自律故障補償可能なニューロンのフォワード・パス部の設計とFPGA上への実装

山森一人[1)] 草野真道[2)] 吉原郁夫[3)]

Design and Implementation of Self Defect Compensatable Neuron on FPGA device

Kunihito YAMAMORI[1)] Masamiti Kusano[2)] Ikuo YOSHIHARA[3)]

Abstract

Larning time of large-scale neural network is a serious problem. To achieve fast learning, some researchers proposed to implement a neural networks into Wafer Scale Integration(WSI). When neural networks is implemented into a WSI, it has to have a mechanism to compensate defect or fault. Partial Retraining(PR) scheme was proposed as one of the methods of compensating for hardware defects. However, PR scheme does not evaluate its performance on the hardware still yet. In this paper, we implement a forward pass of PR scheme into a Field Programmable Gate Array(FPGA) and test its functions.

Key Words:
Partial Retraining scheme, FPGA, neural network, fault-tolerance

1 はじめに

ニューラルネットワークはパターン認識などの問題に対する解法の一つであるが、現実の問題を解こうとした場合、学習に膨大な時間がかかるという問題がある。この問題を解決する手法の一つとして、目的とする機能を持った回路をシリコンウェーハ上に直接構築することで高速化を図るWSI(Wafer Scale Integration) が注目されている。しかし、WSI ではシリコンウェーハ上に直接回路を構築するため、回路上の故障が目的の機能に対して重大な影響を与えるという問題がある。したがって WSI 上にニューラルネットワークを実装する場合は、故障を補償する何らかの仕組みも実装する必要がある。

具体的な故障補償法としては、あらかじめ故障を想定して学習を行っておく FTBP 法 [1)] や、故障発生時に誤差逆伝搬学習法 (BP 法) を使用し再学習を行うことで結合荷重の調整を行う方法、山森ら [2, 3, 4)] が提案した、BP 法による学習の対象を故障の影響を受けるニューロンに限定することで学習の高速化を図る部分再学習法 (PR 法) などがある。これらの手法は多くがソフトウェアを用いたシミュレーションによる検証に留まっており、実際にシリコンウェーハ上にニューラルネットワークを構築した場合の故障補償機能を検証していない。本研究では、実際にシリコンウェーハ上で PR 法の故障補償機能を検証する前段階として、動的に回路構成が変更可能な FPGA(Filed Programmable Gate Arry) 上に再学習可能なニューロンのフォワード・パス部を実装し、その動作を検証することを目的とする。

‥情報システム工学科准教授
‥情報システム工学科 (現、NTT ネオメイト)
‥情報システム工学科教授

2 ニューラルネットワークにおける故障補償法

2.1 誤差逆伝搬学習法

本研究ではニューラルネットワークとして図 1 のような階層型ネットワークを対象とした。階層

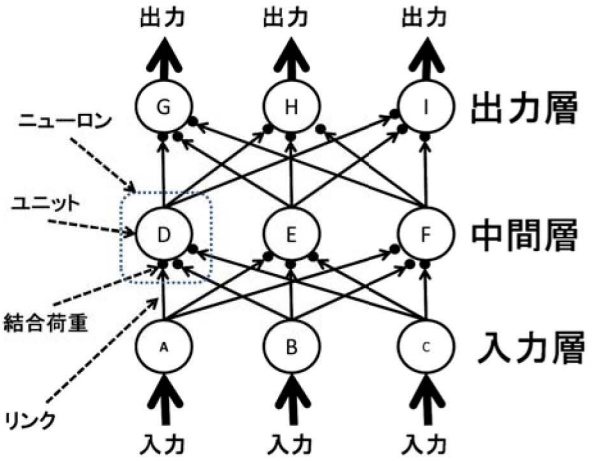

図1 階層型ニューラルネットワークの例

型ニューラルネットワークにおいて、各階層間のニューロンは重み付きのリンクにより全結合しており、階層内のニューロン間には結合がない。ある学習パターン p を入力した時の第 k 層のニューロン n における入出力 $i_n^k(p)$ 、$o_n^k(p)$ を (1) 式、(2) 式にそれぞれ示す。

