卒業論文

画像処理アルゴリズムの

ハード/ソフト最適分割の検討

氏名　　：乾　圭佑
学籍番号：2260050091-4
指導教員：山崎　勝弘　教授
提出日　：2009 年 2 月 19 日

立命館大学　理工学部　電子情報デザイン学科

内容梗概

本論文では、Verilog-HDLを用いたハードウェア設計とハード/ソフト協調設計におけるハード/ソフト分割の検討について述べる。対象アプリケーションとして、ラプラシアンフィルタによるエッジ検出と、平滑化フィルタによるノイズ除去をとりあげた。ラプラシアンフィルタによるエッジ検出と、平滑化フィルタによるノイズ除去をそれぞれソフトウェアおよびハードウェアモジュールとして設計をした。 ハード/ソフト最適分割の検討では、ソフトウェアでの処理の負荷をモジュール毎に計測し、その結果からハード/ソフトの分割パターンを決定し、各パターンにおける検討と考察を行った。

目次

図目次

表目次

1. はじめに

近年の半導体集積技術の向上は、LSI の小型化、高速化、省電力化などを可能にしている。特に、複数の機能を 1 つのチップ上に集積したシステム LSI は、携帯電話や車載機器、デジタルカメラのような、我々が普段手にする様々な電子機器に広く用いられ、高い付加価値を与えている。 一方で、これらの対象アプリケーションやアーキテクチャの多様化、大規模化は、半導体設計の複雑化という問題をもたらしている。また、信号処理技術の発展のような、対象アプリケーションにおける技術の向上は、製品における開発期間の短縮を促進しており、設計量の増大と同時に、仕様変更に耐え得る柔軟な設計が求められている。現状では、LSI の複雑度が 50%上昇する間に、設計の生産性は 20%しか向上しないという統計があり、設計生産性、設計品質の危機として問題となっている。

この問題に対処するため、ハード／ソフト協調設計という手法が注目されている。これは、ハードウェアとソフトウェアの協調によって様々な機能を実現しているシステム LSI において、ハード／ソフトを別々に設計、検証するのではなく、双方の設計者が協調して設計、検証を進めていくことで、システム全体の最適化と、設計の効率化を図る手法である。 しかし、ハード／ソフト協調設計は、その手法がまだ確立されておらず、特にシステム全体の最適化において重要となるハード／ソフトの分割は、多くの部分が人手に委ねられている。設計生産性の危機を解決するための手段として、これらの設計手法の確立が求められている。同時に、設計者に対しては、ハードウェア、ソフトウェア、そしてシステム全体の設計に関する体系的な知識が求められている[2][3]。

本研究室では、ハードウェアとソフトウェアの最適な分割するために、対象となる問題のハードウェアのリファレンスにもなるC言語のプロトタイプが完成した段階でソースコードを解析して、設計の早期段階から、ハードウェアとソフトウェアの最適分割を見つけ出すことを目的に取り組んでいます。そこで私はシステムの検証用として、アプリケーションを作成した[2]。

本研究では画像のノイズ除去とエッジ検出をとりあげる。これらの画像処理は、携帯電話やデジタルカメラのような電子機器で広く用いられている。しかし、多量のデータに対して計算を繰り返し行うため、演算量が大きく、組み込み用途の低速なCPUだけで必要となる性能を満たすことは難しい。そのため、一部もしくは全ての処理をハードウェアで実現することで性能の向上が図られることが多く、ハード／ソフト協

調設計を適用する対象として適していると考えられる。

本研究における協調設計の流れは、まず画像処理全体をソフトウェアによって実現する。さらに、システム上のCPUにおける処理の負荷を、分割した機能ブロック毎に計測する。次に、各機能ブロックをハードウェアモジュールとして設計した後、ハードとソフトの複数の分割パターンを検討する。最後に、性能や規模などから各システムを比較評価し、画像処理における最適な分割パターンを決定する。

2. ラプラシアンフィルタによるエッジ検出のハードウェア設計

2.1 ラプラシアンフィルタのアルゴリズム

ラプラシアンフィルタはエッジの方向によらず，エッジの強度のみに敏感に反応するフィルタである。ラプラシアン∇^2は空間二次微分を行うオペレータであり，次のような演算を行う。

$$\nabla^2 f(x,y) = f_{xx}(x,y) + f_{yy}(x,y) \tag{2.1}$$

これを差分形式で表すと，

$$f_{xx}(x,y) = f(x-1,y) - 2f(x,y) + f(x+1,y) \tag{2.2}$$
$$f_{xx}(x,y) = f(x,y-1) - 2f(x,y) + f(x,y+1) \tag{2.3}$$

となることから式(3.1）は次のようになる。

$$\begin{aligned}\nabla^2 f(x,y) &= f_{xx}(x,y) + f_{yy}(x,y) \\ &= f(x-1,y) + 2f\,(x,y) + f(x+1,y) + f(x,y-1) - 2f(x,y) + f(x,y+1) \\ &= f(x-1,y) + f(x+1,y) + f(x,y-1) + f(x,y+1) - 4f(x,y)\end{aligned} \tag{2.4}$$

これを係数で表現すると，

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \tag{2.5}$$

となる。これをラプラシアンフィルタという。また，４５°方向も含めると，

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{2.6}$$

という形式になり，この形もよく使われる。

ラプラシアンフィルタは二次微分を使っているため，画像の明るさの変化に強く反応する。ただし，雑音まで強調してしまうという欠点もある。そこで画像の平滑化を行ったあとでラプラシアンを使うといった工夫が行われている［4］［5］。図１に、画像とフィルタの畳み込み積分を示す。図１の●を中心に、フィルタと画像の対応する各値を掛け合わせ、それらすべてを足し合わせた値を出力とする。このような操作をすべての画素について施す処理を、空間フィルタリング処理という[4][5]。本研究ではフィルタとしてラプラシアンフィルタと、平滑化フィルタを設計した。

