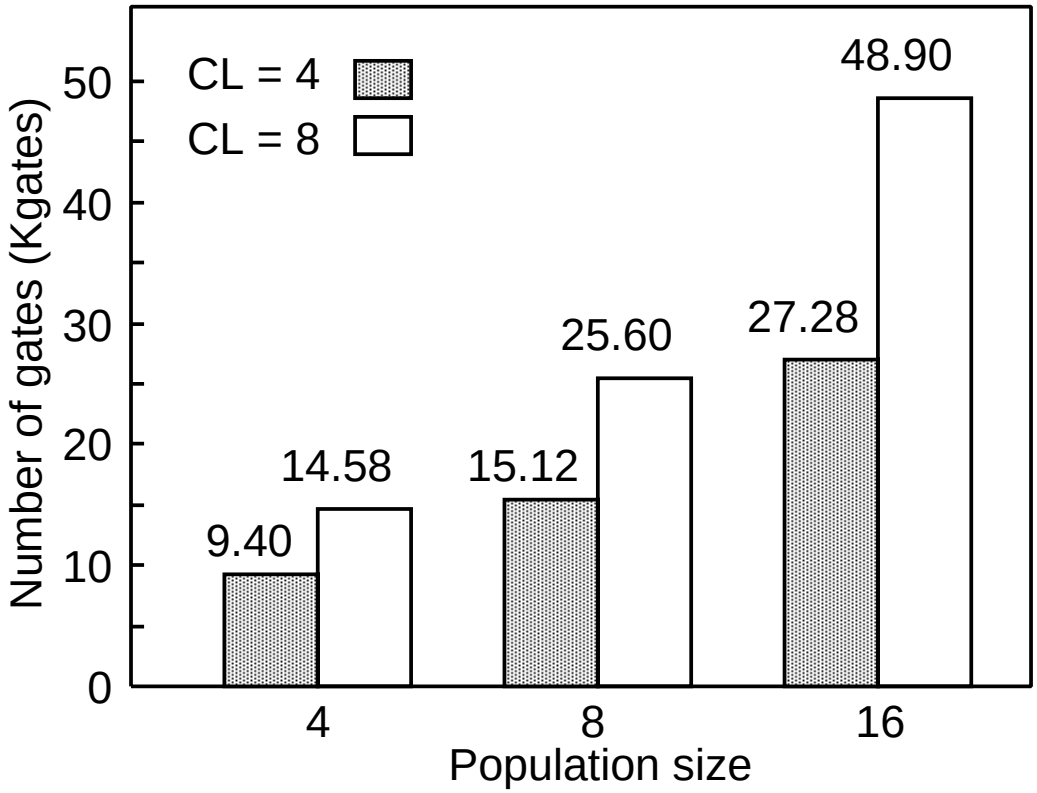


図 4.11: One-Max 問題のための GA 回路の回路規模

4.4.3 NN の重みトレーニングのための GA 回路とその評価

One-Max 問題を解くための GA 回路をもとに，NN の重みトレーニングを行う GA 回路を設計する．基本的な回路構成は図 4.10 に示した One-Max 問題のための GA 回路と同じであるが，交叉ユニットと適応度評価ユニットに変更を加え，最良個体選択用回路を追加した．

NN の重みトレーニングのための個体は個体長が長くなるため，一点交叉ではなく一様交叉を使用する．そこで，`cross` ユニットを一点交叉ユニットから一様交叉ユニットに変更した．一点交叉では，交叉点を一つ決めて，その前後で二つの親個体の遺伝子を入れ替える．これに対し，一様交叉では，交叉時にマスクをかけて，それによってどちらの親の遺伝子を受け継ぐかを決定する．まず，二つの親個体とマスクを設定する．マスクのビットが 0 の時は，子個体 1 に親個体 1 の遺伝子をコピーし，マスクのビットが 1 の時は，親個体 2 の遺伝子をコピーする．子個体 2 に対してはこれと逆の操作を行う．

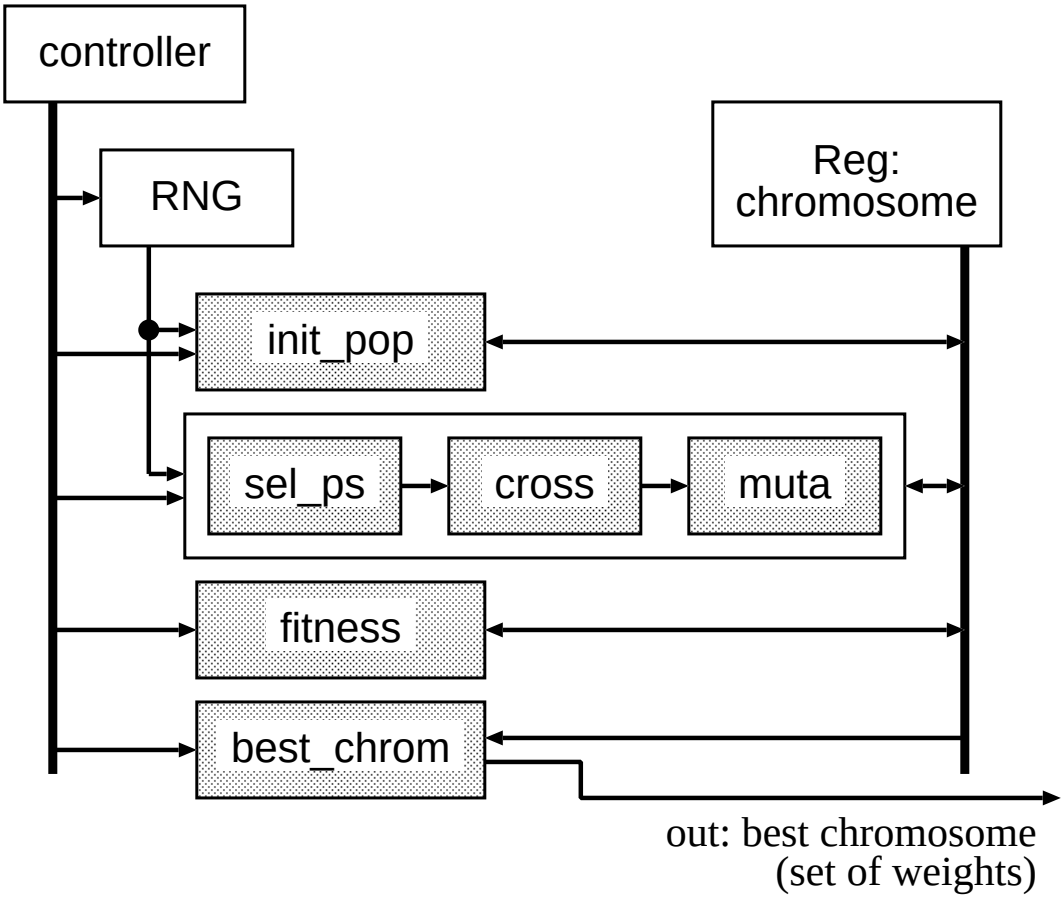


図 4.12: 重みトレーニングのための GA 回路の構成

適応度評価では，各個体が表現する結合重み，しきい値を階層型 NN にマッピングし，トレーニングパターンを与えたときの出力誤差を適応度として使用する．そのため，`fitness` ユニットは階層型 NN 回路と出力誤差計算回路で構成される．`fitness` ユニット内の階層型 NN 回路には第 3 章で述べた故障補償機能なし NN 回路を使用した．

GA 回路による重みトレーニングによって得られた NN の結合重みは階層型 NN の実行で使用される．そのため，GA の実行終了後，最終世代集団内の最良個体を選び出す必要がある．そこで，最終世代集団の中で最も適応度の高い個体を選択する `best_chrom` ユニットを付加した．

NN の重みトレーニングのための GA 回路を構成するユニットは下記の通りである．また，図 4.12 に NN の重みトレーニングを行う GA 回路の回路構成を示す．

- RNG：乱数生成器
  初期集団の生成や遺伝的操作で用いられる乱数を生成する．8 ビットの LFSR (Linear Feedback Shift Register) を使用している．

- controller：制御回路
  世代数のカウント，実行終了等の制御を行う．One-Max 問題のための GA 回路に最終世代での最良個体選択のための制御を追加している．

- init_pop：初期集団生成回路
  RNG ユニットにより得られる乱数を用いて初期集団を生成する．

- sel_ps：選択回路
  適応度に応じて，交叉の対象となる 2 つの親個体を選択する．ここではルーレット選択に基づき，親個体の選択を行うものとする．また，一様交叉のためのマスクも生成する．

- cross：交叉回路
  選択された 2 個体に対して交叉を行い，2 つの子個体を生成する．ここでは一様交叉を行うものとする．

- muta：突然変異回路
  新たに生成された 2 個体に対して突然変異を適用し，次世代集団を生成する．

- fitness：適応度評価回路
  集団内の各個体の適応度を評価する．NN 回路を含み，個体が表現する結合重みをマッピングし，NN を実行する．NN の実行の結果得られた出力と期待出力との誤差を計算し，適応度とする．

- best_chrom：最良個体選択回路
  最終世代における最良個体を選び出し，出力する．

NN の重みトレーニングのための GA において，各個体は階層型 NN を構成するすべてのニューロンの結合重みとしきい値を 1 次元配列にマッピングした構成になる．図 4.13 に個体の表現例を示す．これは結合重みを 8 ビットとした NN$(2, 2, 1)$ の重みトレーニングを行う場合に使用される個体を表している．NN(L, M, N) の重みト

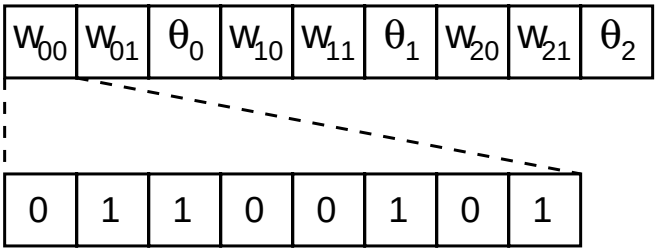


図 4.13: 個体の表現例：NN(2, 2, 1)，結合重み 8 ビットの場合

レーニングを行う場合，各個体のサイズ (個体長) は，$w\{(L+1)\times M+(M+1)\times N\}$ ビットとなる．$L+1$，$M+1$ となっているのは，各ニューロンのしきい値が含まれるためである．ここで，w は結合重みのビット幅である．例えば，図 4.13 に示したような，結合重みを 8 ビットとした NN$(2,2,1)$ の重みトレーニングでは，個体長は 72 ビットとなる．

NN の重みトレーニングのための GA 回路では，`init_pop` ユニットによる初期集団の生成，`sel_ps` ユニット，`cross` ユニット，`muta` ユニットによる遺伝的操作，`fitness` ユニットによる適応度評価は One-Max 問題のための GA 回路と同様に動作する．`controller` により終了判定がなされると，`best_chromo` ユニットによって最良個体の選択が行われ，その適応度が出力される．

