第3回バイオメトリクスと認識・認証シンポジウム
2013年11月26日～27日（日本科学未来館）

縞・斑点画像を生成／修復する反応拡散モデルのFPGA実装

An FPGA IMPLEMENTATION of a Reaction-diffusion Model Performing Stripe- and Spot-Image Generation and Restoration

石村憲意† 小室勝郎† Alexandre Schmid‡ 浅井哲也† 本村真人†
†北海道大学大学院情報科学研究科
‡スイス連邦工科大学ローザンヌ校 (EPFL)

Kazuyoshi ISHIMURA† Katsuro KOMURO† Alexandre SCHMID‡
Tetsuya ASAI† Masato MOTOMURA†
†Graduate School of Information Science and Technology, Hokkaido University
‡Swiss Federal Institute of Technology (EPFL)

アブストラクト

本研究は，指紋などの縞パターン画像の欠損や傷を，生物の形づくりの原理に基づいて修復するモデルのFPGA実装を目指したものである．従来の指紋認証は，指紋パターンの認証精度を上げる為に，ソフトウェアで高度な画像処理を施すが，傷や欠損のある指紋パターンに対する認識率の低下が問題となっている．その解決策の一つとして，生物の体表模様の生成や，傷が周りの模様に合わせて修復される事に着目した．このメカニズムを利用する事で，生物の自然治癒のような縞パターン画像の修復が可能であると考えられる．そこで，このような縞パターンを生成する反応拡散モデルを利用し，指紋パターンを補完するディジタルプロセッサの実装を行なった．

1 はじめに

指紋や掌紋，虹彩，静脈などの身体的特徴は千差万別である為に，個人を同定する有力な手がかりとなる．そこで，これらのパターンを認証に用いることにより，パスワード管理に煩わされる事無く，スマートフォンやATMを操作したり，あるいは入国審査における照合など，幅広い分野で利用されている [1].

それらのパターンの中でも指紋は，経年変化が少ないため，古くから認証に利用されている．指紋パターンの認証精度を上げる為，ハードウェア面では様々なセンサ方式が提案され，ソフトウェア面で高度な画像処理や認証が行なわれている [2].

しかし，指紋に傷や汚れなどがついている状態で認証する場合，高度な画像修復を行うために時間を要したり，低下する認識率が問題となっている．そこで我々は，解決策の一つとして，生物の体表模様 (ヒョウ柄やシマウマの縞模様) が生成される様子や，体表の傷が周りの模様に合わせて修復される事に着目した．このメカニズムを利用する事で，傷や欠損のある指紋パターンの自然な自己修復が可能であると考えられる．このような指紋，掌紋パターンを自己修復するディジタル反応拡散システムが提案されている [3]．本研究では画像の自己修復を行うプロセッサのFPGA実装が目的である．

本稿の構成は以下の通りである．まず２章において，本研究で利用する，縞模様を生成/修復するモデルの原理を示す．その後に、そのモデルを用いてどのように指紋が修復されるかをCシミュレーションにより示す．３章では，このモデルのFPGA実装の過程を一次元のアーキテクチャから説明し，二次元平面上で画像修復に必要な縞模様生成の様子を示す．最後にまとめと今後の予定について述べる．

2 縞・斑点模様生成／修復モデル

本研究では，指紋や掌紋画像の修復を行なうプロセッサを実現する為に，生体の模様生成や修復機能をモデル化した反応拡散方程式を用いる．この方程式は縞・斑点パターンを生成する上で最も重要な部分である．通常は，連続値を取り扱うため，偏微分方程式で表されるが [4]，より少ない計算資源 (ダイナミクスの単純化) で縞・斑点パターンを再現する試みが為されてきている [5], [6].