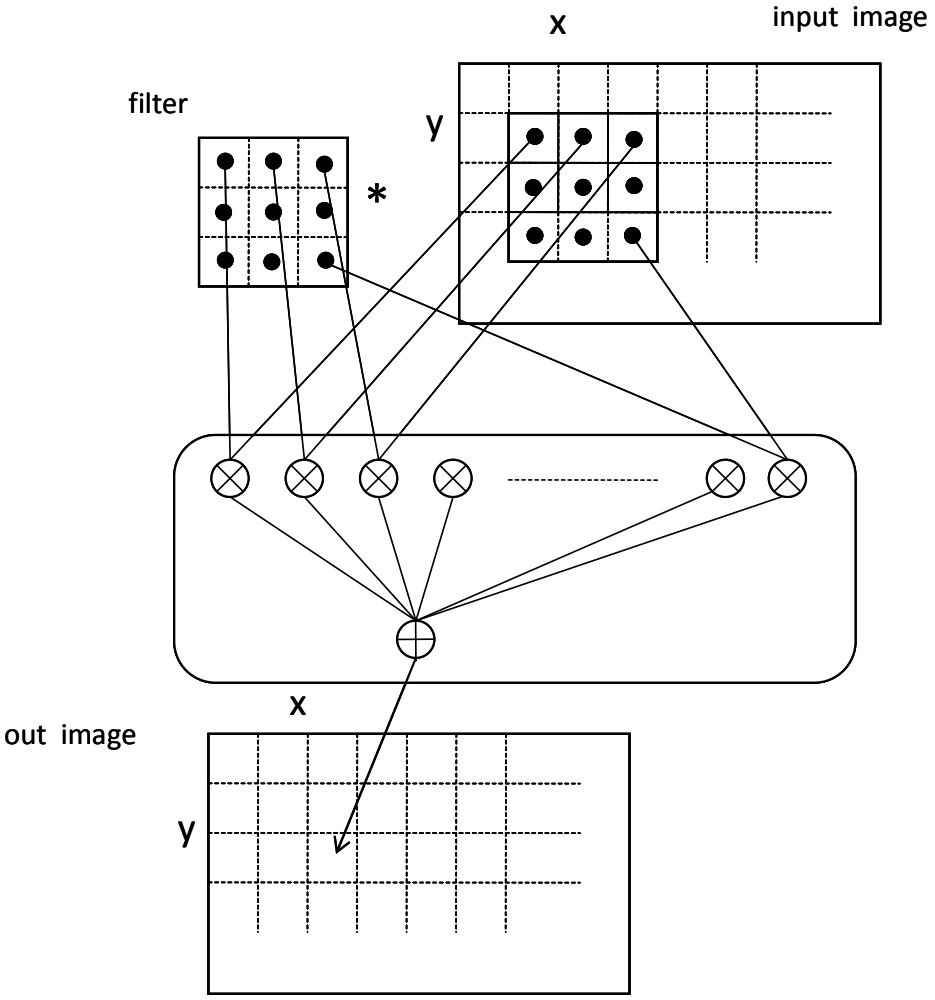

図１：画像とフィルタの畳み込み積分

2.2 ラプラシアンフィルタによるエッジ検出回路の設計

ラプラシアンフィルタによるエッジ検出アルゴリズムをハードウェアモジュールと

して実現した。ラプラシアンフィルタによるエッジ検出回路の全体構成を図２に示す。フィルタ演算できるように画像データをメモリに格納し、格納したデータを水平・垂直同期をとりアドレス指定し、３行×３列の画像を抽出するモジュールと、ラプラシアンフィルタ演算をするモジュールと、演算処理したデータをメモリに格納しディスプレイに出力するモジュールの３つのモジュールで構成している。

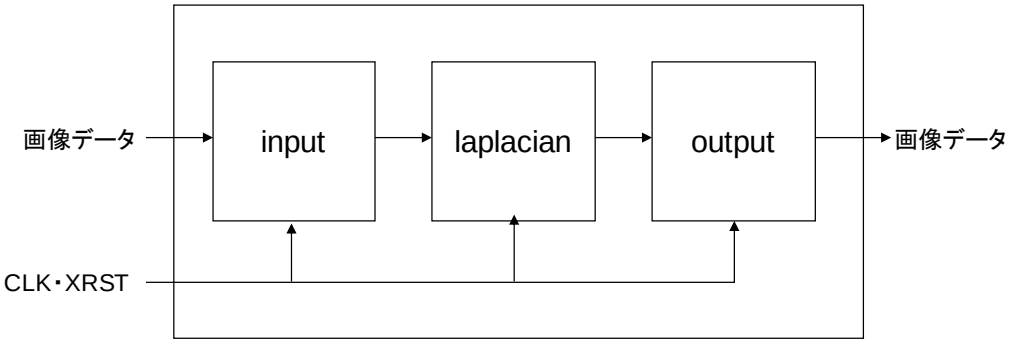

図２：ラプラシアンフィルタによるエッジ検出回路

2.3 各モジュールの説明

2.3.1 input モジュール

input モジュールの入出力インターフェースを図３に、信号線の説明を表 1 に示す。このモジュールはまず、画像データを読み込みＦＰＧＡ内部にあるブロックＲＡＭを構築し、アドレスとデータを格納する。アドレスはアップカウンタを使用し構成する。格納が終わるとイネーブル信号が入力されメモリの構築が終える。次にメモリのアドレスを指定し、水平・垂直制御をとり、３行×３列の９画像を順番に抽出するモジュールである。

