自律故障補償可能なハードウェアニューロンのバックワードパス演算部の設計

山森 一人[1] 田代 圭佑[2] 吉原 郁夫[3]

Design of Backward-pass Module for Self-defect-compensatable Hardware Neuron

Kunihito YAMAMORI[1] Keisuke TASHIRO[2] Ikuo YOSHIHARA[3]

Abstract

In neural network, there is a problem that learning time becomes so long for real world problems. To achieve fast learning, some researchers proposed to implement a neural network into a Wafer Scale Integration (WSI). WSI uses one wafer as a parallel computer, a part of defect leads entire system fault. Therefore a defect compensation method is necessary to implement a neural network into WSI. Partial Retraining (PR) scheme has proposed as one of the defect compensation methods for neural networks. However, PR scheme is not verified whether it will perform good on hardware devices or not. It is also not clear how much is circuit required. In our laboratory, we have been finished a design of forward-pass module of self-defect-compensatable hardware neuron. In this paper, we report a design of backward-pass module, and evaluate the performance of our design by simulations.

Key Words:
Partial retraining scheme, Neural network, Defect compensation, Field programmable gate array

1. はじめに

ニューラルネットワークはパターン認識や最適化問題などの有力な解法の１つである．しかし，現実規模のこれらの問題の多くはデータ量が非常に多く，学習時間が膨大になるという問題がある．

この問題の解決策として，シリコンウェーハ上に回路を実装し，これをそのまま並列計算機として利用して高速化を図るWSI(Wafer Scale Integration)が注目されている．ANN(Artificial Neural Network)はニューロン１つの規模が比較的小さく，また多数のニューロンを用いるのでWSI実装に適しており，Schemmelら[1]による実装例が報告されている．しかしWSIは一枚のシリコンウェーハをそのまま回路として利用するため，一箇所でも故障箇所が存在すると回路全体の故障となる．従ってWSIを実現する場合には，何らかの故障補償手段が必要となる．

WSI実装を前提としたハードウェアニューラルネットワークの故障補償法には，いくつかの手法が提案されている．具体的な手法として，山森ら[2]が提案した，再学習する範囲を故障の影響を受けるニューロン部分に限定することで高速化を図る部分再学習法(PR法)や，菅原ら[3]が提案した，予備ニューロンを用いて故障箇所を切り換える機能シフト法がある．またKhunasaraphanら[4]は，再学習によらない結合荷重更新の手法を提案した．

これらの手法の多くはソフトウェアでのシミュレーションによる検証のみに留まっており，シリコンウェーハ上へニューラルネットワークを実装した場合の故障補償能力を検証していない．本研究では，草野ら[5]によって設計されたハードウェアニューロンをベースに故障検知機能と故障補償機能を加え，自律故障補償