ここでは，その中でもハードウェア化が容易な，反応拡散 (Reaction-diffusion: RD) セルオートマトン (Cellular Automata: CA) モデル [7], [8] を用いた．実際に，このRDCAモデルを用いたアナログ集積回路実装が行なわれている．我々は，センサなどから入力された指紋画像に対して画像の修復を，高速でディジタル信号処理する為に，FPGA実装を目指す．

このRDCAモデルのダイナミクスを以下に示す．

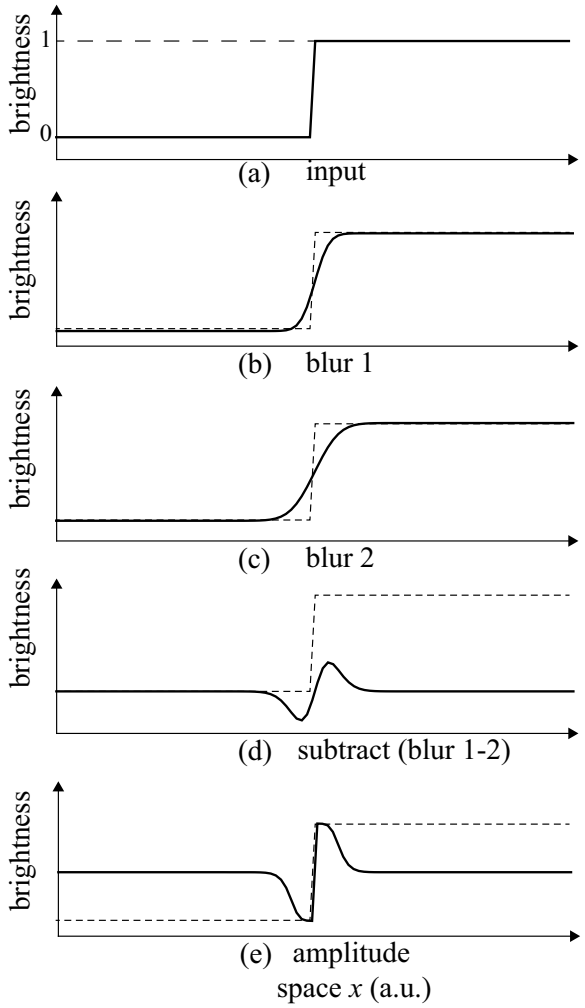

図 1: 一次元 RDCA モデルにおける波生成過程: (a) 初期入力波形 (ステップ関数), (b) Δt_0 拡散後の波形, (c) Δt_1 - Δt_0 拡散後の波形, (d) (c) から (b) を減算した波形, (e) 差分結果 (d) をシグモイド関数で増幅した波形.

$$
\begin{aligned}
&1(\text{Diffusion}) \\
&\partial u(\mathbf{r},t)/\partial t = D_u \nabla^2 u(\mathbf{r},t), \\
&\partial v(\mathbf{r},t)/\partial t = D_v \nabla^2 v(\mathbf{r},t), \\
&2(\text{Reaction}) \\
u(\mathbf{r},\delta t(n+1)) &= v(\mathbf{r},\delta t(n+1)) \\
&= f(u(\mathbf{r},\delta t\cdot n) - v(\mathbf{r},\delta t\cdot n) - c), \\
f(x) &= (1+\exp(-\beta \mathrm{x}))^{-1},
\end{aligned}
$$

n は時間のカウント，$\mathbf{r}$ は (x, y) 座標，c はシグモイド関数のオフセット値，β は関数の勾配の尺度を示している．そして，このモデルのダイナミクスを持つセルが格子状にネットワークを構成している場合を考える．各々のセルの状態は，隣接する四つのセル間で作用する重み付け加算と，シグモイド関数による増幅によって決定される．

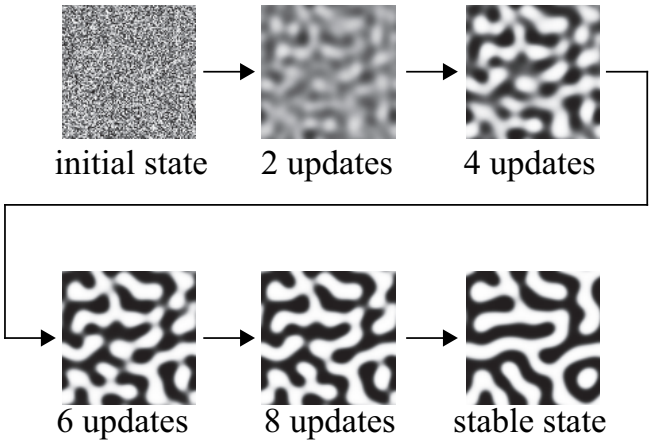

図 2: RDCA モデルを用いた，二次元平面上のランダムパターンから縞模様が生成される過程.