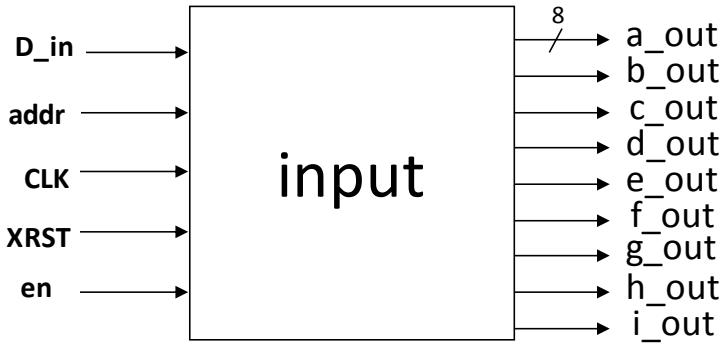

図３：input モジュールインターフェース

表１：input モジュールの入出力信号

信号名	方向	幅(bit)	詳細
d_in	入力	8bit	元画像のデータ入力
addr	入力	17bit	アドレス
CLK	入力	1bit	クロック信号
XRST	入力	1bit	リセット信号
en	入力	1bit	イネーブル信号
a～i_out	出力	各8bit	並び替えた画像データ出力

次に水平・垂直同期のメカニズムを図４に示す。通常画像データを読み込むとき１行ずつ左から順番に読み込んでいく。しかし、フィルタ処理する時は３行３列の画素を同時に読み込んでいかなくてはならないので水平垂直同期が必要になってくる。そこでいったん画像データをメモリ読み込んで、アドレス指定して読み込んでいく構成にした。本研究では画像のサイズは６４×６４画素で PGM 形式の画像を使用した。入力データを８bit、出力データ８bit で９画素分を同時に出力するように構成した。

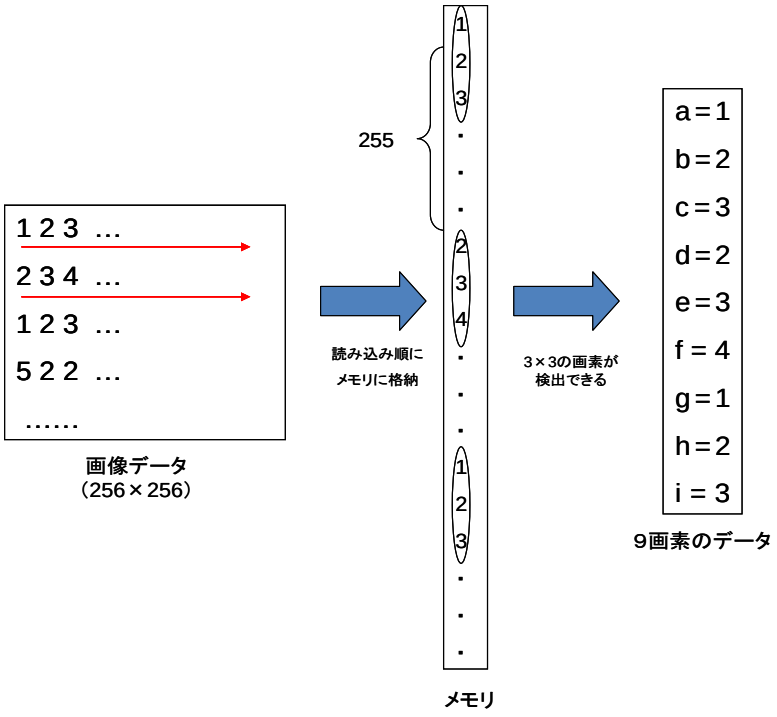

図４：水平垂直同期メカニズム

2.3.2 laplacian　モジュール

laplacian モジュールの入出力インターフェースを図５に、信号線の説明を表２に示す。また図６に laplacian モジュールの内部処理を示す。laplacian モジュールは input モジュールで水平・垂直同期を取った画像データを３行×３列の大きさのラプラシアンフィルタに通して二次微分を行い、エッジ検出処理をして画像データを出力する。

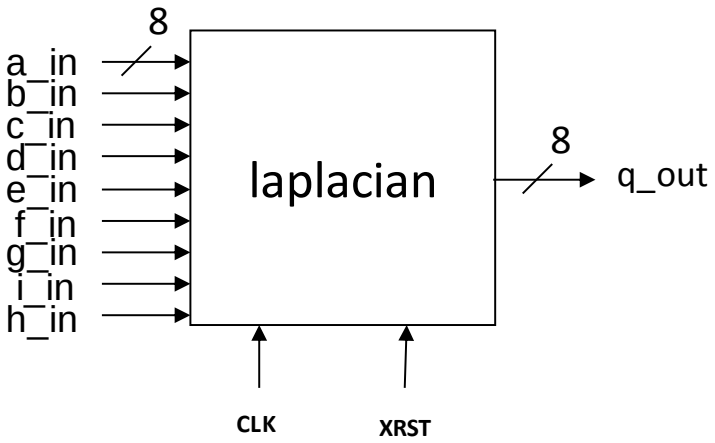

図５：laplacian モジュールインターフェース

表２：laplacian モジュールの入出力信号

信号名	方向	幅(bit)	詳細
a_in～i_in	入力	各8bit	元画像入力
CLK	入力	1bit	クロック
XRST	入力	1bit	リセット信号
q_out	出力	8bit	エッジ検出した画像出力