[1] 情報システム工学科准教授
[2] 情報システム工学科(現在，株式会社IIM)
[3] 情報システム工学科教授

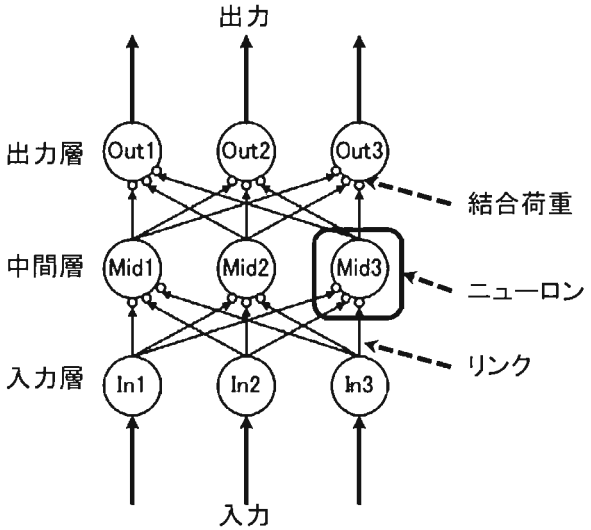

図１　階層型ニューラルネットワークの例

可能なハードウェアニューロンを設計することを目的とする．

2. ニューラルネットワークにおける故障補償

2.1 誤差逆伝播法

階層型ニューラルネットワークの例を図１に示す．図１で各階層のニューロンは結合荷重とリンクを介して全結合しており，同一階層内のニューロン間の結合はない．

ある学習パターンpを入力したときの第k層におけるニューロンの入出力$i^k(p), o^k(p)$を(1)式，(2)式にそれぞれ示す．

$$i_{p,j}^{k}(p) = \sum w_{i,j}^{k-1,k}(p) o_{p,i}^{k-1}(p), \tag{1}$$

$$o_{p,j}^{k}(p) = f_j^k\left(i_{p,j}^{k}(p)\right), \tag{2}$$

$w_{i,j}^{k-1,k}$は第k層j番目のニューロンと第$(k-1)$層i番目のニューロンとの結合荷重を表し，f_j^kは活性化関数を表す．活性化関数には(3)式のシグモイド関数が一般に用いられる．

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

教師信号t^kと出力信号o^kの二乗誤差$E^k(p)$は(4)式で示され，$E^k(p)$が最小となるように最急降下法によって結合荷重を調整する．

$$E^k(p) = \frac{1}{2}\sum\left(t^k - o^k\right)^2. \tag{4}$$

本研究では故障補償法として2.2節で述べるPR法を採用しているため，任意の結合荷重の状態から$E^k(p)$の極小値に達するための結合荷重修正量Δwは全てのニューロンについて(5)式で求められる．

$$\Delta w_{i,j}^{k-1,k} = o_{p,j}^{k} \times f'\left(i_{p,j}^{k}\right) \times \left(o_{p,j}^{k} - t_{p,j}^{k}\right). \tag{5}$$

2.2 部分再学習法

本研究では故障補償機能として部分再学習法(PR法)[2]を採用する．PR法は結合荷重の調整を故障の影響を受けるニューロンのみに限定することで学習の高速化を図るものである．

PR法はまず故障のない状態でBP法による初期学習を行い，各ニューロンの入力信号、出力信号、結合荷重などを保存しておく．運用時には初期学習時の学習パターンを適当に入力し，中間層と出力層ニューロンの出力を，それぞれに保存してある無故障時の出力や教師信号と比較し，故障発生の有無を調べる．故障が確認された場合，故障の影響を受ける範囲のニューロンは保存してある値に近づくよう結合荷重の修正を行う．

図１で入力層ニューロンIn2と中間層ニューロンMid2間の入力リンクに故障が発生した場合を考える．この時，故障の影響を受けるのは中間層のMid2と出力層のOut1, Out2, Out3の４つのニューロンである．そこで，まず入力層と中間層のニューロン(Mid2, In1, In2, In3)からなる部分的なネットワークでBP法による再学習を行い，故障の補償を行う．その後，故障の影響を受けているニューロンの集合である(Out1, Mid2, Mid2, Mid3)，(Out2, Mid1, Mid2, Mid3)，(Out3, Mid1, Mid2, Mid3)からなる部分ネットワークにおいてBP法による再学習を行い，故障の補償を行う．

2.3 対象とする故障

階層型ニューラルネットワークは図１に示した通りニューロンとリンク，結合荷重から構成され，これらが故障発生箇所となりえる．本研究ではリンクが断線し，そのリンクからの入力が常に０に固定される0-スタック故障を想定する．これは，ニューロンの故障はニューロンからのすべての出力リンクの故障と等価であり，また結合荷重の故障は対応するリンクの故障と等価であるためである．故障が発生した場合，出力信号と教師信号や故障以前の出力値との間に誤差が生じる．初期学習に用いた学習パターンが入力された時，出力信号と保存してある教師信号，または無故障時の出力信号の値が一致しないニューロンが存在する場合をニューロンの故障と定義する．