重み付け加算はそれぞれ拡散速度の異なるぼかし処理 (u と v) を表しており，それらは各セルにおいて畳み込まれる．セルの状態は，各点 (x, y) において，u，および v の二つの状態の差分を取って計算される．u と v による拡散方程式を，時間 δt で積分する．セルの次の状態は，$u-v$ をシグモイド関数で増幅した値によって決定される．

まず始めに，この一次元 RDCA モデル (各々のセルが，前後二つの近接セルと直列に繋がっている状態) における波の生成過程を図 1 に示す．図 1(a) はステップ関数を入力した初期状態を示している．図 1(b) はステップ関数が Δt_0 の時点で，パラメータ D_v で拡散した後の様子を示している．図 1(c) は Δt_1 - Δt_0 間で拡散させ，パラメータ D_u で入力されたステップ関数を拡散させたことを示している．図 1(d) は，図 1(b) と図 1(c) の差分であり，ここで初めて波が一つ生成された様子を示している．最後にこの差分がシグモイド関数により増幅された様子を図 1(e) に示す．ここで，図 1(e) に示される生成波形を，再びこのモデルに入力するプロセスを，入力信号の“更新”とする．一次元の波パターンは，この更新を繰り返すことによって形成される．

図 2 では，一次元 RDCA モデルから，実際の指紋画像修復に用いる二次元 RDCA モデル (格子状にセルが配置され，各々が四近傍セルと接続されている状態) に拡張し，縞パターンが形成される様子を示している．ランダムノイズパターンを RDCA モデルに入力し，出力されたパターンを，RDCA モデルの入力として繰り返し更新することで，安定した縞パターンが形成されることを C シミュレーション上で確認できた．

そこで，実際に指紋画像が入力された場合のパターン形成の過程を図 3 に示す．時間の経過とともに，指紋画像に付いている傷やノイズが，周囲の構造 (指紋の縞模様

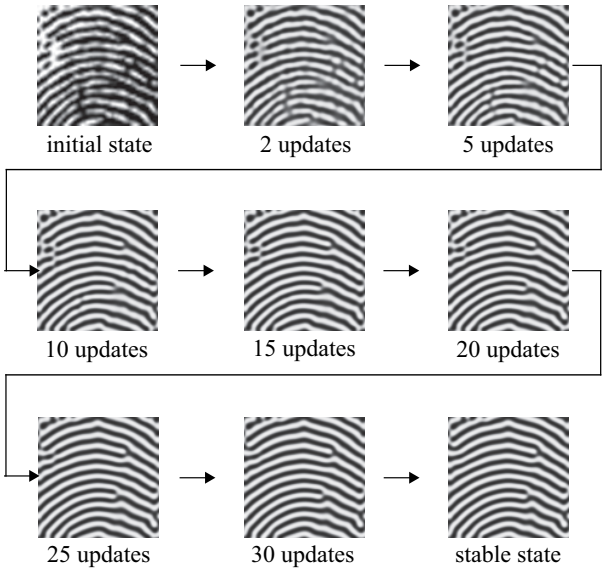

図 3: 傷やノイズが付着した指紋画像が RDCA モデルの縞模様生成によって自己修復される過程.

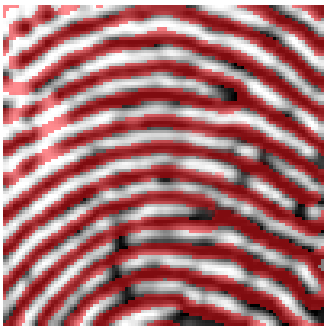

図 4: 傷やノイズが付着した指紋画像と、修復されて安定状態になった指紋パターンを重ね合わせた図.