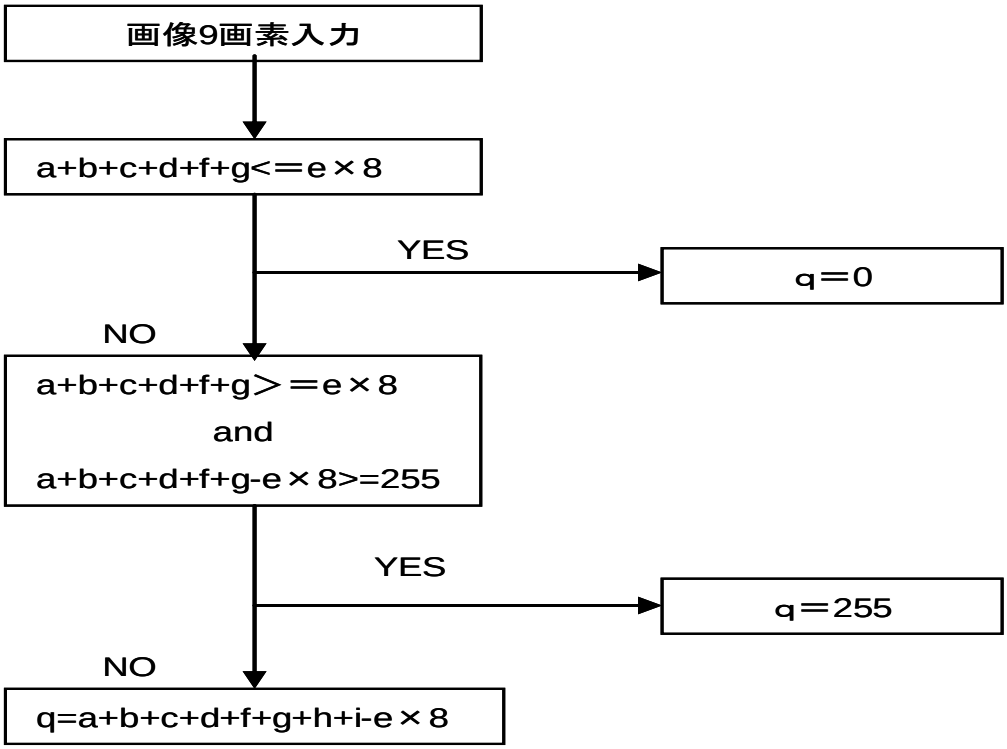

図６：ラプラシアンの内部処理

2.3.3　output モジュール

output モジュールの入出力インターフェースを図７に､信号線の説明を表３に示す。output モジュールはＦＰＧＡ内部にあるブロックＲＡＭを構築し、アドレスとデータを格納する。アドレスはアップカウンタを使用し構成する。格納が終わるとイネーブル信号が入力されメモリの構築が終える。laplacian モジュールでエッジ検出した画像をメモリに格納し、コンピュータ画面上に出力するモジュールである。

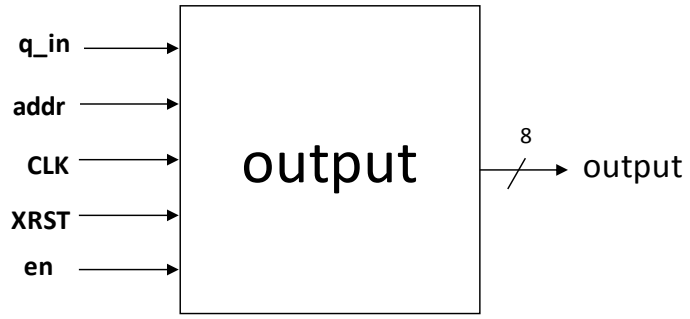

図７：output モジュールインターフェース

表 3：output モジュールの入出力信号

信号名	方向	幅(bit)	詳細
q_in	入力	8bit	画像処理された画像データ入力
CLK	入力	1bit	クロック
XRST	入力	1bit	リセット信号
Output	出力	8bit	画面へ画像出力
addr	入力	17bit	アドレス
en	入力	1bit	イネーブル信号

2.4 ハードウェアの検証と評価

ラプラシアンフィルタ回路を Verilog-HDL で記述したものを、各モジュールごとにＩＳＥで論理合成を行い、回路規模と遅延時間を測定した。また Modelsim でシミュレーションを行い、ハードウェアクロック数を計った。計測に用いたデータは，PGM 形式の 64×64 画素の画像を用いた。計測結果を表 4 に示す。

表 4：ラプラシアンフィルタによるエッジ検出の回路規模と遅延

	画像ファイル読み込み	ラプラシアンフィルタ演算	画像処理結果出力
回路規模(slice)	31685	248	31685
遅延(ns)	3.862	18.524	3.832
HW クロック数	14400	14400	14400

HW クロック数については逐次的に処理されるため、各モジュールのクロック数は同じになった。次に図 8 にエッジ検出の結果を示す。

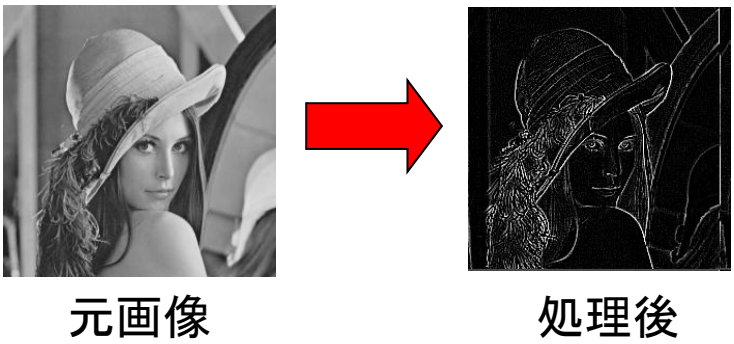

図８：ラプラシアンフィルタによるエッジ検出結果

3. 平滑化フィルタによるノイズ除去のハードウェア設計

3.1 平滑化フィルタのアルゴリズム

画像の濃淡値の変化を滑らかにする処理を平滑化と呼ぶ。処理により画像はぼけたような印象になる。ノイズの除去にも用いられる。
フィルタ係数は

$$\begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix} \tag{3.1}$$

となり、フィルタ係数の中央の値が重なる入力画像の画素の位置が、出力画像の画素の位置に対応する。出力画素の濃淡値は入力画素の濃淡値にフィルタ係数との積和を取ることにより得られる。 この処理では、入力画像の各画素の値は係数の総数である９により除算され、最後にそれらの値の和をとることから、画素の濃淡値の値を平均化していることが分かる。 この例では3×3の大きさの配列を用いているが、より大きな5×5や7×7などのフィルタ係数を用いることも出来る。