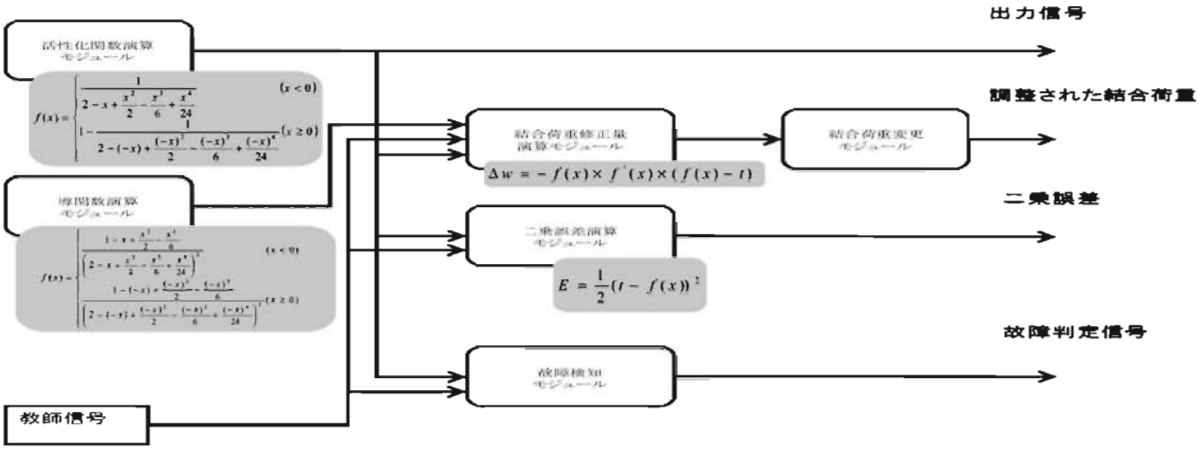

図 2　バックワード演算部のブロック図

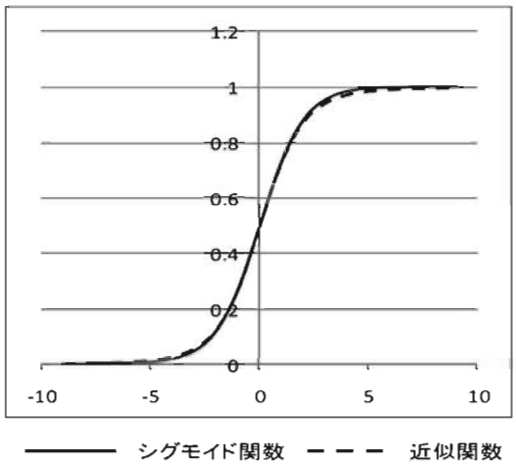

図 3　シグモイド関数とその近似式

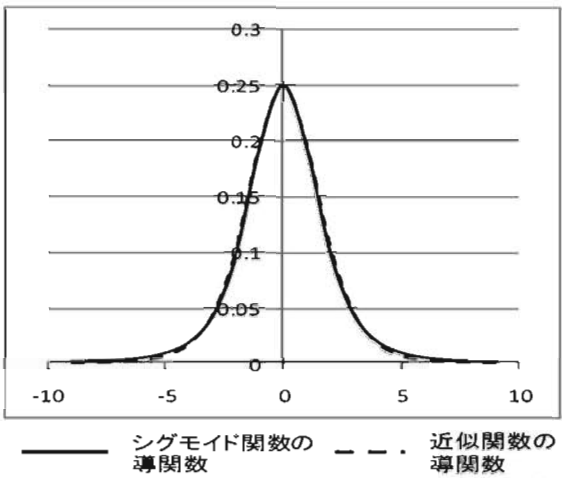

図 4　シグモイド関数の導関数とその近似式

3. バックワードパス演算部の設計

3.1 全体のモジュール構成

本研究では，草野らによる実装をベースに PR 法を前提としたハードウェアニューロンのバックワードパス演算部のコーディングを行う．図 2 に設計したバックワードパス演算部のブロック図を示す．設計する部分は大きく分けて故障検知と結合荷重修正演算である．次節で活性化関数の近似について説明した後，バックワードパス部の各状態について順に説明する．