の間隔) に従いながら滑らかに修復されて行く様子が確認出来る．パターンが安定するまでに更新が約 45 回行なわれた．このようにパターンが安定する為に要する更新回数は，縞模様の線の幅 (空間周波数) にのみ依存し，画像サイズには依存しない．

図 4 に，初期状態として入力された指紋画像に，修復されて安定状態になった指紋パターンを重ねた様子を示す．この図から，元の指紋画像の構造を崩す事無く，途切れた箇所や，指紋の縞模様をまたいだノイズを修復している事がわかる．

3 RDCA モデルの FPGA 実装

3.1 一次元 RD プロセッサのアーキテクチャ

前述のシミュレーションにより，RDCA モデルを指紋画像修復に利用可能であることを確認できた．本章では，

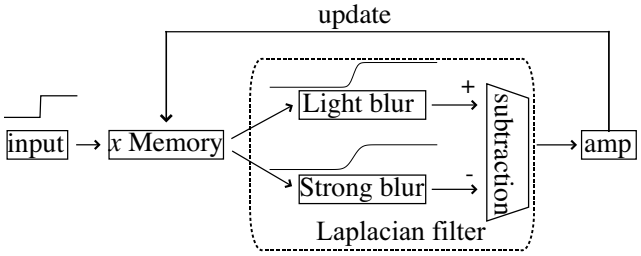

図 5: 一次元 RD プロセッサのアーキテクチャ

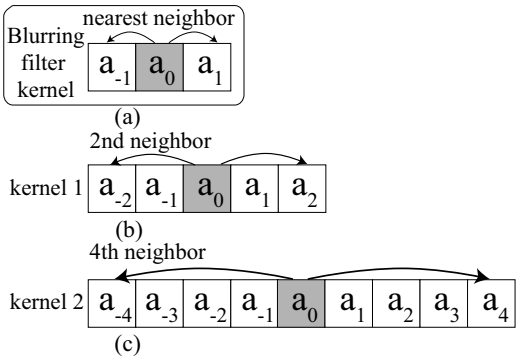

図 6: 一次元ぼかしフィルタのカーネル．(a) 二近傍ぼかしフィルタのカーネル．(b) 弱いぼかしフィルタとして利用するサイズ 5 のカーネル (kernel 1)．(c) 強いぼかしフィルタとして利用するサイズ 9 のカーネル (kernel 2).

このモデルを FPGA 実装してディジタル信号処理を行なうためのアーキテクチャについて説明し，実装するまでの過程を一次元モデルから示す．

図 5 に一次元反応拡散 (RD) プロセッサのアーキテクチャを示す．ここでは，一次元に配列した RDCA モデルのセルに対応させる為に，8bit の値を持ち，120 個のアドレスを持つメモリを用意した．前後二つのアドレスとは繋がっているが，両端同士は繋がっていない．図 1 で示したように，波を生成する為には，入力波形を二つの異なる拡散強度でぼかす必要がある．そこで，ぼかしの強度が異なる二つのフィルタで構成されるフィルタ部を用意した．各フィルタが二つの異なる波形を出力した後，差分計算をする．これらのぼかし処理と差分計算はひとまとめにして，一般的なディジタル・ラプラシアンフィルタで実装可能である．最後に，差分結果を増幅し，メモリに格納する．このメモリから再び読み出してフィルタに入力する事で更新を繰り返し，波や縞模様生成を実現する．

ここで，ぼかしフィルタを構築するためのカーネルについて説明する．図 6(a) は二近傍のぼかしフィルタのカーネルである．中央部の a_0 の次の状態 (出力) は，両端のセルが持つ値の影響を受ける (高い値から低い値へ移り、ぼ