$$i_n^k(p) = \sum_{m=1}^{N_{k-\bullet}} o_n^{k\bullet\ 1}(p) w_{m,n}(p), \quad (1)$$

$$o_n^k(p) = f(i_n^k(p)). \quad (2)$$

(1) 式において $N_{k\bullet\ 1}$ は第 $(k-1)$ 層のニューロンの数を表し、$w_{m,n}(p)$ は第 k 層のあるニューロン n と第 $(k-1)$ 層のあるニューロン m との結合荷重を表す。また、(2) 式において f は活性化関数を表す。活性化関数 f には (3) 式のシグモイド関数が一般に用いられる。

$$f(x) = \frac{1}{1+e^{\bullet\ x}}. \quad (3)$$

出力層 Z のニューロン l の教師信号 $t_l^Z(p)$ と出力信号 $o_l^Z(p)$ の二乗誤差 $E_l^Z(p)$ は (4) 式で表わされ、$E_l^Z(p)$ が最小になるよう最急降下法により結合荷重 $w_{m,n}(p)$ の調整を行う。

$$E_l^Z(p) = \frac{1}{2}\sum_l (t_l^Z(p) - o_l^Z(p))^2. \quad (4)$$

結合荷重の修正量を $\Delta w_{m,n}(p)$、学習係数を ε とすると $\Delta w_{m,n}(p)$ は (5) 式で表わされ、結合荷重の修正は (6) 式に従って行われる。

$$\Delta w_{m,n}(p) = -\varepsilon \delta_n^k o_m^{k\bullet\ 1}(p), \quad (5)$$

$$\delta_l^Z = (t_l^z(p) - o_l^z(p)) f'(i_l^z(p)),$$

$$\delta_m^{k\bullet\ 1} = f'(i_m^{k\bullet\ 1}(p)) \sum_{n=1}^{N_k} w_{m,n}(p)\delta_n^k,$$

$$w_{m,n}(p+1) = w_{m,n}(p) + \Delta w_{m,n}(p). \quad (6)$$

また、学習の収束を安定させるために (5) 式の代わりに (7) 式を用いる方法もある。

$$\Delta w_{m,n}(p) = \alpha \Delta w_{m,n}(p-1) - \varepsilon \delta_n^k o_m^{k\bullet\ 1}(p). \quad (7)$$

このとき α は慣性項と呼ばれる定数を表している。

2.2 対象とする故障とその判定

ニューラルネットワークは図1に示したようにニューロンごとのユニットとリンク、及びユニット間の結合荷重から構成され、これらが故障の発生しうる箇所である。

本研究ではユニット、リンク、結合荷重の故障としてそれぞれの出力、入力、結合荷重が0に固定される0-スタック故障が発生すると仮定した。このとき、ユニットでの故障は故障が発生したユニットの全出力リンクにおいて故障が発生した場合と等価であり、結合荷重での故障は当該荷重に対応するリンクの故障と等価であることから、故障の対象としてリンクの故障を想定した。

学習済みのニューラルネットワークにおいて故障が発生した場合、出力信号 $o_l^Z(p)$ と教師信号 $t_l^Z(p)$ の間で誤差が発生する。実際の出力 $o_l^Z(p)$ と教師信号 $t_l^Z(p)$ 間の許容誤差を η としたとき、(8) 式を満たさない出力ニューロンが存在することをニューラルネットワークの故障と定義する。

$$\sqrt{(t_l^Z(p) - o_l^Z(p))^2} < \eta. \quad (8)$$

2.3 部分再学習法 (PR 法)

本研究では故障補償法として部分再学習法 (PR 法) を想定した。PR 法では最初に BP 法による初期学習を行う。このとき各ニューロンは学習パターンごとの入力や出力、結合荷重をニューロンが持つメモリに保存する。学習終了後、一定周期ごとに初期学習時に使用した学習パターンを入力し、各ニューロンにおいて (8) 式の計算を行うことで故障の有無を調べる。