3.2 平滑化フィルタによるノイズ除去回路の設計

平滑化フィルタによるノイズ除去アルゴリズムをハードウェアモジュールとして実現した。平滑化フィルタによるノイズ除去回路の全体構成を図９に示す。フィルタ演算できるように画像データをメモリに格納し、格納したデータを水平・垂直同期をとりアドレス指定し、３行×３列の画像を抽出するモジュールと、平滑化フィルタ演算のモジュールと、演算処理したデータをメモリに格納しディスプレイに出力するモジュールの３つのモジュールで構成している。

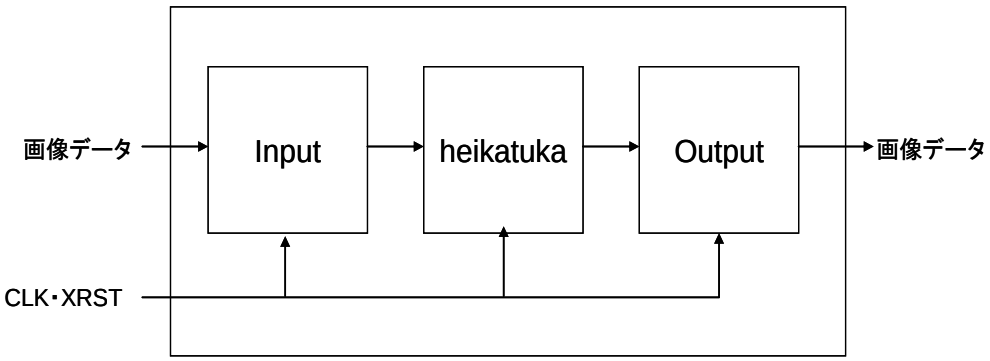

図９：平滑化フィルタによるノイズ除去回路

3.3 各モジュールの説明

3.3.1 input output モジュール

ラプラシアンフィルタで使ったInput, Outputモジュールを使用した。

3.3.2 Heikatuka モジュール

Heikatuka モジュールはの入出力インターフェースを図 10 に、信号線の説明を表５に示す。また図 11 に Heikatuka モジュールの内部構造を示す。
Heikatukaモジュールは３行×３列の元画像を切り出した部分に３行×３列のフィルタに通して、平均化するというモジュールである。

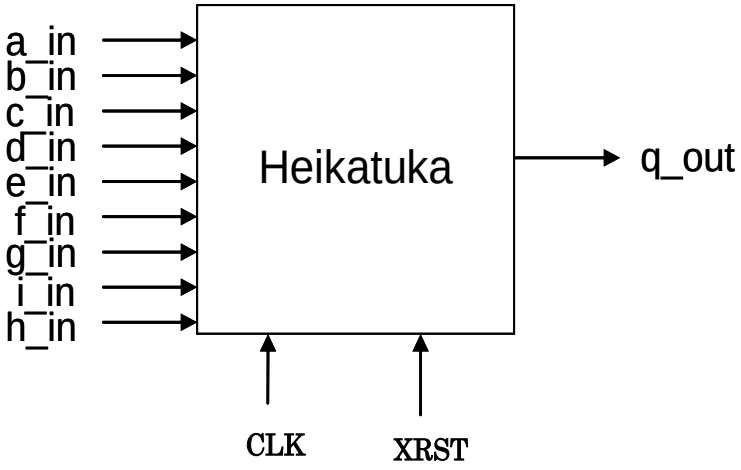

図１０：Heikatuka モジュールのインターフェース

表 5：Heikatuka モジュールの入出力信号

信号名	方向	幅(bit)	詳細
a_in～i_in	入力	各8bit	元画像データ入力
CLK	入力	1bit	クロック
XRST	入力	1bit	リセット信号
q_out	出力	8bit	平滑化処理結果出力