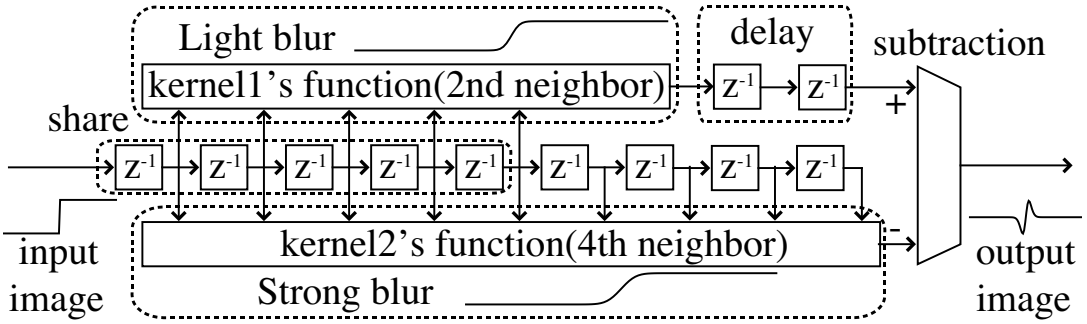

図 7: 一次元 RD プロセッサのフィルタ部におけるデータストリームアーキテクチャ

やけていく) ため,

$$\mathrm{out}_0 = (a_{-1} + 2a_0 + a_1)/4$$

と表される. 図 6(a) は最近接セルだけであったが, 図 6(b), (c) のように, ぼかし計算を行なうカーネルの範囲を広げる事で, ぼかし強度が強くなる. カーネルサイズが 5 (kernel 1), 9 (kernel 2) の場合における, a_0 の出力は順に

$$\text{kernel1 } \mathrm{out}_0 = (a_{-2} + 4a_{-1} + 6a_0 + 4a_1 + a_2)/16$$

$$\text{kernel2 } \mathrm{out}_0 = (a_{-4} + 5a_{-3} + 11a_{-2} + 15a_{-1} + 16a_0 + 15a_1 + 11a_2 + 5a_3 + a_4)/80$$

と表される.

図 7 はフィルタ部の構成を示している. メモリから読み込まれたセル, あるいはピクセルの値が一つずつ, クロック同期してレジスタに格納されながら移動する. 入力波形は, 図 6(b), (c) で示した, ぼかし強度の異なるカーネル 1 とカーネル 2 のフィルタに通される. その際に, それらのカーネルで使用するレジスタの一部を共有することで省面積化を実現している. また, ぼかし強度が異なる二つのフィルタを通す際, 処理に必要となるセル (ピクセル) 数が異なることから, 出力の遅延が発生する. その為, カーネル 1 のフィルタの出力部と差分器の間に遅延レジスタを複数個挿入することにより, 差分のタイミングを調整する. 差分後に増幅された波形をメモリに格納する. そして次の入力波形として読み出されて更新される.

ぼかしフィルタの式と図 7 からわかるように, セルの値が読み込まれてからアンプ部を通るまでに 5 クロック要する. その為, 読み込むメモリアドレスを指定するカウンタと, それから 5 クロック遅れて, 書き込むメモリアドレスを指定するカウンタを用意した.

3.2 一次元 RD プロセッサの実装結果

前章で述べたアーキテクチャを基に一次元 RD プロセッサを FPGA 上に実装し, その基本的な動作確認を行った.

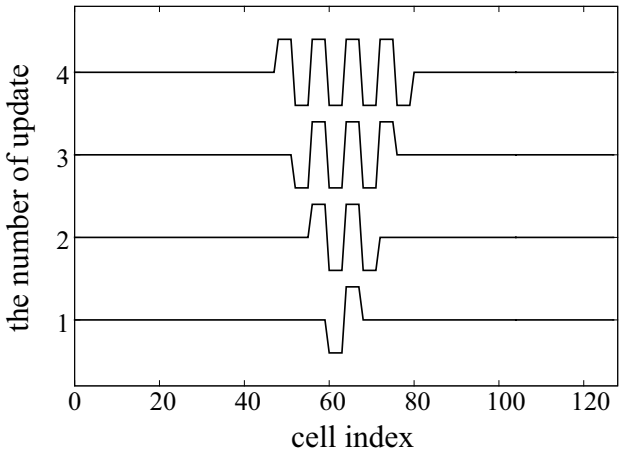

図 8: 一次元 RD プロセッサの FPGA 実装結果

図 8 は, FPGA 上の一次元 RD プロセッサに対し、ステップ関数を入力した時の出力結果を示している. 横軸はセルのアドレスを表し, 縦軸は更新の回数を表している. 更新プロセスを繰り返していくことで, 始めはステップ関数だった波形から, 徐々に波が生成されていく様子が観測できる.