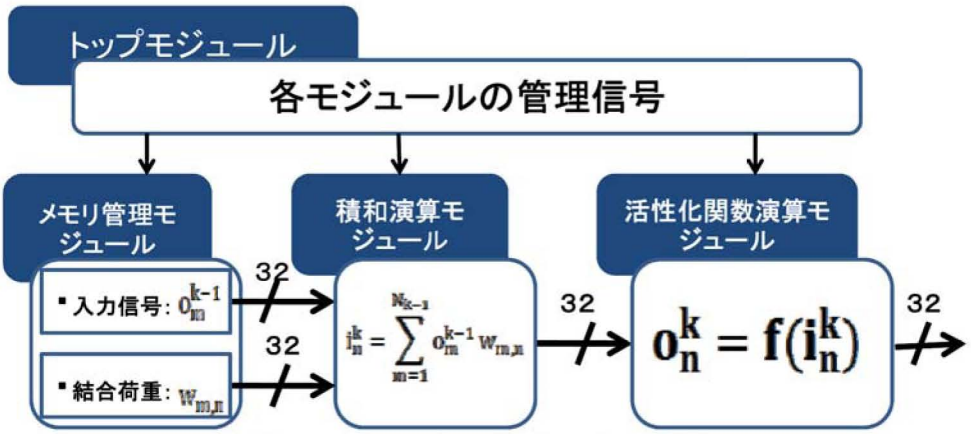

図2　フォワード・パス部のブロック図

図1において中間層のニューロンEと出力層のニューロンH間のリンクで故障が発生した場合を考える。このとき、故障の影響を受けるのは出力層のニューロンHのみである。そこでニューロン{D、E、F、H}からなる部分ネットワークにおいてBP法による学習を行うことで故障を補償する。また、このときニューロンHへの入力信号として、中間層のニューロンが持つ故障発生前の出力を使用し、ニューロンHの持つ教師信号に近づくよう重みを調整する。

図1において入力層のニューロンBと中間層のニューロンEの間のリンクで故障が発生した場合を考える。このとき、中間層のニューロンE及び出力層のニューロンG、H、Iが故障の影響を受ける。はじめに入力層、中間層のニューロン{A、B、C、E}からなる部分ネットワークにおいて部分再学習を行い、次いで中間層-出力層間のニューロン{D、E、F、G}、{D、E、F、H}、{D、E、F、I}からなる3個の部分ネットワークで並列に部分再学習を行う。

3　設計の方針

3.1　VHDLによる再学習可能なニューロンの設計

本研究ではハードウェア記述言語としてVHDLを使い、FPGA上にPR法におけるニューロンのフォワード・パス部のコーディングを行った。図2に、設計したニューロンのフォワード・パス部のブロック図を示す。

図2におけるトップモジュールは他のモジュールの状態を管理する。メモリ管理モジュールはニューロンの入力信号と結合荷重のメモリへの読み書きを行う。積和演算モジュールにはメモリから読み込んだ入力信号と結合荷重が入力され、それらの積を乗算器により演算し、その結果とそれまでの累積和を加算器により演算することで(1)式の計算を行い、その結果を出力する。活性化関数演算モジュールでは積和演算モジュールの結果を入力として、(2)式の計算を行う。

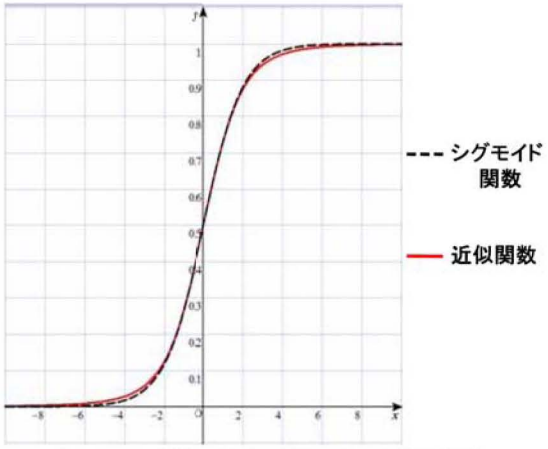

図3　シグモイド関数とその近似関数

入力信号、結合荷重及び各演算モジュールの入出力信号は、最上位ビットを符号ビットとし、バイアス値が127で8bitの指数部及び23bitの仮数部からなるIEEE754形式の32ビット二進化浮動小数点数形式で表現した。また、各モジュールで使用する演算器はXilinx社が提供するIPコア作成ツールを用いて生成した。