NN$(2,2,1)$ のための重みトレーニングを行う GA 回路の回路規模を図 4.15 に示す．横軸は集団サイズを表し，縦軸は回路規模を表す．図中の CL は個体長を表している．個体数が倍になると回路規模もほぼ倍になる．これは個体数が倍になると個体を格納しているレジスタの量が倍になることが原因と考えられる．個体長が倍になるときも同様の理由で回路規模が倍になる．これは図 4.11 に示した One-Max 問題のための GA 回路の回路規模と同様の結果である．

表 4.1 に重みトレーニングのための GA 回路を構成しているユニットの回路規模を示す．ただし，表 4.1 に示した回路規模のデータは各ユニットをそれぞれ回路合成ツールでマッピングした結果得られた回路規模であり，重みトレーニングのための GA 回路における各ユニットの実際の回路規模とは厳密には異なる．

回路規模の評価より，重みトレーニングの対象となる階層型 NN のネットワーク構成を NN(L,M,N) から NN(L',M',N') に変更した場合の回路規模は

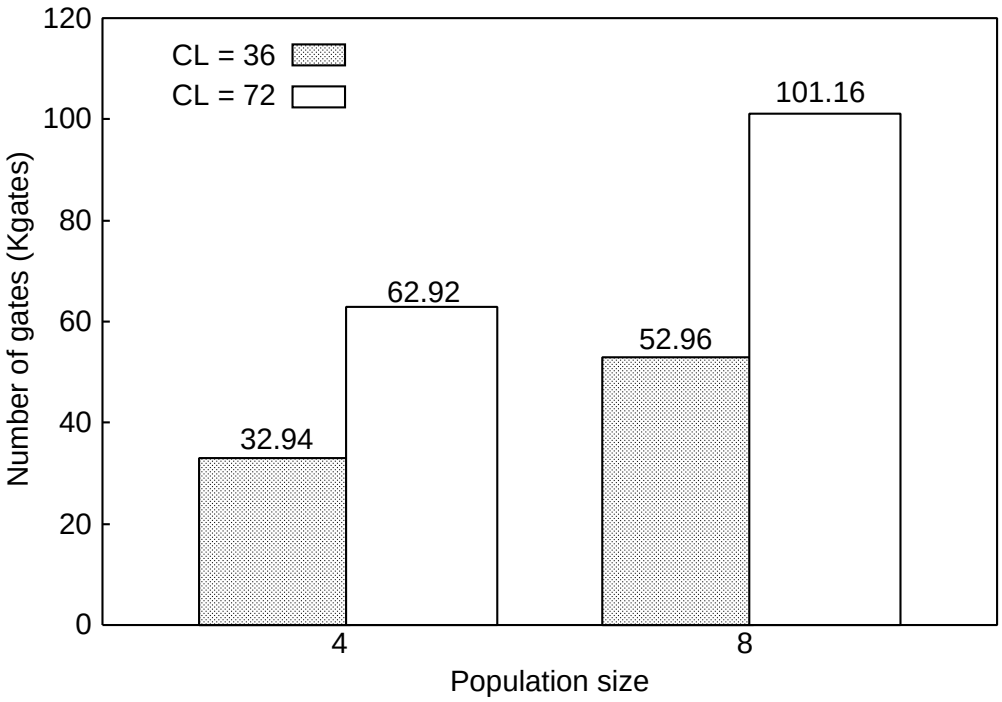


図 4.14: 重みトレーニングのための GA 回路の回路規模

表 4.1: 構成ユニットの回路規模 (Kgates)

	個体数=4		個体 数=8	
	$CL = 36$	$CL = 72$	$CL = 36$	$CL = 72$
`init_pop`	3.10	5.98	6.02	11.78
`sel_ps`	5.16	9.48	13.18	23.98
`cross`	2.22	4.46	2.24	4.48
`muta`	6.74	12.50	9.76	18.10
`fitness`	4.32	7.80	6.84	12.70
`best_chrom`	3.78	7.56	6.92	13.44

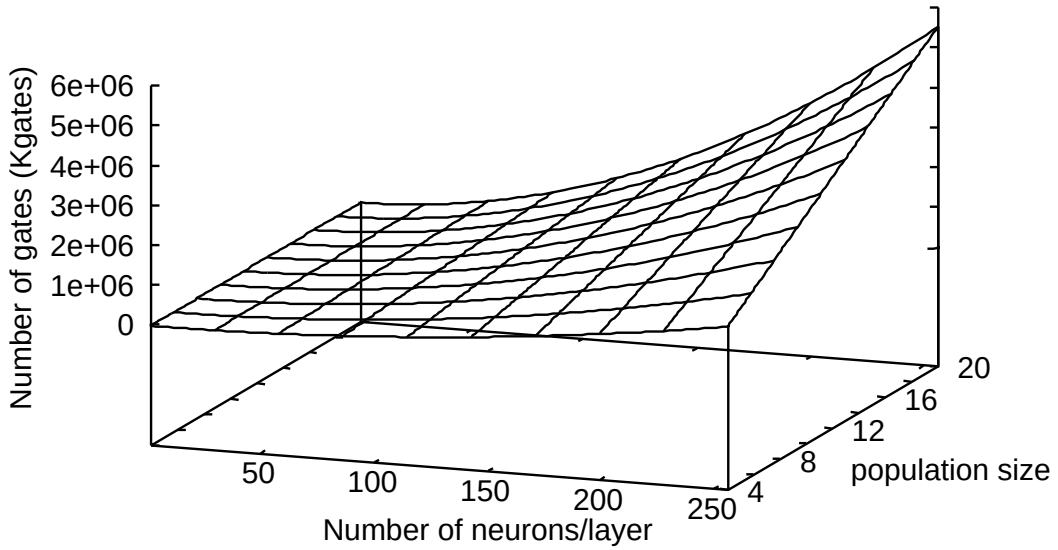


図 4.15: 重みトレーニングのための GA 回路の回路規模の変化

$$c'_{ga} = 0.8 \times c_{ga} \frac{L' \times M' + M' \times N'}{L \times M + M \times N} \tag{4.3}$$

と表すことができる．ここで，c'_{ga} は $\mathrm{NN}(L', M', N')$ の重みトレーニングを行う GA 回路の回路規模，c_{ga} は $\mathrm{NN}(L, M, N)$ の重みトレーニングを行う GA 回路の回路規模を表す．図 4.15 にネットワーク規模の増加に伴う GA 回路の増加量を示す．$\mathrm{NN}(256, 256, 256)$ の重みトレーニングを行う GA 回路で，個体数を 4 とした場合に回路量が 1G ゲートを越えると推定される．

次に，GA 回路の処理時間について考察する．図 4.16 に遺伝的操作 (選択，交叉，突然変異) 部分の処理の流れを示す．図 4.16(a) には GA 回路が行っている処理方式を示し，図 4.16(b) には遺伝的操作を逐次的に行った場合の処理を示す．図 4.16 中の `s`，`c`，`m` はそれぞれ選択，交叉，突然変異を表す．GA 回路での遺伝的操作は選択，交叉，突然変異の 3 つのステージからなり，それぞれ 2 個体の選択，交叉による 2 個体の生成，新たに生成された 2 個体に対する突然変異を行う．逐次処理の場合，2 個体の選択と交叉による 2 個体生成を個体数分の子個体を生成するまで繰り返し，その後，突然変異が行われることになる．ここで，1 個体の選択に要する時間，1 回の交叉に要する時間，1 個体の突然変異に要する時間を等しいものと

し，これを遺伝的操作の 1 ステップに要する時間 t_g とする．このとき，パイプライン処理の処理時間は

$$2t_g + 2t_g + \frac{P}{2}2t_g = (P+4)t_g \tag{4.4}$$

となる．P は個体数を表す．これに対し，逐次処理の場合は

$$Pt_g + \frac{P}{2}t_g + Pt_g = \frac{5}{2}Pt_g \tag{4.5}$$

となる．図 4.10，4.12 に示した GA 回路では，遺伝的操作の実行をパイプライン的に行うことで，処理を高速化している．GA 回路全体の処理時間は，初期集団生成フェーズにおいて 1 個体の生成に要する時間を t_i，1 個体の適応度評価に要する時間を t_f とし，最大世代数を GN とすると，

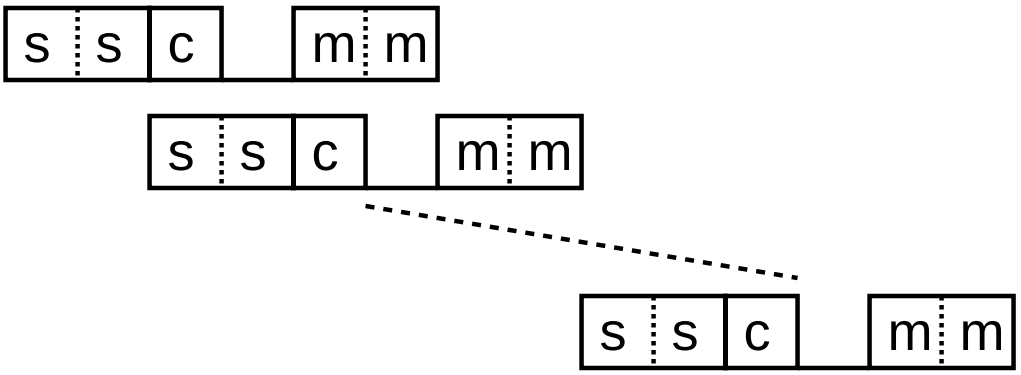
$$Pt_i + GN\{(P+4)t_g + Pt_f\} \tag{4.6}$$

となる．

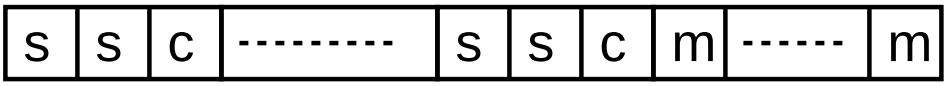
NN の重みトレーニングを行う場合，初期集団の生成に CL/r 回の乱数生成が必要である．r は乱数のビット幅を表す．また，適応度評価では個体ごとにトレーニングパターン分の階層型 NN の実行とその実行結果を用いた出力誤差の計算が行われる．そのため，初期集団生成と適応度評価の処理時間は個体数だけでなくネットワークサイズによっても変化する．ここで，乱数 1 個の生成に要する時間を t_r とすると，初期集団生成に要する時間は以下のように表される．

$$t_i = \{(L+1)M + (M+1)N\}t_r \tag{4.7}$$

ここでは乱数のビット幅と結合重みのビット幅を等しいものとした．また，第 3 章の式 3.1 に示した NN(L, M, N) の実行時間から，適応度評価に要する時間は以下のように表される．t_{n1} はニューロンの積和演算に要する時間，t_{n2} は活性化関数