3.3 二次元 RD プロセッサのアーキテクチャ

ここまでで, 一次元 RD プロセッサを FPGA 実装し, 波が広がっていく様子を確認できた. そこで, 一次元 RD プロセッサを基に, 二次元 RD プロセッサへと拡張していく過程を説明する.

まず二次元平面上のぼかし処理の方法について説明する. 格子状に RDCA モデルのセルを配置した場合における, 四近傍のぼかしフィルタカーネルを図 9(a) に示す. ある一つのセル ($\mathrm{a_C}$) の状態に着目すると, その上下左右に隣接するセル ($\mathrm{a_N}, \mathrm{a_E}, \mathrm{a_W}, \mathrm{a_S}$) の状態と影響を及ぼし合う. この二次元ぼかしフィルタをそのまま実装するには, フィルタの前に, カーネルサイズに応じた規模のラインバッファを設ける必要があり, 回路規模が大きくなりやすい.

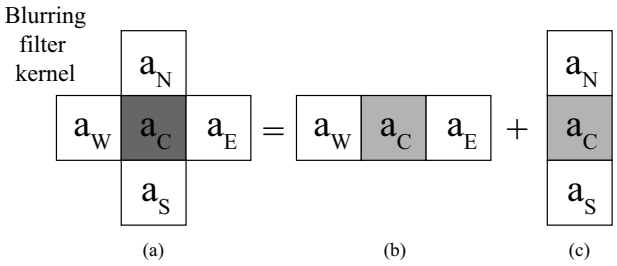

図 9: 二次元ぼかしフィルタのカーネルとその x 軸方向および y 軸方向への分割．(a) 四近傍ぼかしフィルタのカーネル．(b) x 軸方向のぼかし処理を行う一次元ぼかしフィルタのカーネル．(c) y 軸方向にぼかし処理を行う一次元ぼかしフィルタのカーネル．

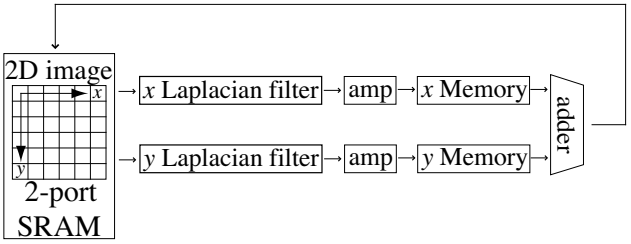

図 10: 二次元 RD プロセッサのアーキテクチャ

そこで，この二次元ぼかしフィルタカーネル (図 9(a)) を，二つの一次元フィルタカーネルに分解する事を考える (図 9(b)，(c))．このようにすることで，x 軸，y 軸に分けてそれぞれの方向にぼかし処理を施すことができ，さらに一次元 RD プロセッサで利用したラプラシアンフィルタがそのまま適用出来る．その後に足し合わせれば，二次元のぼかし処理をそのまま行なうのと同じ結果が得られる．

二次元 RD プロセッサのアーキテクチャを図 10 に示す．二次元画像における縞模様生成には，x 軸，y 軸の二方向のぼかし処理が必要であった．ここで，画像データを保持するメインメモリとしてトゥルー・デュアルポート SRAM を使用することにより，メモリアドレスを独立に二つ指定して読み出し，並列に出力することができる．その為，x 軸，y 軸方向のぼかし処理を並列化することができる．そこで，一次元 RD プロセッサで用いたラプラシアンフィルタを二つ用意した．これらのフィルタの直後に x メモリ，y メモリ (一次元 RD でも使用したシングルポート SRAM) を用意し，一時保存する．これらの一時メモリは、x，y 軸方向のぼかし処理の結果を足し合わせて，二次元のぼかし画像生成のタイミングを調整する為に用意されている，