3.2　活性化関数の近似

ニューラルネットワークの出力の計算を行う際に用いられる関数を活性化関数と呼び、(3)式で示したシグモイド関数が一般に用いられている。しかし、VHDLでは指数関数を直接実装する事が困難であるため、活性化関数としてシグモイド関

数の近似関数である (9) 式を使用した。

$$f(x)=\begin{cases} \dfrac{1}{2^{\bullet}\ x+\frac{x^{\bullet}}{\cdot}^{\bullet}\ \frac{x^{\bullet}}{\cdot}+\frac{x^{\bullet}}{\cdot\cdot}} & (x<0) \\ 1-\dfrac{1}{2^{\bullet}\ (^{\bullet}\ x)+\frac{\cdot -x^{\bullet}}{\cdot}^{\bullet}\ \frac{\cdot -x^{\bullet}}{\cdot}+\frac{\cdot -x^{\bullet}}{\cdot\cdot}} & (x\geq 0) \end{cases} \tag{9}$$

(9) 式は、入力 x が負の場合はシグモイド関数における $e^{\bullet\ x}$ を 4 次の項までテーラー展開したものであり、入力 x が正の場合は、x が負の場合の近似関数を点 $(0, 0.5)$ を中心として 180 度回転させたものである。

図 3 にシグモイド関数、及び (9) 式で示した近似関数の波形をそれぞれ示す。シグモイド関数と近似関数での誤差は小さく、シグモイド関数の近似関数として使用する上で問題ない近似となっている

3.3 トップモジュール

トップモジュールでは各モジュールの管理とデータの橋渡しを行う。また、各モジュールは処理開始フラグとモジュールの動作状況を表す状態信号を持っており、トップモジュールで処理開始フラグを '1' にすることで処理を開始し、その後、各モジュールで状態信号が待機状態になることで処理の終了を検知して状態遷移を進める。状態遷移の流れは以下の通りである。

DEFAULT 待機状態。ボード上にあらかじめ設定しておいたスタートスイッチが押されることで STEP0 へ状態遷移し処理を開始する。

STEP0 入力信号及び結合荷重の初期値をメモリに書き込む処理を行う。データ書き込み終了後 STEP1 へ状態遷移する。

STEP1 STEP0 で書き込んだ入力及び結合荷重のデータの一部をメモリから読み込こむ。データ読み込み終了後 STEP2 へ状態遷移する。

STEP2 積和演算モジュールへデータを渡し、積和演算を行う。演算終了後、STEP1 で最後のデータまで読み込みが終了していた場合、STEP3 へ状態遷移する。それ以外の場合は STEP1 へ状態遷移する。

STEP3 活性化関数演算モジュールへデータを渡し活性化関数演算を行う。演算終了後 STEP4 へ状態遷移する。

STEP4 拡張機能用予備状態。STEP5 へ状態遷移する。

STEP5 拡張機能用予備状態。STEP6 へ状態遷移する。

STEP6 あらかじめ調べておいた計算結果と今回の計算結果との比較を行う。比較終了後、DEFAULT へ状態遷移する。

3.4 メモリ (SDRAM) 管理モジュール

メモリ管理モジュールではトップモジュールからの処理開始フラグ及び読み書き命令を受けて、SDRAM に対して入力データ、各ニューロンの出力及び結合荷重のデータの読み書きを行う。データの読み書きを行う場合、ボード上の SDRAM は 16 ビット ×4M アドレスの構成であることから 1 アドレスあたりに保存できるデータは 16 ビットとなる。したがって、32 ビットで表現されるデータを読み書きするには、2 アドレスに分けて格納されているデータを結合・分割する必要がある。読み込みを行う場合は 32 ビットのデータを 16 ビットづつ 2 回に分けてトップモジュールに渡し、その際にトップモジュールでデータの結合を行う。書き込みを行う場合は 32bit のデータをトップモジュールから受け取り、メモリ管理モジュール内でデータを分割して書き込みを行う。なお、データはビッグエンディアンでメモリ上に格納することとした。