3.2 活性化関数とその導関数の近似

図 2 の活性化関数演算モジュールと導関数演算モジュールは，入力と結合荷重の積和を入力信号として，シグモイド関数とその導関数を近似した(6)式，(7)式の計算を行う．

$$f(x)=\begin{cases}\dfrac{1}{2-x+\frac{x^2}{2}-\frac{x^3}{6}+\frac{x^4}{24}} & (x<0),\\ 1-\dfrac{1}{2-(-x)+\frac{(-x)^2}{2}-\frac{(-x)^3}{6}+\frac{(-x)^4}{24}} & (x\geq 0).\end{cases} \tag{6}$$

$$f'(x)=\begin{cases}\dfrac{1-x+\frac{x^2}{2}-\frac{x^3}{6}}{\left(2-x+\frac{x^2}{2}-\frac{x^3}{6}+\frac{x^4}{24}\right)^2} & (x<0),\\ \dfrac{1-(-x)+\frac{(-x)^2}{2}-\frac{(-x)^3}{6}}{\left(2-(-x)-\frac{(-x)^2}{2}-\frac{(-x)^3}{6}+\frac{(-x)^4}{24}\right)^2} & (x\geq 0).\end{cases} \tag{7}$$

(6)式，(7)式は指数関数をテイラー展開し，シグモイド関数の対称性を利用して近似性を高めたものである．図 3 にシグモイド関数とその近似式，図 4 に導関数とその近似式の波形をそれぞれ示す．シグモイド関数と近似式の誤差は，図 3 で最大 0.0107，図 4 で最大 0.0061 と小さく，(6)式，(7)式はシグモイド関数とその導関数の近似式として使用する上で十分な近似となっている．

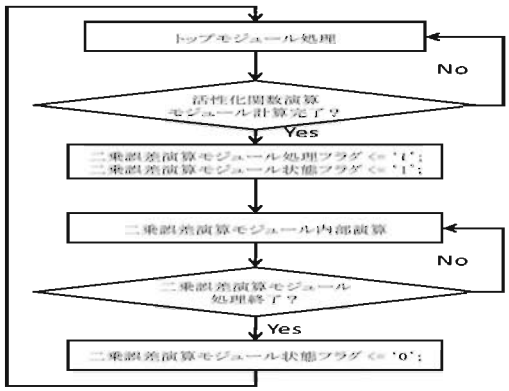

図5　二乗誤差演算モジュールフローチャート

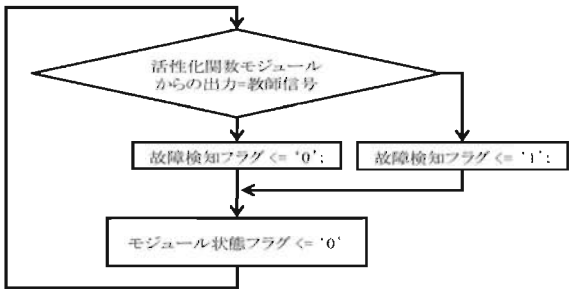

図6　故障検知モジュールフローチャート

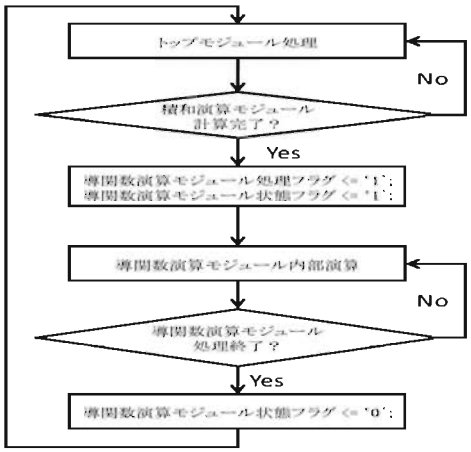

図7　導関数演算モジュールフローチャート

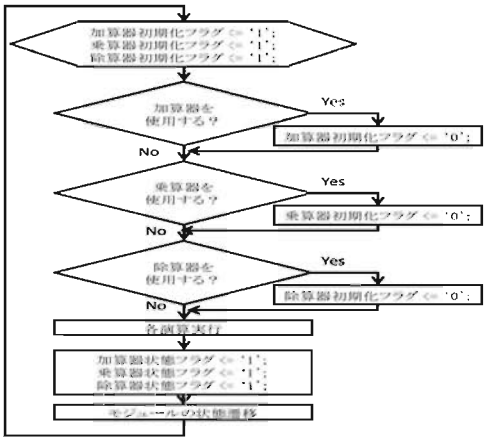

図8　各演算器フローチャート

3.3 二乗誤差演算モジュール

二乗誤差演算モジュールでは，活性化関数から出力された信号とあらかじめ保存されていた教師信号との誤差の計算を行う．図5に二乗誤差演算モジュールのフローチャートを示す．処理開始フラグが‘1’になると活性化関数の出力oと教師信号tを受け取り，状態フラグを‘1’にして処理を開始する．