(a) パイプライン処理



(b) 逐次処理

図 4.16: 遺伝的操作の処理方式

処理に要する時間，t_e は出力誤差の計算に要する時間を表す．pt はトレーニングパターン数を表す．

$$t_f = \{(L+M)t_{n1} + t_{n2}\}pt + t_e \tag{4.8}$$

式 4.6，4.7，4.8 より，NN の重みトレーニングのための GA 回路の処理時間を概算する．トレーニングパターン数を 4，最大世代数を 1,000 世代とした．また，各処理に要する時間は回路のシミュレーション結果より，$t_r = 2cc$，$t_g = 3cc$，$t_{n1} = 2cc$，$t_{n2} = 2cc$，$t_e = 5cc$ とした．図 4.17 に GA 回路の処理時間の概算を示す．

また，図 4.18 にソフトウェアによる重みトレーニングとの比較を示す．これは個体数を 10 としたときに 1,000 世代までの実行に要するステップ数を表す．横軸は重みトレーニングの対象となる階層型 NN の各層に含まれるニューロン数を表す．

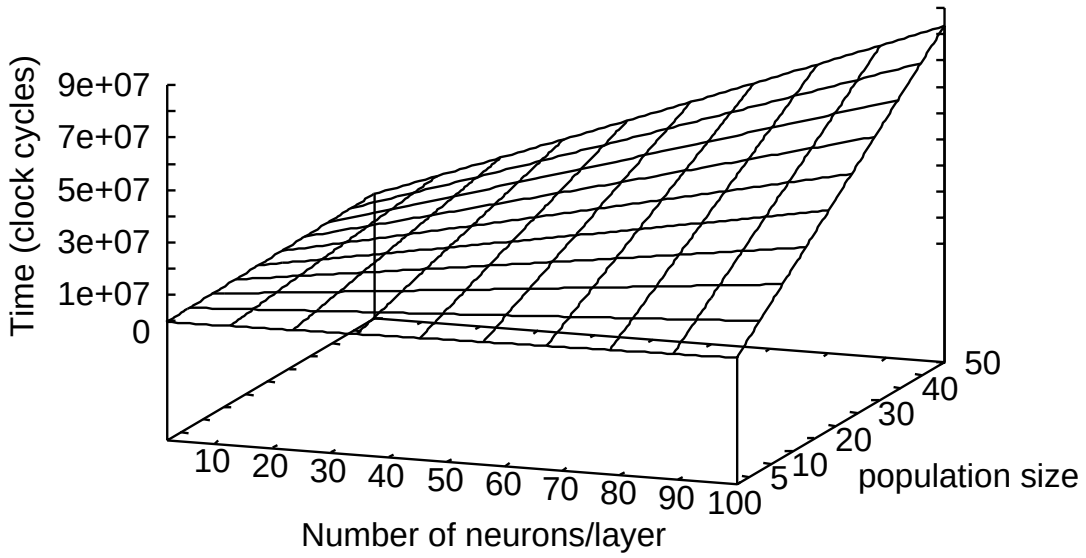


図 4.17: GA 回路の処理時間

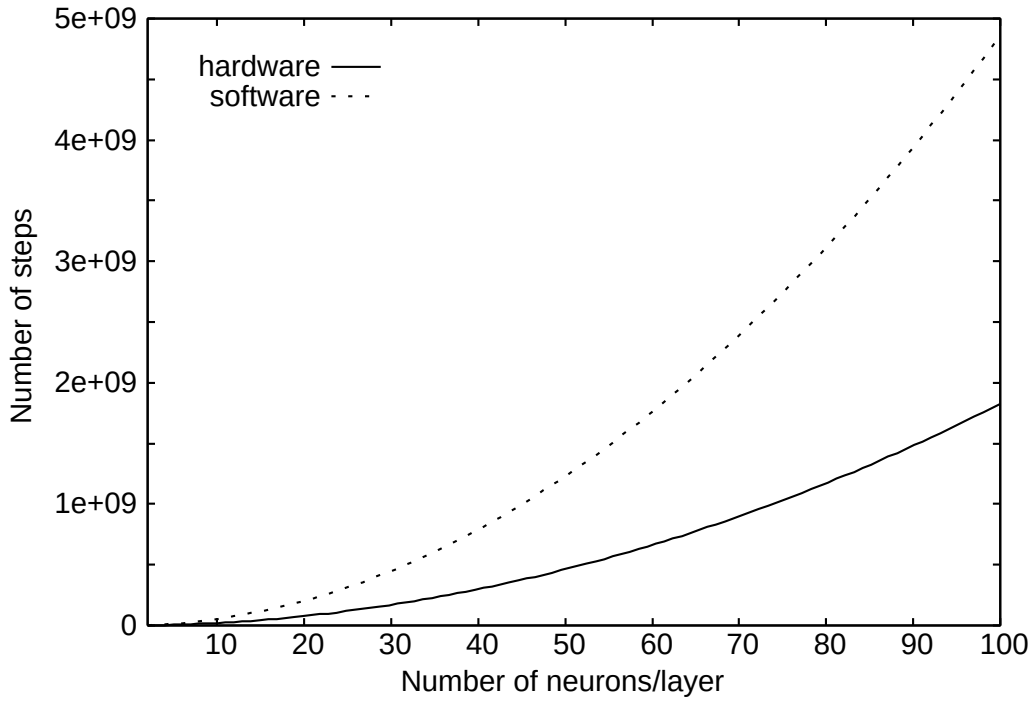


図 4.18: ソフトウェアとの比較

4.5 遺伝的アルゴリズムによる重みトレーニングシステムのハードウェア実装

4.5.1 ハードウェア実装環境

第 4.4 節で述べた GA 回路を FPGA 上に実装し，その動作確認を行った．使用した FPGA ボードは Alitec 社の P5E-100E で，ALTERA 社の EPF-10K100EQC240-3 を搭載している．図 4.19 に使用した FPGA ボードを示し，実装環境を表 4.2 にまとめる．

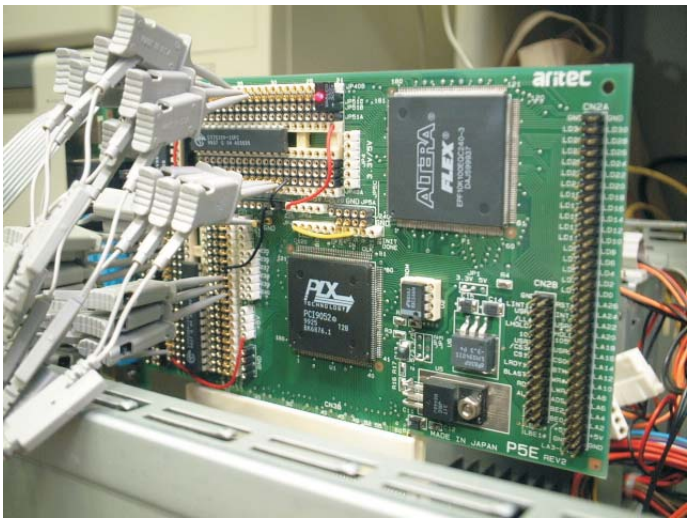


図 4.19: 使用した FPGA ボード

表 4.2: 実装環境

FPGA ボード	P5E-100E (Alitec)
FPGA チップ	EPF-10K100EQC240-3 (ALTERA, 100Kgate)
回路設計・論理合成ツール	MAX+Plus II (ALTERA)

4.5.2 ハードウェアGAの動作検証

図 4.20 に One-Max 問題のための GA 回路の動作波形を示す．実行に使用した回路では，個体数を 4，個体長を 8 としている．図中の各信号は以下の通りである．

- clk, reset：GA 回路のクロック，リセット信号
- xs, fit：各世代ごとの個体群とその適応度
- generation：世代数
- itr：GA の繰り返しを制御するための制御信号
- end_pop, end_sel, end_muta, end_fit：それぞれ初期集団生成，選択，突然変異，適応度評価の終了を示す制御信号
- ps1/ps2：選択操作によって選択された個体
- cxs1/csx2：交叉によって生成された子個体
- mxs：突然変異適用後の個体群

reset 信号が low になると，初期集団の生成が開始され，初期集団生成後に end_pop 信号が high になる．end_pop 信号が high になると GA の繰り返し処理が開始される．itr 信号が high の間は GA の 1 世代分の処理が行われていることを示している．end_sel 信号は交叉の対象となる 2 個体を選択したことを示す信号で，この信号が high になると交叉が実行される．2 個体が選択されるごとに交叉，突然変異が実行され，次世代集団が生成される．sel_ps ユニットから muta ユニットまではパイプライン的に処理を行っているのが分かる．end_muta 信号が high になると新たに生成された集団の適応度が評価される．評価が終わると，end_fit 信号が high になるとともに gen に 1 が加えられ，新しい世代に対する処理が始まる．

次に，NN の重みトレーニングのための GA 回路の動作波形を図 4.21 に示す．実行に使用した回路では，個体数を 4，結合重みのビット幅を 8 ビットとし，NN$(2, 2, 1)$ の重みトレーニングを行う．図中の各信号は以下の通りである．

- clk, reset：GA 回路のクロック，リセット信号

- xs, : 各世代ごとの個体群
- end_pop, end_sel, end_muta, end_fit : それぞれ初期集団生成，選択，突然変異，適応度評価の終了を示す制御信号
- end_ga：GA 回路の実行終了を示す信号
- bestchrom : 最終世代における最良個体 (NN(2, 2, 1) で使用される結合重みセット)

交叉，突然変異等の処理に関しては One-Max 問題のための GA 回路と同様の動作をするが，最終世代の個体に対する適応度評価が終了すると，best_chrom ユニットにおいて最良個体の選択が実行される．これが終了すると GA 回路の実行が終了することになり，end_ga 信号が high になる．fitness ユニットでは，各個体が表現する結合重みとしきい値を階層型 NN にマッピングし，トレーニングパターンを与えて階層型 NN を実行する．そして，その結果得られた出力と期待出力との誤差を計算している．そのため，適応度評価に長時間を要する．

図 4.20，図 4.21 に示した動作波形は論理合成ツールによるシミュレーション結果であるが，FPGA 上に実装した GA 回路の実行でも，これらのシミュレーション結果と同様の結果が得られることを確認した．また，この回路がクロック周波数 33MHz で動作することを確認した．

4.6 まとめ

本章では，ネットワーク構成の確定した階層型 NN に対する結合重みの獲得手法として，GA による NN の重みトレーニングについて検討した．GA による NN の重みトレーニングは NN と GA の融合という観点で行われている研究の一例であり，GA の持つ大域サンプリングの特徴を活かし，局所解に陥る可能性を低減し，効率の良いトレーニングが可能になる．また，BP などの学習アルゴリズムによる重みの更新とは異なり，階層型 NN 回路とは独立して動作するシステムとして重みトレーニングを実現できるという利点がある．まず，ソフトウェア・シミュレーションによるパターン認識実験を行い，階層型 NN から故障ニューロンをすべて