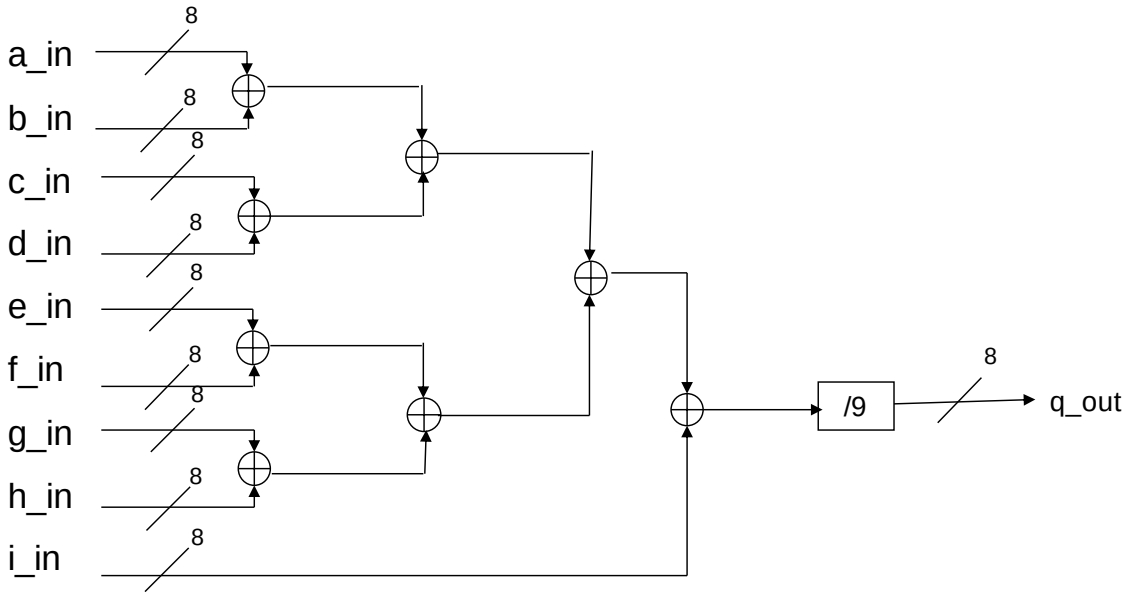

図１１：Heikatuka モジュールの内部構造

3.4 ハードウェアの検証と評価

平滑化フィルタによるノイズ除去のハードウェアをＩＳＥで論理合成し、回路規模と遅延時間を測定した。また Modelsim で動作を確認しハードウェアクロック数を測定した。結果を表６に示す。

表 6：平滑化フィルタによるノイズ除去の回路規模と遅延

	画像ファイル読み込み	平滑化フィルタ	画像処理結果出力
回路規模(slice)	31685	47	31685
遅延(ns)	3.862	9.491	3.862
HW クロック数	14400	14400	14400

ラプラシアンフィルタに比べて平滑化フィルタ処理部分の回路規模、遅延時間は小さくなっていることが分かる。これは処理がより単純になっているためである。画像読み込みと出力については、同じ画像を使用しているため回路規模、遅延時間は同じになった。次に平滑化フィルタによるノイズ除去の結果を図12に示す。

図１２：平滑化フィルタによるノイズ除去結果

4. ハード/ソフトの最適分割の検討

4.1 分割フロー

ハード/ソフト協調設計において、ハードウェアとソフトウェアの最適な設計空間を探索するのは重要であると同時に非常に難しい問題である。そこで本研究での分割の流れを図13に示す。

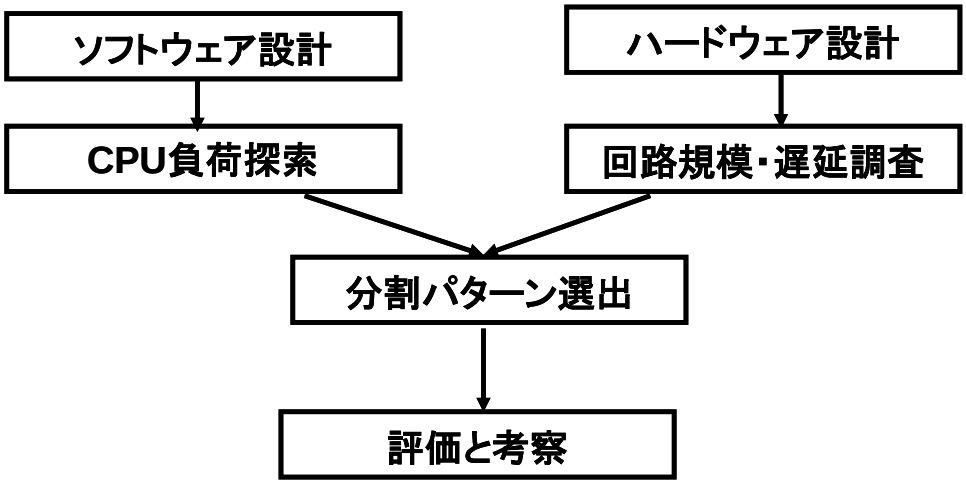

図１３：分割フロー

まず対象アプリケーションをC言語で記述してアルゴリズムの理解や正しい結果の確認などを行う。次にソフトウェア実行においてCPU負荷を計測し、どのモジュールやブロックが過負荷をかけているかを把握する。CPU負荷の計測には自作したクロックカウンタを使用する。その後、各モジュールのハードウェア化について検討してから、性能や規模について考慮した上で分割パターンを選び出す[2][3]。

4.2 ノイズ除去とエッジ検出の CPU 過負荷部の調査

ハードウェアとソフトウェアの分割探索のために、平滑化フィルタによるノイズ除去とラプラシアンフィルタによるエッジ検出をソフトウェアでシステムを実現して、モジュールごとのCPU負荷を計測した。計測に用いたデータは，120×120画素の画像を用いた。全体の処理に対する各機能ブロックのCPU負荷の割合をグラフにしたものを図１４に示す。