計算が完了するとトップモジュールに対して結果を出力し，状態フラグ‘0’をトップモジュールへ返す．

3.4 故障検知モジュール

故障検知モジュールでは，活性化関数からの出力信号と，予め保存されていた教師信号や故障前の出力値との比較を行う．図6に処理の流れを示す．処理開始フラグが‘1’になりトップモジュールから活性化関数演算モジュール出力と教師信号を受け取ることで処理を開始する．教師信号と出力信号が一致した場合は故障検知フラグ‘0’を，一致しなければ‘1’をトップモジュールへ出力する．故障検知フラグをトップモジュールへ出力したのち，故障検知モジュールの状態フラグ‘0’をトップモジュールへ返す．

3.5 導関数演算モジュール

導関数演算モジュールでは積和演算モジュールから出力された値xを入力として，(7)式の演算を行う．図7に導関数演算モジュールのフローチャート，図8に各演算器使用時のフローチャートを示す．導関数演算モジュール処理フラグが‘1’になり積和演算モジュールからの出力を受け取ると処理を開始し，導関数演算モジュールの状態フラグ‘1’をトップモジュールへと返す．

また各状態で使用される演算器は各演算器の初期化フラグを‘0’にすることで演算を開始し，演算器の状態フラグは同時に‘0’になる．演算が終了すると状態フラグは‘1’になり，各状態結果を出力させたのちに初期化フラグを‘1’にして演算器の初期化を行う．演算器の状態フラグは本モジュール全体の状態遷移トリガーとなっているため，各演算器が演算終了後に状態フラグ‘1’を返すことによって導関数演算モジュールの状態遷移を行う．計算が完了すると計算結果をトップモジュールへ出力し，導関数演算モジュールの状態フラグが‘0’になる．