また、SDRAM は揮発性メモリなので一定時間ごとにリフレッシュを行う必要がある。そのためメモリが各処理を行う際に、一定時間が経過すると自動でリフレッシュを行う。

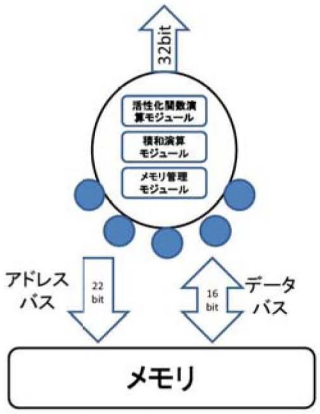

図 4 設計したニューロン

活性化関数演算モジュールへの入力 (sig_in)	2進数	0.609375
	10進数	00111111000111000000000000000000
出力 (sig_out)	2進数	0.647702
	10進数	00111111001001011100111111000111

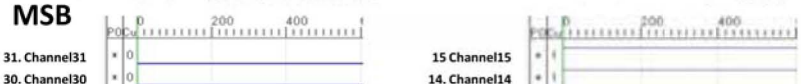

図 5　無故障時の活性化関数演算モジュールのシミュレーション波形

図 6　無故障時の活性化関数演算モジュールの観測波形

3.5　積和演算モジュール

積和演算モジュールではトップモジュールからの処理開始フラグ及び入力信号、結合荷重のデータを受けて、(1) 式の計算を行う。設計した回路では、あらかじめ累積和として初期値を $(0)_{10}$ としたラッチを用意しておき、入力信号と結合荷重の乗算が行われる度に結果をそのラッチに加算していくことで Σ の計算を実現した。

3.6　活性化関数演算モジュール

活性化関数演算モジュールではトップモジュールからの処理開始フラグ及び結合荷重計算の結果 x を受けて、(2) 式の計算を行う。実際には、(2) 式を多項式で近似した (9) 式の計算を行う。

また、活性化関数の近似計算を行う際に、$x > 0$ における (9) 式は、$x < 0$ における (9) 式を 1 から減じたものと等価であるため、途中計算時に同一の回路を使用することで回路規模を小さくしている。

4　FPGA 上への実装と動作検証

4.1　実験環境

回路設計と合成、シミュレーション、及び実装対象の FPGA には以下を使用した。

回路合成ツール :Xilinx ISE10.1.03

シミュレーションツール :ModelSim SE 6.3e

評価ボード :TD-BD-TS-101

FPGA :Xilinx Spartan3 XC3S400-4
(40 万ゲート相当)

ロジックアナライザ :Sophia Systems LA800

使用言語 :VHDL

4.2　動作確認

動作確認を行うために図 4 に示す 5 つの入力リンクを持つニューロンを設計し、FPGA 上にコンフィグレーションして出力の検証を行った。このニューロンに与えた入力を (10) 式、結合荷重を (11) 式にそれぞれ示す。

$$input = \{2.000000, -0.750000, 4.125000, -3.250000, -1.750000\}, \quad (10)$$

$$weight\_in = \{0.500000, 0.250000, -0.125000, -0.500000, 0.750000\}. \quad (11)$$

また、故障が発生した場合の動作例として、このニューロンの持つ 5 つの入力リンクの内の 1 つが故障の影響を受け、(10) 式に示したリンクの出力が (12) 式に示すように変化した場合についても出力の検証を行った。

$$input = \{2.000000, -0.750000, 4.125000, -3.250000, 0.000000\}. \quad (12)$$

これらの場合において、各演算モジュールで FPGA ボード上で実際に観測した波形とシミュレーショ

活性化関数演算モジュールへの入力 (sig_in)	2進数	1.921875
	10進数	00111111111101100000000000000000
出力 (sig_out)	2進数	0.867025
	10進数	00111111010111011111010101100000