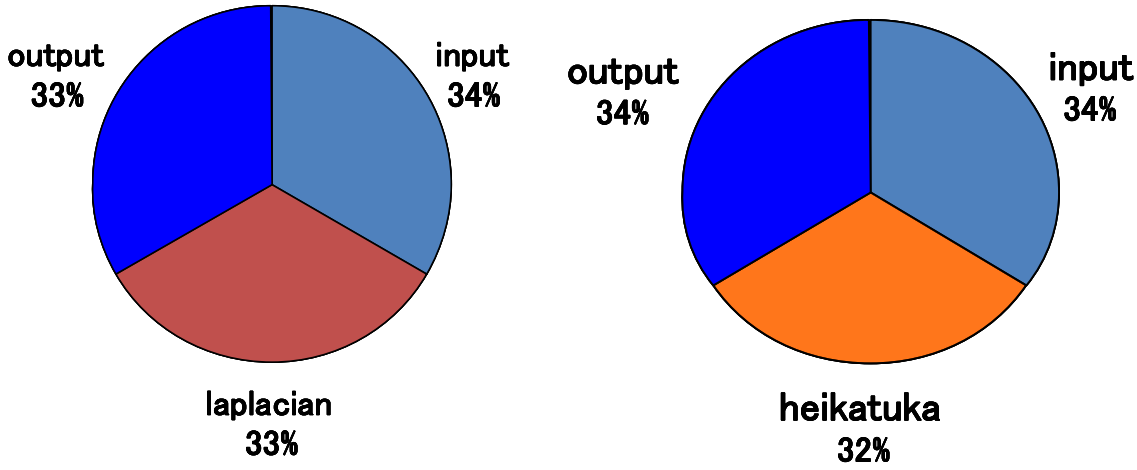

図１４：CPU 負荷の割合

Input、output、laplacian、heikatuka すべてのモジュールのＣＰＵ負荷割合はほぼ同じとなった。これは，どのモジュールも逐次的に処理されると考えられる。

4.3 ハードとソフトの切り分け

画像処理システムの構成を図１５に示す。システムではラプラシアンフィルタ、平滑化フィルタを選択できるように制御部分を追加し構成することを考えた。ハードとソフトの切り分けパターンを表７に示した．なお便宜上，表７のパターンの順は評価結果から得たデータを基に，表７の上方から回路規模の順で並べてある。

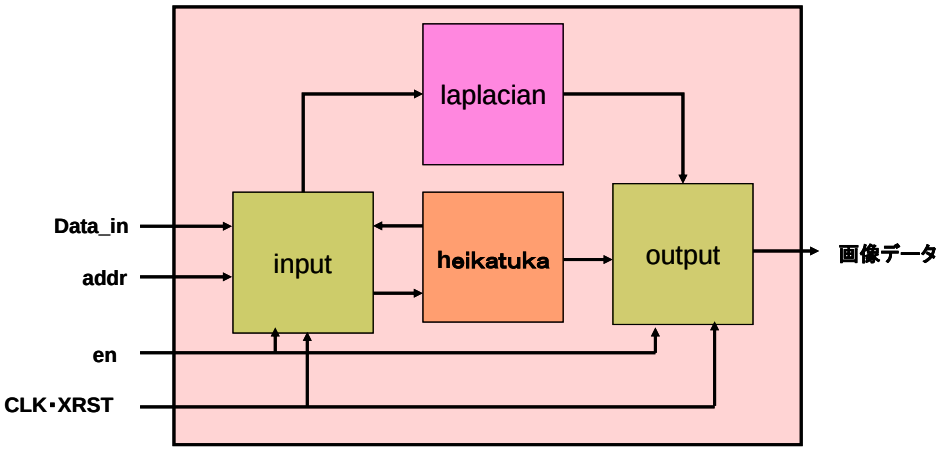

図１５：画像処理システムの構成

表７：機能ブロック単位での切り分けパターン

	ハードウェア処理部分	ソフトウェア処理部分	回路規模（slice）	速度（clk）
A		laplacian heikatuka 制御 input output	0	38400000
B	heikatuka	laplacian 制御 input output	47	×
C	laplacian heikatuka	制御 input output	295	×
D	input output	heikatuka laplacian 制御	31685	×
E	laplacian heikatuka 制御 input output		64025	14400

表７の A パターンで全処理をソフトウェアで実現し、評価の基準とする。B のパターンでは回路規模の小さい平滑化フィルタ演算部のモジュールをハードウェア化し、他の部分をソフトウェアで構成する。C のパターンでは次に大きいラプラシアンフィルタと平滑化フィルタをハードウェア化し、他の部分をソフトウェアで構成する。E のパターンでは一番回路規模の大きい入出力部分をハードウェア化し他の部分をソフトウェアで構成する。Ｆのパターンでは全体をハードウェアで構成する。Ｆのパターンでは速度は向上するが回路規模が大きくなるためコストが大きくなってしまう。ＣＰＵ負荷割合が全てのモジュールで同じなので、回路規模の小さいラプラシアンフィルタ演算モジュールと平滑化モジュールをハードウェア化する D のパターンがより効率的な回路を実現することができると予測できる。

今回の研究における問題点を挙げると、分割パターンの選出を行う指標が少ないことで、ソフトウェア実行の際の CPU 負荷しか見ていないので、考慮する情報が不足し

ている状態となっている。よって選出したパターンも 5 つと、かなりの数になってしまった。問題点としてもうひとつ、評価項目が少ないこともあるだろう。今回はクロックサイクル数、回路規模、ＣＰＵ負荷割合の 3 つしか上げていない。これでは実際に開発するときの制約には程遠い。 今後の課題として、対象アプリケーションをソフトウェアで動かすことがハード/ソフト最適分割探る第一段階となるが、そのソフトウェアの段階でどれだけ最適な分割パターンを見つけるための情報を収集できるかである[3]。

5. おわりに

本研究では、現在のシステムLSIの設計について考察し、ハード/ソフト協調設計における最適分割を検証するために、ラプラシアンフィルタと平滑化フィルタを使用した画像処理システムの設計をした。Ｃ言語で実行したときと比較して2667倍の速度向上が得られた。またモジュールごとに回路規模とＣＰＵ負荷を計測することによってハード/ソフトの分割案を考えることができ、ハード/とソフトの協調に関して理解を深めることができた。今後の課題として、Xilinx社の提供するソフト・マクロCPUであるMicroBlazeを用いて、Memec社のFPGAボードの上に実装することがある。

謝辞

本研究の機会を与えてくださり、貴重な助言、ご指導をいただきました山崎勝弘教授に深く感謝いたします。 また、本研究に関して様々な相談に乗っていただき、貴重な助言を頂いた和田氏、また共同研究者の松田氏、中山氏、伊藤氏及び色々な面で励ましを下さった高性能計算研究室の皆様に心より深く感謝いたします。

参考文献

[1] 小林優：ハードウェア記述言語の速習＆実践　入門 Verilog-HDL 記述,CQ 出版社,2002.