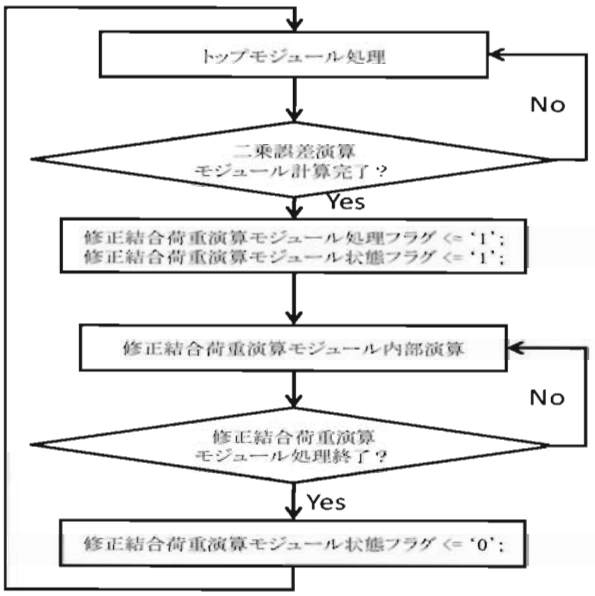

図9　結合荷重修正量演算モジュールフローチャート

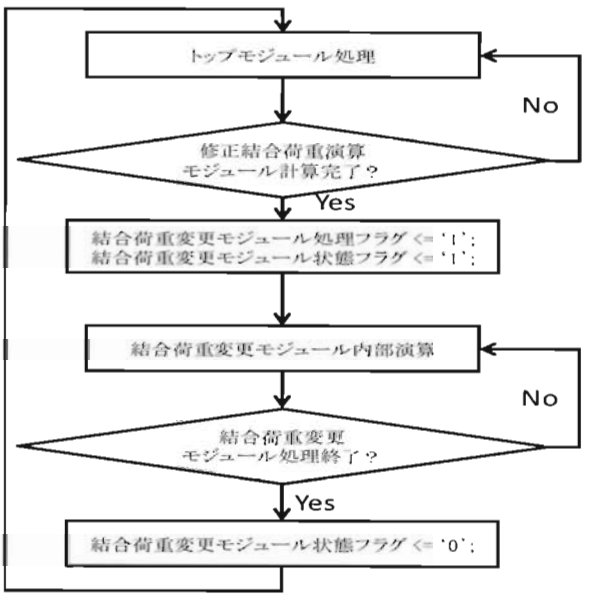

図10　結合荷重変更モジュールフローチャート

3.6　結合荷重修正量演算モジュール

結合荷重修正量演算モジュールは(5)式に従い結合荷重の修正量を計算する．図9に修正結合荷重演算モジュールのフローチャートを示す．トップモジュールで処理開始フラグ‘1’になったのち，活性化関数出力値$f(x)$，導関数出力値$f'(x)$，教師信号tを受け取り処理を開始する．

本モジュールでも各演算器は図8に従って動作を行い，各演算器が状態フラグ‘1’を返すことで結合荷重修正量演算モジュールの状態遷移を行う．

計算結果はトップモジュールへ出力され，結合荷重修正量演算モジュールの状態フラグ‘0’をトップモジュールへ返すことで処理を終了する．

3.7　結合荷重変更モジュール

このモジュールでは結合荷重を読みこみ，結合荷重修正量演算モジュールから出力された結合荷重修正量を加算し，新規結合荷重を作成する．図10に結合荷重変更モジュールのフローチャートを示す．処理開始フラグが‘1’になり結合荷重と結合荷重修正量の値を受け取ることで処理を開始し，同時に状態フラグを‘1’にする．計算された結合荷重をトップモジュールへ出力すると状態フラグが‘0’になりモジュールでの処理は終わる．その後最上位モジュールにおいて新規結合荷重をメモリへ書き込み，結合荷重の修正を終える．

4.　シミュレーションによる検証

4.1 設計環境

回路合成ツール，及び波形シミュレーションツールとターゲットデバイスとして選択したFPGA(Field Programable Gate Array)には以下のものを用いた．

回路合成ツール：Xilinx ISE11.4
波形シミュレーションツール：ISim11.4
FPGA：Spartan3 XC3S400-4

4.2 動作確認

動作確認を行うために，入力の1つに0-スタック故障が発生したと想定してシミュレーションを行った．その時の入力を(8)式，結合荷重を(9)式にそれぞれ示す．

$$\text{input} = \{0.100000, 0.900000, 0.100000, -0.100000, 0.000000\}, \quad (8)$$

$$\text{weight} = \{0.500000, 0.250000, -0.125000, -0.500000, 0.750000\}. \quad (9)$$

各演算モジュールにおいてシミュレーション波形を調べた．図11に全体のモジュール，図12に故障検知モジュール，図13に結合荷重変更モジュールのシミュレーション出力を示す．故障検知モジュールは故障発生信号を正しく出力し，結合荷重変更モジュールも160msごとに正しく結合荷重を変更していることを確認した．