次に加算結果をメインメモリ (図 10 左側) に格納する．これをさらにぼかしフィルタへ入力し，更新を繰り返す

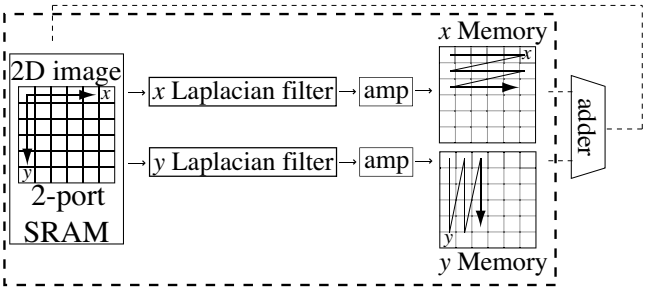

図 11: メインメモリからデータが読み出され，フィルタとアンプを通って x，y メモリに書き込まれる時の，各メモリの読み込みと書き込みの方向を示した図．

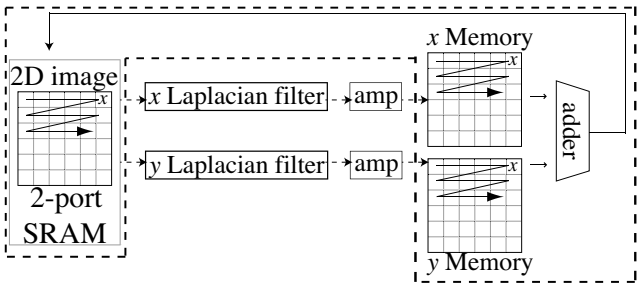

図 12: x，y 一時メモリからデータが読み出され，加算器を通って，メインメモリに書き込まれる時の，各メモリの読み込みと書き込みの方向を示した図．

事で，二次元の縞模様が生成されていく．

二次元 RD プロセッサのアンプ部は，本来ならば加算器の後に置くことで一つに出来る．しかしそこにアンプを設置すると，ぼかしフィルタの出力で負の値を持つ部分がある為に，符号ビットも合わせた 9bit の値の保持が必要となってしまい，一時保存用の x，y メモリの規模が大きくなってしまう．その為に，それぞれのぼかしフィルタの直後にアンプ部を置き，ぼかしフィルタの出力結果を符号無しの 8bit に増幅している．

このアーキテクチャの動作は、大きく次の二つに分けられる．まず一段階目は，メインメモリから二つの値を読み出していき，それぞれぼかしフィルタとアンプを通して一次メモリに保存する (図 11)．その際に、x 軸方向にメモリを読み込む為のアドレスカウンタは，メモリアドレスの先頭から順に 1 ずつカウントアップする．その為，一時メモリ x には行方向に値が格納されて行く．一方，y 軸方向に読み込む為のアドレスカウンタは，$N \times N$ ピクセルの正方形画像を扱う為，N ずつカウントアップしていく．そして読み出している画像列の最後のピクセルの後，次列の最上部のメモリアドレスを指定する．これらの読み出しを画像の最後のピクセルまで繰り返す．

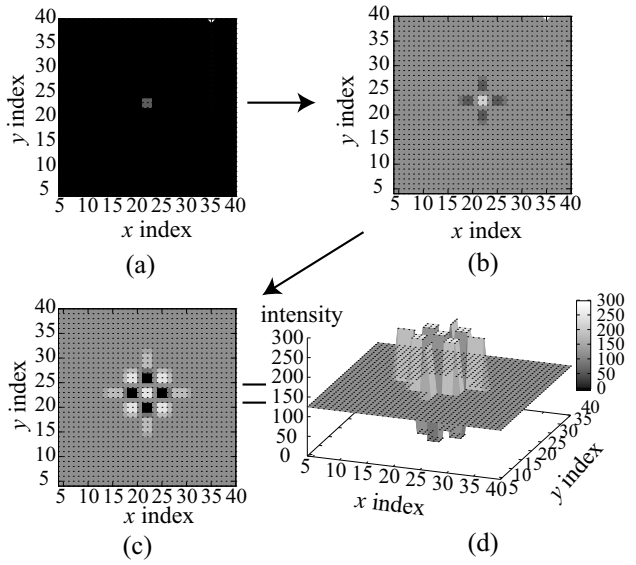

図 13: 二次元 RD プロセッサの FPGA 実装結果．(a) 初期状態 (インパルス入力)．(b) 更新一回目の縞生成の様子．(c) 更新二回目の縞生成の様子．(d) 更新二回目で生成された縞模様の三次元描画．