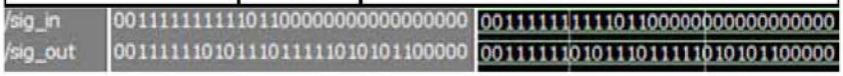

図 7　故障時の活性化関数演算モジュールのシミュレーション波形

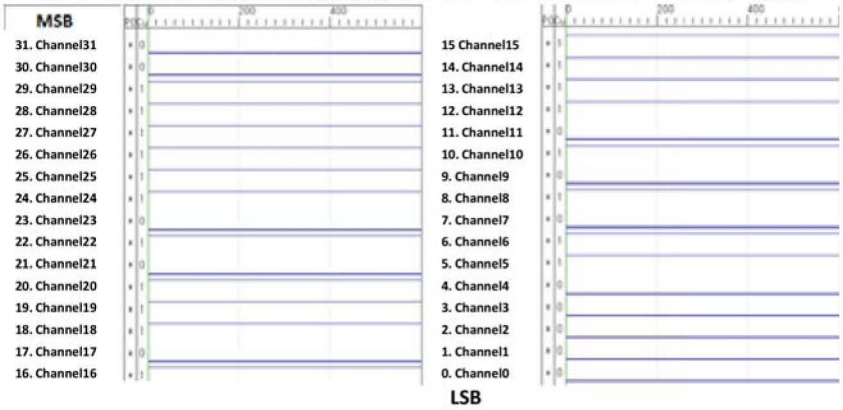

図 8　故障時の活性化関数演算モジュールの観測波形

ン波形の比較を行った。図 5 に無故障時のシミュレーション出力を、図 6 にはロジックアナライザで観測した FPGA からの出力波形をそれぞれ示す。また、図 7 に故障発生時のシミュレーション出力を、図 8 にその時の FPGA 上での観測波形を示す。図 5、図 7 中の sig_in は活性化関数演算モジュールへの入力を、sig_out は活性化関数演算モジュールの出力を表しており、sig_out が設計した回路の最終的な出力となっている。各演算モジュールでロジックアナライザを用いて実際に観測した出力とシミュレーション出力は一致しており、作成した回路が FPGA 上で正しく動作することを確認した。

5　おわりに

本研究では、故障補償法として PR 法を組み込んだニューラルネットワークをシリコンウェーハ上に構築する前段階として、ニューロンのフォワード・パス部を設計し、FPGA 上へ実装することを目的とした。

作成したプログラムは回路合成ツールを使用して論理合成し、生成された論理回路の FPGA へのコンフィグレーションを行った。作成した回路は FPGA 上のフリップフロップの 71%、4 入力の組み合わせ回路の 63%を使用することが分かった。

また、生成された回路の FPGA 上での各演算モジュールの出力はシミュレーション出力と一致し、正しく動作することをロジックアナライザにより確認した。

今後の課題としては故障判定及び結合荷重の修正を行う機能を備えたモジュールについて FPGA 上への実装を行うことである。また、これらのモジュールを加えた上で FPGA 上に回路を構築する場合、回路規模が大きすぎて今回使用した FPGA ボードでは回路が構築できないことが考えられる。そのためより大規模な FPGA を持ったボード上でプログラムを動かせるようプログラムの変更を行うことが必要である。

参考文献

[1] 高浪五男, 楊雲平, 堀口進, “重み故障にフォールトトレラントな階層型ニューラルネットワークの構築”, 信学技報 (FTS, フォールトトレラントシステム), Vol. 98, No. 68, pp.47–52 (1998).

[2] 山森一人, 阿部亨, 堀口進, “階層型ニューラルネットワークの部分再学習法による高速な故障補償”, 信学技報 (NC, ニューロコンピューティング), Vol. 98, No.128, pp.79–86 (1998).

[3] 山森一人, 阿部亨, 堀口進, “部分再学習法による故障補償を行ったニューラルネットワークの汎化能力”, 機能集積情報システム研究会発表論文集 III, pp.108–115 (1998).

[4] 山森一人, 阿部亨, 堀口進, “複合故障を持つ階層型ニューラルネットワークにおける部分再学習法の性能評価”, 機能集積情報システム研究会発表論文集 III, pp.183–191 (2001).