[2] 和田智行：Misty1 暗号回路の設計とハード/ソフト最適分割の検討,立命館大学理工学部情報学科卒業論文,2007.

[3]梅原直人:設計仕様解析によるハード/ソフト最適分割システムの実現と評価,立命館大学理工学研究科修士論文,2007.

[4]山田宏尚：はじめてのデジタル画像処理,技術評論者,2008.

[5]酒井幸一：デジタル画像処理入門,CQ 出版社.2002.

付録 A　ラプラシアンフィルタによるエッジ検出回路の Verilog-HDL 記述

//input モジュール

```
`timescale 1ns / 1ps
module ad_block( adck,
                 reset,
                 end_f,
                 a_out,
                 b_out,
                 c_out,
                 d_out,
                 e_out,
                 f_out,
                 g_out,
                 h_out,
                 i_out);

input            adck,reset;
output   [7:0]   a_out, b_out, c_out, d_out, f_out, g_out, h_out, i_out;
output   [8:0]   e_out;
output end_f;
parameter        LINENUM   = 14400;

reg end_reg;
reg      [32:0]   addr;
reg      [7:0]    mem[0:LINENUM-1];

initial $readmemh("lena-3.txt",mem );

always @(posedge adck or negedge reset)begin
        if (!reset) begin
                addr <= 0;
                end_reg <= 1'b0;
        end
        else if(addr==14280)begin
                end_reg <= 1'b1;
        end
        else begin
                addr <= addr + 1'd1;
        end

end
assign   end_f = end_reg;
assign   a_out = mem[addr];
assign   b_out = mem[addr + 10'd1];
assign   c_out = mem[addr + 10'd2];
assign   d_out = mem[addr + 10'd120];
assign   e_out = mem[addr + 10'd121];
```

```
assign   f_out = mem[addr + 10'd122];
assign   g_out = mem[addr + 10'd240];
assign   h_out = mem[addr + 10'd241];
assign   i_out = mem[addr + 10'd242];

endmodule
```

//ラプラシアンフィルタ演算モジュール

```
`timescale 1ns / 1ps
module add(a_in,
           b_in,
           c_in,
           d_in,
           e_in,
           f_in,
           g_in,
           h_in,
           i_in,
           q_out,
           adck,
           reset);

    input       [10:0]      a_in,b_in,c_in,d_in,f_in,g_in,h_in,i_in;
    input       [10:0]      e_in;
    input                   adck, reset;
    output      [7:0]       q_out;

    reg         [7:0]       r_reg;

    always@(posedge adck or negedge reset)begin

        if(!reset)
        r_reg = 0;

else        if (a_in+b_in+c_in+d_in+f_in+g_in+h_in+i_in>=e_in+e_in+e_in+e_in+e_in+e_in+e_in+e_in &&
                a_in+b_in+c_in+d_in-e_in-e_in-e_in-e_in-e_in-e_in-e_in-e_in-e_in+f_in+g_in+h_in+i_in>=255)

                r_reg = 255;

else        if (a_in+b_in+c_in+d_in+f_in+g_in+h_in+i_in<=e_in+e_in+e_in+e_in+e_in+e_in+e_in+e_in )

            r_reg = 0;

            else

r_reg=a_in+b_in+c_in+d_in-e_in-e_in-e_in-e_in-e_in-e_in-e_in-e_in-e_in+f_in+g_in+h_in+i_in;

end

assign q_out = r_reg;

endmodule
```

//output モジュール

```
module da_block(
                        adck,
                        reset,
                        we,
                        q_in,
                        outflag
                        );
input                adck,reset,we;
input   [7:0]     q_in;

parameter            LINENUM = 2048;

reg       [17:0]     addr;
reg       [32:0]     mem[1024:LINENUM-1];
input        outflag;
integer        i, mcd;

always @(posedge we or negedge reset)begin
          if (!reset)
                    addr <= 0;
          else if (we) begin

                    addr <= addr + 1'd1;
                    mem[addr+1024] <= q_in;
          end
end

//ファイルへの１０進出力
//呼び出し側で outflag を 1 にすると出力

initial begin
     wait( outflag );
     mcd = $fopen( "lena_5.pgm");
     for ( i=0; i<LINENUM; i=i+1 )
          $fdisplay(mcd, "%0d", mem[i] );
          $fclose(mcd);
end
endmodule
```

付録 B　平滑化フィルタによるノイズ除去回路の Verilog-HDL 記述

//平滑化モジュール

```
`timescale 1ns / 1ps

module kajyuu(a_in,

               b_in,

               c_in,

               d_in,
```

```
            e_in,
            f_in,
            g_in,
            h_in,
            i_in,
            q_out,
            adck,
            reset);

    input       [10:0]      a_in,b_in,c_in,d_in,f_in,g_in,h_in,i_in;
    input       [10:0]      e_in;
    input                    adck, reset;
    output      [7:0]       q_out;

    reg         [31:0]      r_reg;


    always@(posedge adck or negedge reset)begin

            if(!reset)
                r_reg = 0;

            else
                r_reg = a_in+b_in+c_in+d_in+e_in+f_in+g_in+h_in+i_in ;

end

        assign q_out = (r_reg >> 3) ;

endmodule
```