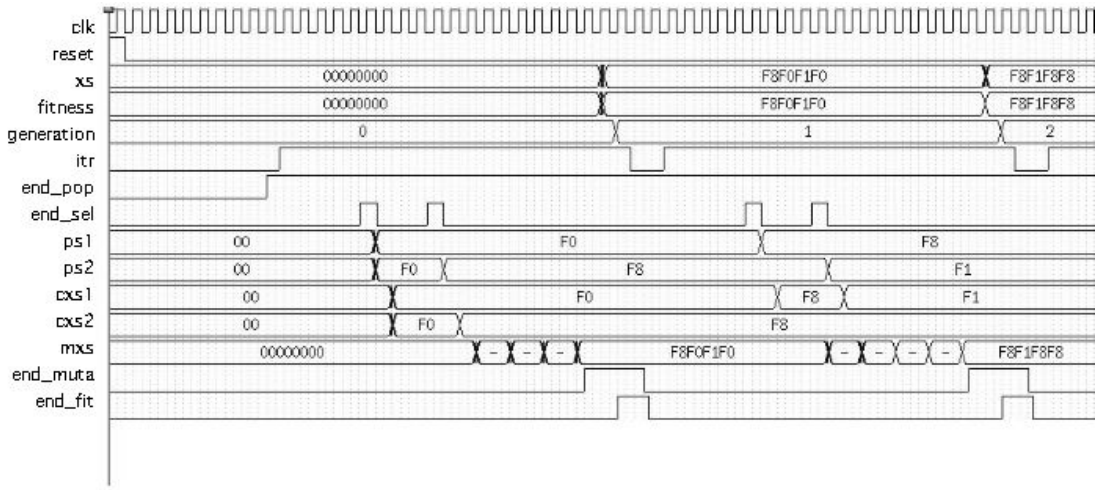


図 4.20: One-Max 問題のための GA 回路の動作

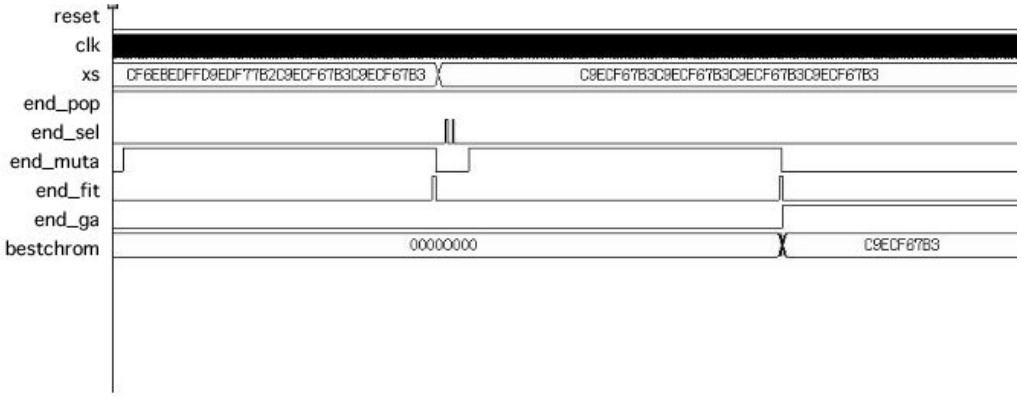


図 4.21: NN の重みトレーニングのための GA 回路の動作

取り除いた縮退ネットワーク上での GA による重みトレーニングの評価を行った．次に，NN の重みトレーニングを行うハードウェア GA の回路設計およびハードウェア実装を行った．

ソフトウェア・シミュレーションによるパターン認識実験から，故障の発生により使用できる中間層のニューロン数が減った状態，つまり，ネットワーク構成が変更された状態でも GA による重みトレーニングによって正常に動作する階層型 NN の結合重みが得られることを確認した．このことから，ハードウェア NN システムにおいて，機能シフト法を用いてネットワーク構成を変更した場合の結合重みの獲得に，GA による重みトレーニングを利用できることが示された．

GA のハードウェア化では．GA は適用する問題に応じて遺伝的操作を選択する必要があること，問題に応じて適応度を評価する評価関数が異なることから，GA アルゴリズムの実行ステップごとにユニット化した．ユニット単位で回路の変更が行えるため，選択や交叉といった処理の変更が容易である．One-Max 問題のための GA 回路と重みトレーニングのための GA 回路の設計を行い，それぞれの回路規模を評価した結果，個体数や個体長に応じて個体を格納するために大きなメモリ領域を必要とすることが分かった．重みトレーニングのための GA 回路では，$NN(256, 256, 256)$ の重みトレーニングのために，個体数 4 の場合でも 1G ゲートを越える回路量が必要になると推定された．

設計した GA 回路を FPGA 上に実装し，その動作確認を行った．また，FPGA 上に実装した GA 回路はクロック周波数 33MHz で動作することを確認した．

第 5 章

自律再構成能力を持つ階層型ニューラルネットワーク・システム

5.1 はじめに

ニューラルネットワーク (Neural Network: 以下，NN) のハードウェア実装の目的の一つとして，組み込みシステムへの応用が挙げられる．組み込みシステムへの応用を想定した NN のハードウェア実装では，より高度な信頼性が求められる．また，外部の制御用コンピュータ等を使用しない独立したシステムであることや，故障発生時には人間の介入なしに復旧できることが望まれる．そのため，ホストコンピュータ等を用いずに，それ自身が自律的に短時間で故障補償を行えることが重要になる．すなわち，組み込みシステム向けの NN には，故障の発生に対して必要に応じて自律的にネットワーク構成を変更するとともに，新しいネットワーク構成に最適な結合重みを自動的に獲得するという，二つの機能を併せ持つ自律再構成能力が必要不可欠である．

これまでの研究では，遺伝的アルゴリズム (Genetic Algorithm: 以下，GA) を用いた NN のネットワーク構造や結合重みの決定自動化に関する研究が行われている [36–41]．NN と GA はともに自然界の生物に関わる原理のモデル化であり，この 2 つの研究には多くの共通点があることから，NN と GA の融合という観点で様々な研究が行われている．ネットワーク構成の決定や結合重みの獲得の自動化はその一例である．GA による結合重みの獲得に関しては第 4 で述べた．ネットワーク

構成の決定に関しては，適用するアプリケーションに応じて最適なネットワーク構成を決定することは非常に困難であるため，その自動化手法が研究されている．GA を用いたネットワーク構成の自動決定はその一手法である．GA の実行により，NN を構成している各ニューロンの接続状態を最適化することで，ネットワーク構成を自動的に決定する．また，北野が提案しているニューロジェネティック・ラーニング理論 [41] は NN の構造決定と結合重みのトレーニングを同時に行うことができる．

GA を用いたこれらの研究はあくまでもソフトウェア・シミュレーションにより実行される NN においてネットワーク構成や結合重みを自動的に決定する試みである．本論文で議論するハードウェア NN システムの自律再構成では，故障を含んだ階層型 NN から故障を完全に取り除くためにネットワーク構成を変更する．すなわち，ネットワーク構成の変更とは，故障ニューロンを取り除くことによって各層のニューロン数を変更することを意味し，GA によるネットワーク構成の自動決定とは目的が異なる．また，ネットワーク構成を変更した場合には，それに適した結合重みを再度獲得しなければならず，ネットワーク構成の変更と結合重みの獲得の両方を実現できなければならない．現時点では，上記のような意味でのハードウェア NN システムにおける自律再構成に関する研究は十分になされておらず，自律再構成能力を有する再構成型 NN のためのハードウェアアーキテクチャに関しても明らかにされていない．

本章では，ハードウェア上に発生する故障に対して外部のホストコンピュータ等を用いることなく，それ自身が自律的に故障補償を行う再構成型 NN の実現を目標に，ハードウェア NN システムにおける自律再構成手法とそのハードウェアアーキテクチャを提案する．まず，冗長手法による故障補償と GA による重みトレーニングによる結合重みの獲得を組み合わせた自律再構成手法を提案する．これは第 3 章で述べた機能シフト法と 4 章で述べた GA による重みトレーニングを組み合わせた手法である．また，提案する再構成手法を実現した自律再構成機能を有する再構成型 NN システムについて論じる．

5.2 階層型ニューラルネットワークの自律再構成

本論文では，(1) 故障ニューロンを完全に取り除くために，必要に応じてネットワーク構成を変更し，(2) 新しいネットワークのための結合重みを獲得すること，を NN の再構成と定義する．また，この二つの処理を外部のホストコンピュータ等を用いずに自律的に実行することをハードウェア NN システムにおける自律再構成とし，自律再構成能力を持つハードウェア NN システムを再構成型 NN と呼ぶ．ハードウェア NN システムの自律再構成を実現するために，第 3 章で述べた機能シフト法と第 4 章で述べた GA による重みトレーニングを組み合わせた自律再構成手法を提案する．提案手法では，NN の再構成の定義の (1) を実現するために機能シフト法を用い，(2) を実現するために GA による重みトレーニングを用いる．これら二つの手法を同一チップ上に実装することで，ハードウェア NN システムの自律再構成を実現する．

機能シフト法は第 3 章で述べたように，予備ニューロンを用いて故障ニューロンを補償する手法である．中間層と出力層にそれぞれ複数個の予備ニューロンを配置し，単純な操作で同一層内に存在する故障ニューロンの機能を予備ニューロンで置き換える．これは冗長手法に分類される．冗長手法には，予備ニューロン数以上の故障が発生した場合には完全な故障補償が不可能であるという欠点がある．機能シフト法においても，故障ニューロン数が予備ニューロン数よりも多い場合にはいくつかの故障ニューロンが取り除かれないまま存在することになる．ハードウェア NN システムの動作を保証するためには，NN の実行時に故障ニューロンを使用しないようにする必要がある．これは，故障ニューロンをネットワークから完全に切り離し，実行に必要なニューロン数よりも少ない状態で実行できるようにすればよい．そこで，NN を正常なニューロンのみで構成されるようにネットワーク構成を変更する．ここで，ネットワーク構成の変更とは，中間層で使用するニューロンの個数を変更することを指す．出力層においては最低限必要なニューロン数が決まっており，出力層で使用するニューロン数を変更することはできないので，出力層ニューロン数の変更については考慮しない．このように，故障ニューロンの置き換えだけでなく，中間層で使用するニューロン数の変更にも機能シフト法を利用することで，自律再構成の第 1 段階であるネットワーク構成の変更を

実現する．

ネットワーク構成を変更した場合，新しいネットワーク構成のための結合重みを再度獲得する必要がある．機能シフト法によって完全に故障を取り除かれた階層型 NN の結合重みは第 4 章で述べたように，GA による重みトレーニングによって獲得できる．これが自律再構成の第 2 段階である結合重みの再獲得になる．GA による重みトレーニングはネットワーク構成変更後の結合重みの獲得だけでなく，初期状態での結合重みの獲得にも使用できる．

提案するハードウェア NN システムの再構成手法は以下の通りである．

1. 中間層と出力層に存在するすべてのニューロンを故障ニューロンと非故障ニューロンに分類する．
2. 故障数に応じて次の処理を行う．

   - 故障ニューロン数 $\leq$ 予備ニューロン数 の場合

   (a) 機能シフト法により故障ニューロンを予備ニューロンで置き換える．

   (b) 重み獲得回路を通じて Weight Table から各ニューロンに重みを割り当てる．

   - 故障ニューロン数 $>$ 予備ニューロン数 の場合 (中間層において)