二段階目は，二つの一時メモリから値を読み込み，それらを加算してメインメモリに書き込む (図 12)．その際に x，y 両方の一時メモリから x 軸方向に読み出す為，共通のアドレスカウンタを用意し、順に 1 ずつカウントアップする．このようにして同じアドレスの値同士を足し合わせて、メインメモリに x 軸方向に順に加算結果を書き込む．各メモリの読み込み，書き込み用アドレスカウンタ，メモリのリード，ライトイネーブル信号は、全て FSM で制御している．

3.4 二次元 RD プロセッサの実装結果

二次元 RD プロセッサの実装結果を図 13 に示す．画像サイズは，45×45 ピクセルであり，境界は固定端である．z 軸方向に、各ピクセルの状態値を濃淡で表す．まず中心部にのみ初期値としてインパルスを与える．ここから入力画像の更新を繰り返すと，波が平面に広がり，縞模様が生成される様子が確認出来る．この結果から，より大きな画像サイズを取り扱えるようにし，指紋画像を入力する事で，指紋画像に付いた傷や欠損を，自然な形で自己修復する二次元 RD プロセッサを実装できると考えられる．

4 まとめ

従来の指紋認証では，傷や欠損のある指紋パターンに対する，高度な画像処理に要する時間や認識率の低さが問題であった．本研究では，この欠損や傷を，生物のように，周りのパターンに合わせて自然な形に自己修復するディジタルプロセッサの実現を目指している．そのために，ハードウェア化が容易な RDCA モデルを選択した．このモデルを利用して指紋パターンの自己修復が出来る事を C シミュレーションにより示した．

次に，FPGA 実装の為に，RDCA モデルによる RD プロセッサのアーキテクチャを提案した．その際に，ラプラシアンフィルタを構成する二つのぼかしフィルタのレジスタ部を共有することにより，省面積化を図った．さらに，二次元 RD プロセッサのアーキテクチャでは，トゥルー・デュアルポート SRAM を採用することにより，x，y 軸方向のぼかし処理を，並列に行なえるようにした．提案したアーキテクチャを FPGA 上に実装し，画像修復の基本となる，縞模様の生成を確認した．今後は，指紋画像の修復を FPGA 上で実際に行なっていく予定である．

参考文献

[1] 児玉充晴, 梅崎太造, 佐藤幸男, “情報漏えい対策システムへの指紋認証の適用とその 発展形態の提案,” 信学論 (D-I) , vol. J87-D-I, no.2, pp. 278-286, 2004.

[2] 若原徹, 木村善政, 鈴木章, 塩昭夫, 佐野睦夫 “指紋隆線方向分布とマニューシャ対応付けを用いた指紋照合,” 信学論 (D-II), vol. J86-D-II, no.1, pp. 63–71, 2003.

[3] 藤田渉, 青木孝文, 樋口龍雄 “指紋画像処理のためのディジタル反応拡散システムの設計,” 信学技報, NLP98-79, pp. 9–16, 1998.

[4] A. M. Turing “The chemical basis of morphogenesis,” *Phil. Trans. R. Soc. Lond B.*, vol. 237, pp.37–72, 1952.

[5] D. A. Young “A local activator-inhibitor model of vertebrate skin patterns,” *Math. Biosci.*, vol. 72, pp.51–58, 1984.

[6] M. Markus and B. Hess “Isotropic cellular automaton for modeling excitable media,” *Nature.*, vol. 347, no. 6288, pp.56–58, 1990.

[7] Y. Suzuki, T. Takayama, I. Motoike, and T. Asai “Striped and spotted pattern generation on reaction-diffusion cellular automata: Theory and lsi implementation,” *Int. J. Unconv. Comput.*, vol. 3, pp.1–13, 2007.

[8] T. Asai and I.N. Motoike, “Self-organizing striped and spotted patterns on a discrete reaction-diffusion model,” *NOLTA,* vol. 2, no. 3 pp. 363–371, 2011.