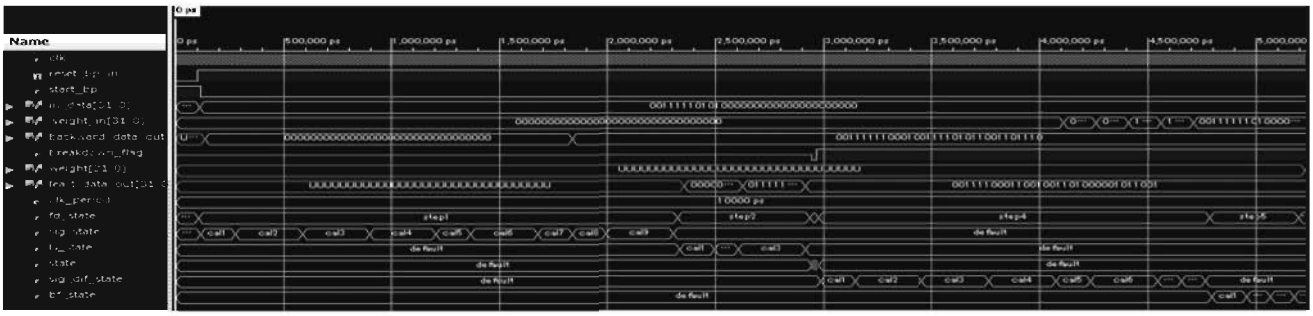

図 11 全体モジュールのシミュレーション波形

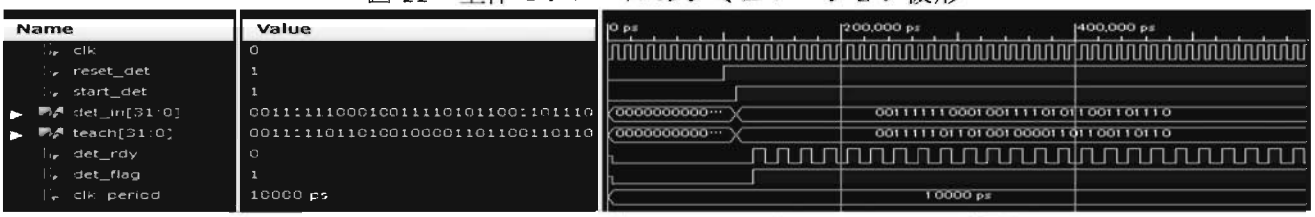

図 12 故障検知モジュールのシミュレーション波形

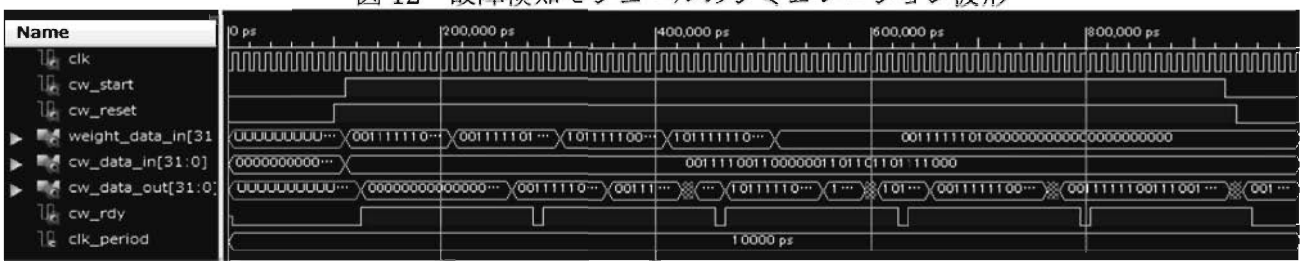

図 13 結合荷重変更モジュールのシミュレーション波形

5. おわりに

本研究では，PR 法を用いた故障補償機能を持ったニューラルネットワークを FPGA へ実装する前段階として，自律故障補償可能なハードウェアニューロンを設計した．

作成したプログラムは回路合成ツールを用いて論理合成を行い，生成された論理回路を用いてシミュレーションによる検証を行った．作成した回路はターゲットデバイスである FPGA 上のフリップフロップを 244%，4 入力組み合わせ回路を 168%使用することが分かった．またシミュレーションは事前の計算結果と一致し，正しく動作することを確認し，1 秒当たり約 625 万個の重み更新が可能である．

ターゲットとした FPGA へは，回路規模の関係から実装には至らなかった．全体の回路量の削減が今後の課題である．

参考文献

[1] Johannes Schemmel, Johannes Fieres and Karlheinz Meier, "Wafer-Scale In tegration of A nalog Neur al Networks," Neural Networ ks, 2008. IJCNN 2008. (IEEE W orld Congress on Computation al In telligence). IEEE International Joint Conference on,pp.:431 - 438, 2008.

[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による高速な故障補償," 電子情報通信学会技術研究報告. HIP, ヒューマン情報処理 Technical report of IEICE. HIP 98(130) pp.79-86, 1998.

[3] 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," 電気情報通信学会論文誌 C,J85-C(9),pp.861-864, 2002.

[4] C.Khunasraphan,K.Vanapipat and C.Lursins ap, "W eight Shifting Tecniques for Self-R ecovery Neur al Networks," IEEE TRANSACTIONS ON NEURAL NETWORAKS. VOL.5, NO.4,pp.651-658, 1994.

[5] 山森一人，草野真道,吉原郁夫, "自立故障補償可能なニューロンのバックワードパス部の設計と FPGA 上への実装," 宮崎大学工学部紀要 第 38 号,pp.349-354, 2009.