   (a) 機能シフト法により，予備ニューロン数と同数の故障ニューロンを取り除く．

   (b) 残りの故障ニューロンを使用停止とする (ネットワーク構成変更)．

   (c) GA による重みトレーニングによって新しいネットワーク構成のための重みを計算し，Weight Table を更新する．

   (d) 重み獲得回路を通じて Weight Table から各ニューロンに重みを割り当てる．

この再構成手法では，故障ニューロン数が予備ニューロン数以下の場合，GA による重みトレーニングを行う必要はなく，機能シフト法のみの実行によって，より高速な故障補償が可能である．また，予備ニューロン数以上の故障が発生した場合でも，機能シフト法によりネットワーク構成を変更した後，そのネットワーク構成に対する結合重みを GA によって獲得することで故障の影響を取り除くことができる．

この再構成処理はシステムの初期化時または故障検出時に行われるため，過渡的な故障の発生等により故障とみなされたニューロンも，次の再構成時に正常であると判断された場合には再度使用可能である．

5.3 再構成型ニューラルネットワークのハードウェア化とその評価

提案した自律再構成手法を実現する再構成型NNのシステム構成を図5.1に示す．このシステムでは，機能シフト法を組み込んだ階層型NN回路と重みトレーニングのためのGA回路を同一チップ上に実装することで，外部のホストコンピュータ等を用いることなく自律再構成が可能な再構成型NNを実現する．再構成型NNはNNパートとトレーニング・パート，RAMアクセス制御パートの3つから構成される．

NNパートは機能シフト法を組み込んだ階層型NNである．ここでは，NNの実行のほかに，故障ニューロン数に応じて，機能シフト法による故障補償とネットワーク構成の変更を行う．故障ニューロン数が予備ニューロン数以下であれば(複数の故障が一つのニューロンに集中することもあり，これは一つのニューロンが故障しているとみなされる)，NNパート内で動作する機能シフト法のみの実行で故障ニューロンを取り除くことができる．回路構成は第3章の図3.6に示した故障補償機能付きNN回路と同等である．NNの実行開始信号が入力されると，結合重みを保持するRAMから必要な結合重みを読み出し，Weight Tableを構成する．そして，階層型NNを構成しているニューロンの状態をチェックし，故障ニューロンが存在する場合には，機能シフト法による故障補償を行う．ここで，故障ニューロンの数が予備ニューロン数以下ならば，機能シフト法により故障ニューロンを取り除いたあとにNNの実行が行われる．しかし，予備ニューロン数以上の故障が存在する場合には，NNの実行を中止し，GAによる重みトレーニングを行うための制御信号を送る．GAの実行が終了し，新たな結合重みがRAMに書き込まれると，再度RAMから結合重みを読み出し，NNの実行を再開する．

トレーニング・パートはNNの重みトレーニングを行うGA回路であり，機能シ

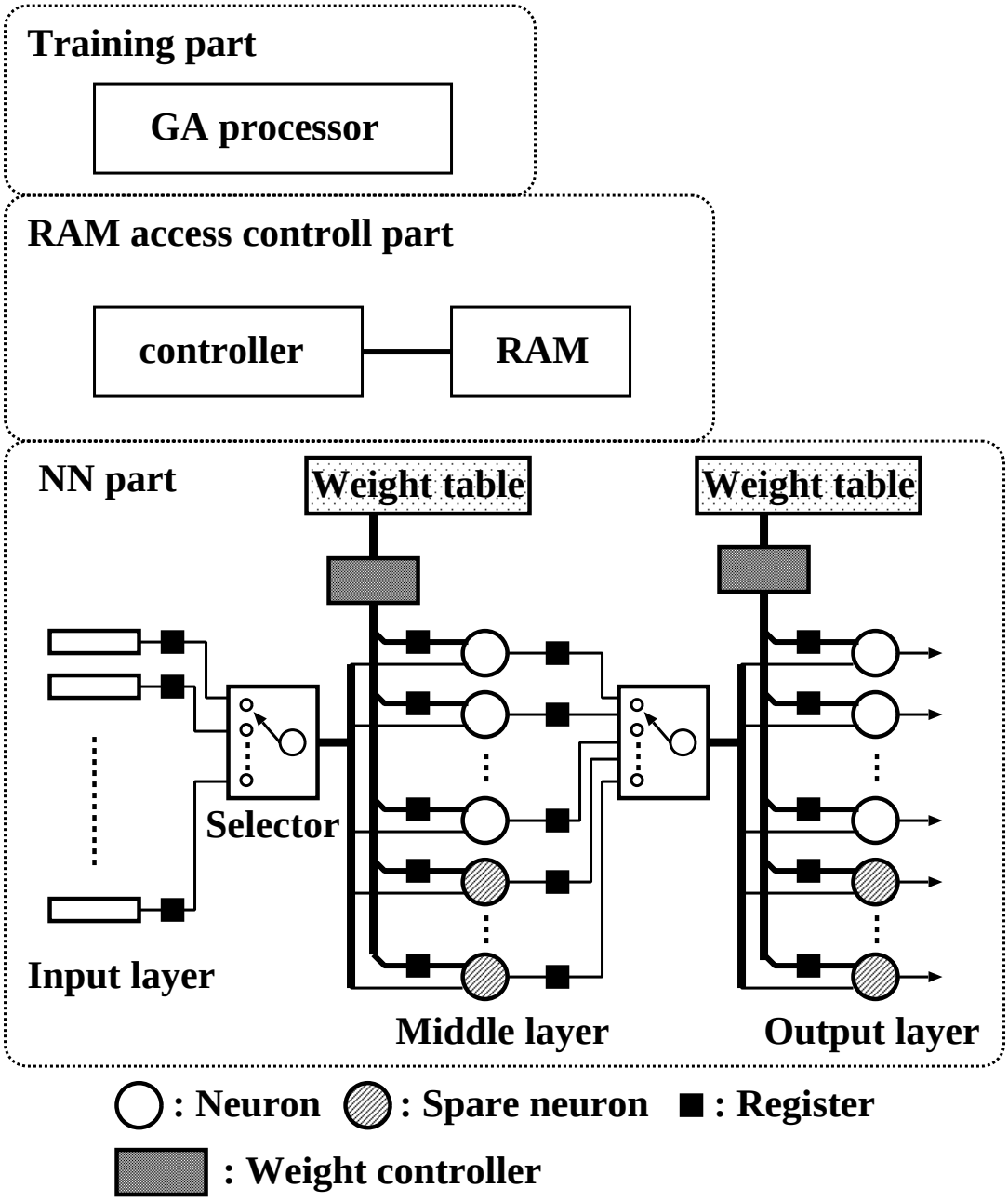


図 5.1: 再構成型 NN のシステム構成

フト法によってネットワーク構成が変更された場合に，GA による重みトレーニングを実行して新しいネットワーク構成のための結合重みを獲得する．回路構成は第 4 章の図 4.12 に示した GA 回路と同等である．GA の実行を開始する制御信号が入力されたときのみ動作し，その実行が終了すると，得られた結合重みとともに実行終了信号を RAM アクセス制御パートに送り，結合重みを RAM に書き込む．

RAM アクセス制御パートはトレーニング・パートと NN パートとの間で結合重みの受け渡しを行う回路である．トレーニング・パートにおいて GA の実行が終了すると，その結果得られた NN の結合重みを RAM アクセス制御パート内のコントローラを通じて RAM に書き込む．このとき，NN パートに対して結合重みの書き込み終了信号を送る．NN パートからのアクセスでは，NN の実行開始信号を受け取ると，RAM から結合重みを読み出し，NN パートに送る．

図 5.2 は再構成型 NN の回路構成を示す．図中の波線は結合重みとして使用されるデータの流れを表し，実線は制御信号の流れを表す．各パートの動作を制御する制御回路 (図中の `GA controller`，`access controller`，`NN controller`) のほかに，全体の動作を制御するための回路 (図中の `global controller`) がある．

図 5.3 に，GA による重みトレーニングによって得られた結合重みを RAM アクセス制御パートを通じてやりとりする様子を示す．実行に使用する階層型 NN は 2 入力排他的論理和 (XOR) 関数のための階層型 NN で，その構成は NN$(2, 2, 1)$ である．図中の各信号は以下の通りである．

- `clk`, `reset`：回路のクロック，リセット信号
- `x`, `y`：NN の入力と出力
- `end_ga`：GA の実行終了を表す信号
- `end_ww`，`end_rw`：結合重みの RAM への書き込み／読み出しが終了したことを示す信号
- `end_nn`：NN の実行終了を表す信号
- `wn00/wn01`：中間層ニューロンの結合重み
- `wo00`：出力層ニューロンの結合重み

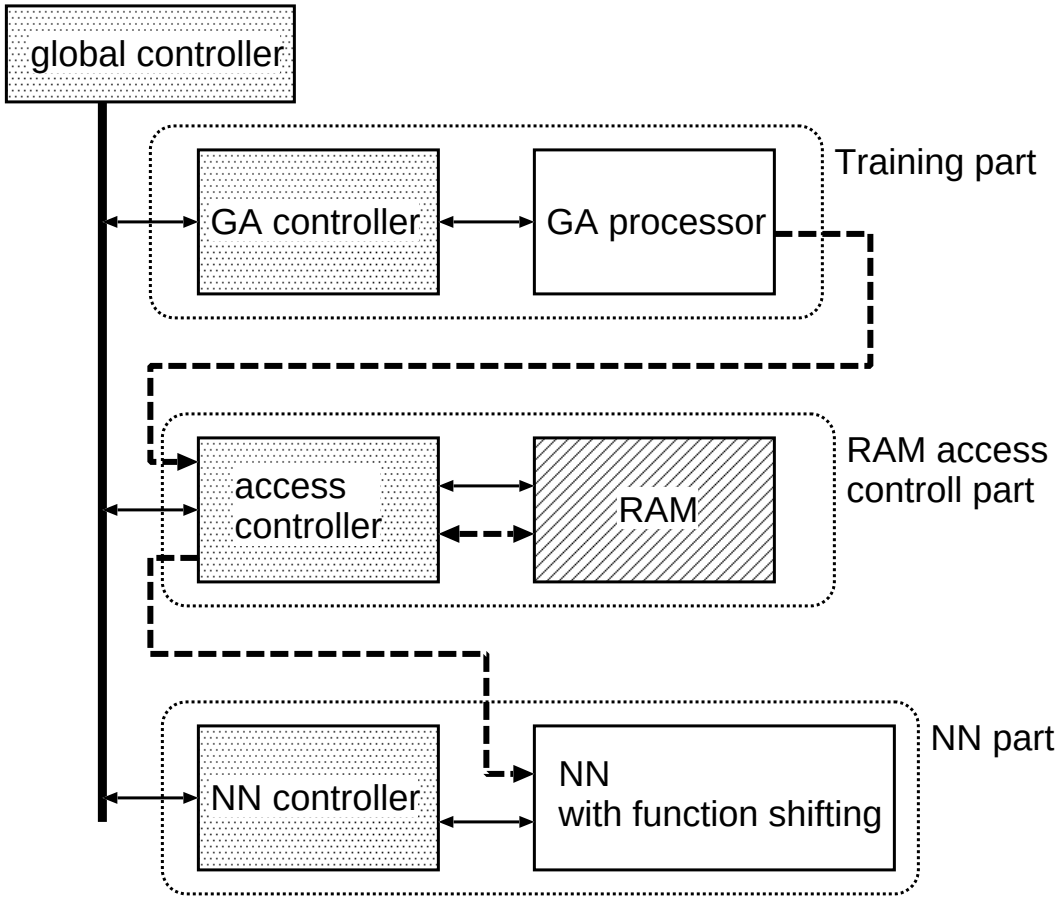


図 5.2: 再構成型 NN の回路構成

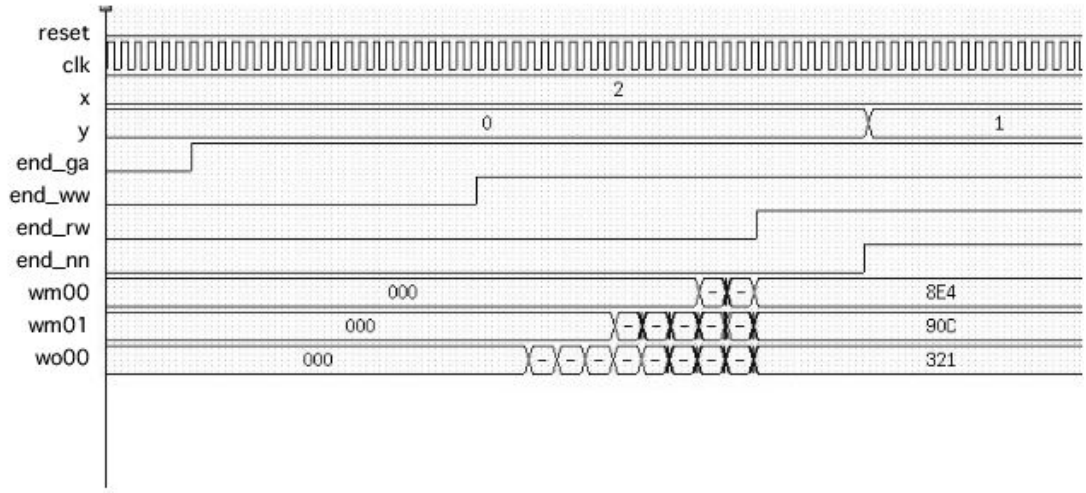


図 5.3: 重み更新時の動作波形

トレーニング・パートにおける GA の実行が終了すると，end_ga 信号が high になり，RAM への書き込みが開始される．RAM への書き込みが終了すると end_ww 信号が high になる．このとき，RAM アクセス制御パートから NN パートへと NN の実行開始信号が送られる．RAM アクセス制御パートから NN パートへ実行開始信号が送られると，RAM から各層の結合重みを読み出し，Weight Table を作成する．この作業が終了すると end_rw 信号が high になり，NN の実行が開始される．wm00，wm01，wo00 の変化は Weight Table 作成時のものである．NN の実行が終了したことを示す end_nn 信号が high になるとともに，NN の出力である y が変化していることが分かる．ここで示した NN の実行では，入力として $(1, 0)$ を与えており，y=1 と正しい出力が得られている．GA 回路の動作は第 4 で述べたものと同等であるため，ここではその説明は省略する．

2 入力排他的論理和のための再構成型 NN の回路規模を表 5.1 に示す．これらの値は ALTERA 社の回路設計・論理合成ツールである MAX+PlusII を用いて FPGA 上にマッピングし，見積もった回路規模である．また，各パートの回路規模はそれぞれを MAX+Plus II によってマッピングした結果得られた値であり，再構成型 NN 回路全体において実際に各パートが占める回路規模とは厳密には異なる．

ネットワークサイズが $(2, 2, 1)$ 程度の再構成型 NN では，故障補償機能付き NN 回路の回路規模はそれほど大きくなく，重みトレーニングのための GA 回路が非

表 5.1: 再構成型 NN の回路規模 (Kgates): NN$(2, 2, 1)$

再構成型 NN	38.14
RAM アクセス制御パート	2.16
NN パート	3.30
トレーニング・パート	32.94

常に大きくなってしまう．これは個体を保持するためのメモリ領域が大きいことが原因であると考えられる．ネットワークサイズが大きくなるにつれて，このメモリ領域だけでなく適応度評価のために GA 回路に組み込まれている階層型 NN 回路の回路規模も大きくなるため，トレーニング・パートの構成に関してはさらなる議論が必要である．

再構成型 NN には結合重みを格納するための RAM が含まれる．結合重みを格納するための RAM は結合重みのビット幅とネットワークサイズにより変化する．結合重みのビット幅を w とすると，$w\{(L+1)M+(M+1)N\}$ ビットの RAM が必要となる．結合重みを 8 ビットとした場合，NN$(256, 256, 256)$ の再構成型 NN のハードウェア実装では，約 130M バイトの RAM が必要になる．これは容易に実装できる容量である．

ここで，ネットワーク規模の増加に伴う再構成型 NN の回路の増加量について考察する．ネットワークサイズを NN(L, M, N) から NN(L', M', N') に変更した場合の NN パートおよびトレーニング・パートの回路の増加量は，第 3 章，第 4 章で議論した通りである．RAM アクセス制御パート内のコントローラ部分の回路規模はネットワークサイズによらず一定である．これらのことから，再構成型 NN の回路規模は次の式で見積もることができる．ここで，c_{nn} は NN(L, M, N) のときの NN パートの回路規模を表し，c_{ga} は NN(L, M, N) のときのトレーニング・パートの回路規模を表す．

$$c_{nn}\frac{L'+M'+N'}{L+M+N}+0.8\times c_{ga}\frac{L'\times M'+M'\times N'}{L\times M+M\times N}+2.16 \tag{5.1}$$

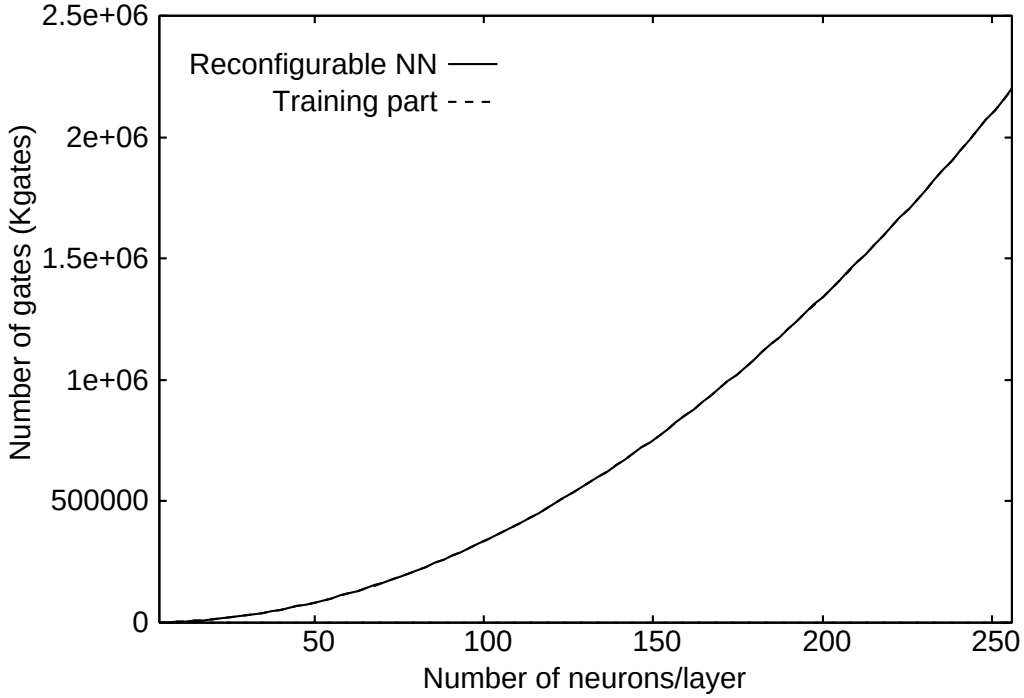


図 5.4: ネットワーク規模による回路規模の変化

図 5.4 に，NN$(2, 2, 1)$ のときの回路規模を基準とし，ネットワーク規模を変化させたときの再構成型 NN の回路の増加量を示す．トレーニング・パートにおいては，個体数を 10 としたときの回路規模を基準としている．再構成型 NN の回路規模を示すグラフとトレーニング・パートの回路規模を示すグラフがほぼ重なっており，再構成型 NN の回路の大部分をトレーニング・パートが占めていることが分かる．また，ネットワーク規模が大きくなるにつれて，トレーニング・パートの回路規模が大幅に増加し，そのために再構成型 NN の回路規模も大幅に増加する．

トレーニング・パートの回路規模の増加に関しては，第 4 章で議論したように，GA の実行に使用する個体や適応度を格納するために膨大な量のレジスタが必要になることが原因と考えられる．ネットワーク規模が大きくなるにつれて，その結合重みをマッピングした個体の個体長は非常に長くなる．個体を格納するためのレジスタ量を削減するには，個体長を短くするために個体の表現方法を変更するか，少ない個体数で効率よく実行できるような GA アルゴリズムを採用するといった方法が考えられる．一例として，Harik らによって提案された Compact GA [31] が挙げられる．Compact GA では，個体集団を一つの確率ベクトルで表現し，実際

の個体集団ではなく，確率ベクトルに対して処理を行う．そのため，個体を格納するためのメモリ容量を大幅に削減できる．そのため，ハードウェア実装向きのアルゴリズムであるといわれており，Aporntewan らによって Compact GA のハードウェア化も行われている [32]．しかし，複雑な問題には不向きであるという欠点があり，Compact GA の拡張アルゴリズム等の研究も行われている．Compact GA はハードウェア化に適したアルゴリズムであると考えられるが，NN の重みトレーニングを行うためにはアルゴリズムの改良が必要である．

5.4 まとめ

本章では，ハードウェア上に発生する故障に対して，外部のホストコンピュータ等を用いることなく，自律的に故障補償を行う再構成型 NN の実現を目標に，ハードウェア NN システムの自律再構成手法とそのハードウェアアーキテクチャについて検討した．まず，機能シフト法と GA による重みトレーニングを組み合わせたハードウェア NN システムの自律再構成手法を提案した．次に，自律再構成能力を持つ再構成型 NN のハードウェアアーキテクチャを提案し，そのハードウェア化を行った．

ハードウェア NN システムの自律再構成を，(1) 故障ニューロンを完全に取り除くために，必要に応じてネットワーク構成を変更し，(2) 新しいネットワーク構成のための結合重みを獲得すること，と定義した．提案した自律再構成手法では，(1) を実現するために機能シフト法を用い，(2) を実現するために GA による重みトレーニングを用いる．機能シフト法は冗長手法に分類される故障補償手法であり，予備ニューロンを付加し，ハードウェア的に故障箇所を取り除くため，予備ニューロン数以上の故障には対処できないという欠点があった．ただし，故障ニューロンをネットワークから切り離すことで，ニューロン数の異なる縮退型ネットワークを構成することが可能である．ネットワーク構成を変更した場合，新しいネットワーク構成のための結合重みを再度獲得する必要がある．ネットワーク構成の確定した階層型 NN の結合重みは GA による重みトレーニングによって獲得することができる．機能シフト法と GA による重みトレーニングを組み合わせることで，ネットワーク構成の変更と結合重みの獲得によって，故障ニューロンを完全

に取り除くことができる．

提案した自律再構成手法では，故障ニューロンが少なく，予備ニューロンで補償可能な場合には，機能シフト法のみの実行によって高速な故障補償が可能である．この場合，ネットワーク構成は変更されないので，GA による重みトレーニングを行う必要がない．予備ニューロン数以上の故障が発生した場合には，機能シフト法によりネットワーク構成を変更し，GA による重みトレーニングによって新しいネットワーク構成に最適な結合重みを獲得することで故障の影響を完全に取り除くことができる．

自律再構成能力を持つ再構成型 NN は機能シフト法を組み込んだ故障補償機能付き NN 回路と重みトレーニングのための GA 回路を同一チップ上に実装することで，故障発生時に外部のホストコンピュータ等を用いることなくネットワーク構成を変更し，新しいネットワーク構成のための結合重みを自律的に獲得できるシステムとなる．また，GA 回路によって，階層型 NN の初期重みの獲得も可能である．再構成型 NN の回路設計を行い，回路規模を見積もった結果，GA による重みトレーニングを行うトレーニング・パートの回路規模が非常に大きく，再構成型 NN の大部分を占めることが分かった．NN パートとトレーニング・パートのと間で結合重みをやりとりするための RAM の容量はそれほど大きくなく，容易に実装できる．トレーニング・パートの回路規模の問題に関しては，ハードウェア化に適したアルゴリズムなど，さらなる検討が必要である．

第 6 章

結言

本論文では，パターン認識の分野でよく用いられる階層型ニューラルネットワーク (Neural Network: 以下，NN) に注目し，組み込みシステムへの応用を前提とした自己修復機能を有する再構成型 NN のハードウェアアーキテクチャを確立することを目的とした．階層型 NN をハードウェア実装する際には，回路規模や配線領域の増加と，ハードウェア上に発生する故障の問題に対処する必要がある．特に，組み込みシステムへの応用を想定したハードウェア実装では，より高度な信頼性が求められ，故障が発生した際には，外部の制御用コンピュータや人間の介入なしに復旧できることが望まれる．そのため，それ自身が自律的に短時間で故障補償できることが重要になる．故障の発生に対して，必要に応じて自律的にネットワーク構成を変更し，新しいネットワーク構成のための結合重みを自動的に獲得する自律再構成能力が必要不可欠である．このように，組み込みシステムへの応用を想定したハードウェア NN システムの実現には，(1) ハードウェア量，特に配線領域を少なく抑える階層型 NN のハードウェアアーキテクチャと，(2) 故障の回避やネットワーク構成の変更と結合重みの獲得を自律的に行うハードウェア・メカニズムの両方を同時に考えなければならない．

まず，配線領域と故障補償容易性を考慮した階層型 NN のハードウェアアーキテクチャと，そのアーキテクチャ上での故障補償手法を提案した．階層型 NN では，各ニューロンが隣接する層のすべてのニューロンと完全結合されるため，ネットワークの大規模化に伴い，ニューロン同士を接続するための配線領域が大幅に増加するという問題がある．このような構造は断線等の故障が増加する原因となる

上に，故障補償が非常に困難になる．そこで，ハードウェアアーキテクチャとして，各層間に入力選択回路を配置したバス接続型 NN を提案した．バス接続型 NN では，層間に配置された入力選択回路によって下位層から順次送られてくる入力を各ニューロン内で逐次的に処理するため，完全結合型の NN 回路に比べ，処理速度は多少劣る．しかし，ニューロン間接続のための配線領域が少なく，ネットワーク規模が大きくなっても配線領域が大幅に増加することがないという利点がある．また，ニューロンの追加による結線数の増加も非常に少なく，故障補償容易性という観点からも優れており，大規模 NN のハードウェア実装に適している．

バス結合型 NN 上での故障補償手法として，機能シフト法を提案した．機能シフト法は冗長手法に分類される故障補償手法で，中間層と出力層に配置した予備ニューロンを用いて，それぞれの層に存在する故障ニューロンを取り除く．同一層内の故障ニューロン数が予備ニューロン数以下である場合には，故障の種類によらず故障補償が可能であるが，予備ニューロン数以上の故障が発生した場合には，実行に必要な数のニューロンで構成された階層型 NN を得ることができない．これは冗長手法による故障補償の欠点でもある．ただし，故障ニューロンをネットワークから完全に切り離すことで，ニューロン数の異なるネットワークを構成することは可能である．つまり，機能シフト法を用いてネットワーク構成を変更し，故障ニューロンを完全に取り除くことが可能である．この場合，新しいネットワーク構成のための結合重みを再度獲得する必要がある．

機能シフト法を実現したバス結合型 NN の回路設計および FPGA を用いたハードウェア実装を行った．回路規模を評価した結果，機能シフト法のための回路追加のオーバーヘッドは比較的小さいことが分かった．しかし，実行に必要なニューロン以外に予備ニューロンを追加する場合には，全体の回路規模と追加する予備ニューロン数とのトレードオフの問題があり，適当な予備ニューロン数を考える必要がある．設計した故障補償機能付き NN 回路を FPGA 上に実装し，その動作を確認した．

次に，ネットワーク構成の確定した NN に対して最適な結合重みを獲得するための手法として，NN の重みトレーニング手法について検討した．階層型 NN の結合重みの更新のために一般に用いられる誤差逆伝搬法 (Back Propagation: 以下，BP) 等の学習アルゴリズムは，結合重みの更新に長時間を要することや，ネット

ワークの大規模化に伴い計算コストが増大することなどの問題点が指摘されている．また，ハードウェア NN システムにおける結合重みの獲得を実現するためには，階層型 NN 回路を構成している多数のニューロン回路に対して重み学習用の特別な回路を付加する必要がある．そこで，階層型 NN 回路とは独立して動作するシステムとして実装できる，GA による重みトレーニングについて検討した．

ソフトウェア・シミュレーションによるパターン認識実験を行い，階層型 NN から故障ニューロンをすべて取り除いた縮退ネットワーク上での GA による重みトレーニングを評価した．この実験により，予備ニューロン数以上の故障ニューロンが存在し，それらを完全に取り除くために実行に必要とされるニューロン数よりも少ない数のニューロンで構成された階層型 NN において，GA による重みトレーニングによって動作可能な結合重みが得られることを確認した．また，重みトレーニングのための GA 回路の設計および FPGA を用いたハードウェア実装を行った．設計した GA 回路の回路規模を評価した結果，個体の格納のために大きなメモリ領域を必要とするため，重みトレーニングの対象となる階層型 NN のネットワーク規模の増大に伴い，非常に多くの回路量を必要とすることが分かった．設計した GA 回路は FPGA 上に実装し，その動作を確認した．

さらに，ハードウェア上に発生する故障に対して，外部のホストコンピュータ等を用いることなく，自律的に故障補償を行う再構成型 NN の実現を目標に，機能シフト法と GA による重みトレーニングを組み合わせたハードウェア NN システムの自律再構成手法を提案し，自律再構成能力を持つ再構成型 NN のハードウェアアーキテクチャを提案した．提案する再構成手法では，故障ニューロンが少なく，予備ニューロンで補償可能な場合には，機能シフト法のみの実行によって高速な故障補償が可能である．この場合，ネットワーク構成は変更されないので，GA による重みトレーニングを行う必要がない．予備ニューロン数以上の故障が発生した場合には，機能シフト法によりネットワーク構成を変更し，GA による重みトレーニングによって新しいネットワーク構成に最適な結合重みを獲得することで故障の影響を完全に取り除くことができる．自律再構成能力を持つ再構成型 NN は機能シフト法を組み込んだ故障補償機能付き NN 回路と重みトレーニングのための GA 回路を同一チップ上に実装することで，故障発生時に外部のホストコンピュータ等を用いることなくネットワーク構成を変更し，新しいネットワーク構

成のための結合重みを自律的に獲得できるシステムとなる．

再構成型 NN の回路設計を行い，回路規模を見積もった．その結果，NN パートと，結合重みをやりとりするための RAM アクセス制御パートの回路規模は比較的小さいが，GA による重みトレーニングを行うトレーニング・パートの回路規模が非常に大きく，再構成型 NN の大部分を占めることが分かった．NN パートとトレーニング・パートのと間で結合重みをやりとりするための RAM の容量はそれほど大きくなく，容易に実装できる．実用的なアプリケーションを実行するためには，少なくとも 10^2 個程度のニューロンを含んだ階層型 NN が必要であると考えられる．トレーニング・パート以外は実装可能な範囲の回路規模であるため，再構成型 NN の実装にはトレーニングパートの回路規模の削減が必要である．トレーニング・パートの回路構成に関して，GA の個体表現やハードウェア化に適した GA アルゴリズムなどを再度検討し，実世界のアプリケーションに適用できる再構成型 NN を構築することが今後の課題である．

謝辞

本研究は北陸先端科学技術大学院大学 情報科学研究科 情報システム学専攻在学中に，堀口 進教授のもとで行われました．本研究を進めるにあたり，終始御指導，御助言を頂いた堀口 進教授に心より感謝致します．

弘前大学 理工学部 電子情報システム工学科 吉岡 良雄教授，北陸先端科学技術大学院大学 情報科学研究科 日比野 靖教授，松澤 照男教授には，審査にあたり御教示，御検討を賜りましたことを深く感謝致します．

本研究全般に渡り貴重な御助言を賜り，また，審査にあたり御教示，御検討頂きました，北陸先端科学技術大学院大学 情報科学研究科 井口 寧助教授に深く御礼申し上げます．

副テーマを行うにあたり，有意義な御指導を賜りました，現 広島大学大学院 工学研究科 中野 浩嗣教授には深く御礼申し上げます．

また，日頃から有益な御助言を頂き，多面に渡り励まして頂いた，現 東北大学大学院 情報科学研究科 福士 将助手，北陸先端科学技術大学院大学 情報科学研究科 林 亮子助手に感謝致します．

最後に，本論文をまとめるにあたって御協力いただいた堀口研究室の皆様に厚く御礼申し上げます．

参考文献

[1] Y. Hirai, K. Kamada, M. Yamada and M. Ooyama, "A Digital Neuro-chip with Unlimited Connectability for Large Scale Neural Networks," Proc. of IJCNN'89, pp.163–169.

[2] J. B. Theeten, M. Duranton, N. Mauduit and J. A. Sirat, "The LNeuro-chip: A Digital VLSI with On-chip Learning Mechanism," Proc. of Int'l Neural Network Conf., pp.593–596, 1990.

[3] Max Stanford Tomlinson Jr, Dennis J. Walker, "DNNA: A Digital Neural Network Architecture," Proc. of Int'l Neural Network Conf., pp.589–592, 1990.

[4] Marcin Skubiszewski, "A Hardware Emulator for Binary Neural Networks," Proc. of Int'l Neural Network Conf., pp.555–558, 1990.

[5] M. Yasunaga, N. Masuda, M. Yagyu, M. Asai, M. Yamada and A. Masaki, "Design, Fabrication and Evaluation of a 5-inch Wafer Scale Neural Network LSI Composed of 576 Digital Neurons," Proc. of IJCNN'90, pp.527-535, 1990.

[6] Brian A. White, Mohamed I. Elmasry, "The Digi-Neocognitron: A Digital Neocognitron Neural Network Model for VLSI," IEEE Trans. on Neural Networks, vol.3, no.1, pp.73–85 (Jan. 1992).

[7] 平井 有三, 落合 辰男, 安永 守利, "1000 ニューロン 100 万シナプスで構成されたニューラルネットワークハードウェアシステム, " 信学論 D-II, vol.J84-D-II, no.6, pp.1185–1193 (Jun. 2001).

[8] M. Marchesi, G. Orlandi, F. Piazza and A. Uncini, "Fast Neural Networks

Without Multipliers," IEEE Trans. on Neural Networks, vol.4, no.1, pp.53–62 (Jan. 1993).

[9] 中條 直也，黒柳 奨，道木 慎二，橋山 智訓，大熊 繁，"FPGA 実装用の反復計算型ニューロン，" 信学技報，NC99-88，2000.

[10] 道木 慎二，大田 祐矢，大熊 繁，"ΔΣ 変調にもとづくパルスニューロンモデル，" 信学技報，NC99-89，2000.

[11] T. kawashima, A. Ishiguro, S. Okuma, "An architecture of Small-Scaled Neuro-Hardware Using Probabilistically coded Pulse Neurons," Electrical Engineering in Japan, Vol.139, No.4, pp.48–55, 2002.

[12] 肥川 宏臣，"ハードウェア化に適した学習機能付き 3 値多層ニューラルネットワーク，" 信学論 D-II，vol.J81-D-II，no.12，pp.2811–2818 (Dec. 1998).

[13] 肥川 宏臣，"多数決ニューロンを用いたパルス駆動型多層ニューラルネットワーク，" 信学技報，ICD98-41，FTS98-41，1998.

[14] 丹 康雄，南谷 崇，"フォールトトレランスを有する階層型ニューラルネットワークとその性質，" 信学論 D-I, Vol.J76-D-I, No.7, pp.380–389, Jul. 1993.

[15] 安永 守利，浅井 光男，柴田 克成，山田 稔，"ニューラルネットワーク集積回路の自律的な欠陥救済能力，" 信学論 D-I, Vol.J75-D-I, No.11, pp.1099–1108, Nov. 1992.

[16] 高浪 五男，堀田 忠義，"階層型ニューラルネットワークの耐故障化 — 値注入学習法と深い学習法 —，" 信学技法 FIIS-03-125, 2003.

[17] M. D. Emmerson and R. I. Damper, "Determining and Improving the Fault Tolerance of Multilayer Perceptions in a Pattern-recognition Application," IEEE Trans. Neural Networks, Vol.14, No.5, pp.788–793, 1993.

[18] D. S. Phatak and I. Koren, "Complete and Partial Fault-Tolerant Neural Networks," IEEE Trans. Neural Networks, Vol.6, No.2, pp.446–456, 1995.

[19] Y. Tohma and Y. Koyanag, "Fault-Tolerant Design of Neural Networks for Solving Optimization Problem," IEEE Trans. Comput., Vol.45, No.12, pp.1450–1455, 1996.

[20] C. P. Fuhrman, S. Chutani, and H. J. Nussbaumer, "Hardware/Software Fault Tolerance with Multiple Task Modular Redundancy," Proc. of Int'l Symp. on Computers and Communications, pp.171–177, 1995.

[21] F. Distante, M. G. Sami, R. Stefanelli and G. Storti Gajani, "Fault Tolerant Characteristics of The Linear Array Architecture for WSI Implementation of Neural Nets," Proc. of IEEE Int'l Conf. on WSI, pp.113–119, 1991.

[22] C. Khunasaraphan, K. Vanapipat, and C. Lursinsap, "Weight Shifting Techniques for Self-Recovery Neural Networks", IEEE Trans. Neural Networks, Vol.5, No.4, pp.651-658, 1994.

[23] D. Simon, "Distributed Fault tolerance in Optimal Interpolative Nets", IEEE Trans. Neural Networks, Vol.12, No.6, pp.1348-1357, 2001.

[24] N. C. Hammadi, T. Ohmameuda, K. Kaneko, and H. Ito, "Fault Tolerant Constructive algorithm for Feedforward Neural Networks", Proc. of Pacific Rim Int'l Symp. on Fault-Tolerant Systems, pp.215-220, 1997.

[25] N. C. Hammadi and H. Ito, "A Learning Algorithm for Fault Tolerant Feedforward Neural Networks," IEICE Transaction on Information and Systems, vol.E80-D, No.1, pp.21–27, Jan. 1997.

[26] U. A. Muller, B. Baumle, P. Kohler, A. Gunzinger, and W. Guggenbuhl, "Achieving Supercomputer Performance for Neural Net Simulation with An Array of Digital Signal Processors", Parallel Computing, Vol.14, No.3, pp.329-346, 1998.

[27] 山森 一人，堀口 進，"並列計算機上の誤差逆伝搬学習法の並列学習モデル，" 信学論 D-II, Vol.J81-D-II, No.2, pp.370–377, Feb. 1998.

[28] K. Yamamori, T. Abe, and S. Horiguchi, "Performance Evaluation of a Partial Retraining Scheme for Defective Multi-Layer Neural Networks," Proc. of the 6th Australasian Conf. on Computer System architecture, pp.138–145, 2001.

[29] W. S. McCulloch and W. A. Pitts, "A Logical Calculus of the Ideas Immanent in Neural Nets," Bull. Math. Biophys., Vol.5, pp.115–133, 1943.

[30] D. E. Goldberg, "Genetic Algorithms in Search, Optimization, and Machine Learning", Addison-Wesley, 1989.

[31] G. R. Harik, F. G. Lobo, and D. E. Goldberg, "The compact genetic algorithm", Technical Report 97006, IlliGAL Report, 1997.

[32] C. Aporntewan and P. Chongstitvatana, "A Hardware Implementation of the Compact Genetc Algorithm", Proc. of the 2001 IEEE Congress on Evolutionary Computation, pp.624–629, 2001.

[33] D. Whitley and T. Hanson, "Optimizing Neural Network Using Faster, More Accurate Genetic Search", Proc. of Int'l Conf. on Genetic Algorithms (ICGA-89), pp.391-396, 1989.

[34] D. Montana and L. Davis, "Training Feedforward Neural Networks Using Genetic Algorithms", Proc. of Int'l Joint Conf.on Artificial Intelligence(IJCAI-89), pp.762-767, 1989.

[35] H. Kitano, "Empirical Studies on the Speed of Convergence of Neural Network Training using Genetic Algorithms", Proc. of National Conf. on Artificial Intelligence, Vol.2, pp.789-795, 1990.

[36] J. David Schaffer, Darrell Whitley, Larry J. Eshelman, "Combinations of Genetic Algorithms and Neural Networks: A Survey of the State of the Art," Proc. of Int'l Workshop on Combinations of Genetic Algorithms and Neural Networks, pp.1–37, 1992.

[37] D. Whitley, T. Starweather and C. Bogart, "Genetic algorithms and neural networks: optimizing connections and connectivity," Parallel Computing, Vol.14, pp.347–361, 1990.

[38] G. F. Miller, P. M. Todd, "Designing Neural Networks using Genetic Algorithms," Proc. of ICGA'89, pp.379–384, 1989.

[39] U. Seiffert, "Multiple Layer Perceptron Training Using Genetic Algorithms," Proc. of European Symposium on Artificial Neural Networks, pp.159–164, 2001.

[40] H. Kitano "Designing Neural Networks Using Genetic Algorithms with Graph Generation System," Complex Systems, Vol.4, pp.461–476, 1990.

[41] H. Kitano, "Neurogenetic learning: an integrated method of designing and training neural networks using genetic algorithms," Physica D, Vol.75, pp.225-238, 1994.

[42] M. Murakawa, S. Yoshizawa, I. Kajitani, X. Yao, N. Kajihara, M. Iwata, and T. Higuchi, "The GRD Chip: Genetic Reconfiguration of DSPs for Neural Network Processing", IEEE Trans. Comput., Vol.48, No.6, pp.628-638, 1999.

[43] C. P. Low, "An Efficient Reconfiguration Algorithm for Degradable VLSI/WSI Arrays", IEEE Trans. Comput., Vol.4, No.6, pp.553-559, 2000.

査読付き論文

1. 菅原 英子，堀口 進，“階層型ニューラルネットワークのニューロン故障補償手法の実装”，電子情報通信学会論文誌 C， Vol.J85-C，No.9，pp.861-864，Sep. 2002.
2. Eiko Sugawara, Masaru Fukushi, Susumu Horiguchi, “Self-reconfigurable Multilayer Neural Networks with Genetic Algorithms”, IEICE Trans. on Information Systems, Vol.E87-D, No.8, pp.2021-2028, Aug. 2004.

国際会議発表論文

1. Eiko Sugawara, Masaru Fukushi, Susumu Horiguchi, “Fault Tolerant Multilayer Neural Networks with GA Training”, The 18th IEEE International Symposium on Defect and Fault Tolerance in VLSI systems, pp.328-335, Nov. 2003.

研究会・口頭発表

1. 菅原 英子，阿部 亨，堀口 進，“ハードウェア化に適した階層型ニューラルネットワークの構成法”，平成 12 年度電気関係学会北陸支部連合大会 公演論文集，pp.244，Sep. 2000.
2. 菅原 英子，堀口 進，“故障補償能力を持つニューラルネットワークの実装”，電子情報通信学会 2001 年総合大会 公演論文集，情報・システム 1，pp.153，March 2001.
3. 菅原 英子，堀口 進，“故障補償能力を持つ階層型ニューラルネットワークのハードウェア実装”，並列処理シンポジウム JSPP2001 論文集，pp.319-326，Jun. 2001.
4. 菅原 英子，堀口 進，“FPGA による階層型ニューラルネットワーク故障補償のハードウェア化”，電子情報通信学会 第 15 回 機能集積情報システム研究会 FIIS-01-86，Jun. 2001.