高位合成を用いた動的背景差分による動体検知機能を備えた組み込み機器の開発

新山田 公平[1]　山脇 彰[1,a)]

概要：一般的にハードウェア処理は高性能かつ低消費電力であるため，組み込みシステムに適している．高位合成は高級言語をハードウェア記述言語へ自動変換するものでハードウェア開発を容易にする．本研究では動体検知機能を備えた組み込み機器の開発にあたって，動的背景差分処理やその他サブモジュールを高位合成によりハードウェア化した．その際，高性能なハードウェアを生成させるための高位合成向き記述を検討し，その効果を確かめた．そして各ハードウェアを組み合わせてソフトウェアで最適制御することで，処理単位でのパイプライン化に成功した．その結果，画像サイズ QVGA，時系列画像数 4 のときに，194fps の処理性能を持つシステムを開発できた．

Development of an Embedded Device with Motion Detection by Dynamic Background Subtraction Using High-level Synthesis

Shinyamada Kohei[1]　Akira Yamawaki[1,a)]

Abstract: In general, hardware processing is suitable for embedded systems because of its high performance and low power consumption. High-level synthesis automatically converts high-level languages into hardware description languages, which facilitates hardware development. In this study, dynamic background subtraction and other sub-modules were converted to hardware by high-level synthesis to develop an embedded system with a motion detection function. In this study, we examined the high-level synthesis orientation description and confirmed its effectiveness. By optimally controlling each hardware by software, we succeeded in pipelining each processing unit. As a result, when the image size was QVGA and the number of time-series images was 4, 194 fps was achieved.

1. 序論

組み込み画像処理システムの市場規模は年々拡大傾向にあり，今後もその傾向が続くことが予想される [1]. このような市場で高い競争力とシェアを得るためには，製品の早期開発と市場への早期投入が必要不可欠である．

画像処理の手法は，ハードウェア処理とソフトウェア処理に大別される．一般的に，ハードウェア処理はソフトウェア処理に比べて高性能かつ低消費電力であるため，組み込みシステムに適している [2], [3]．ハードウェア処理の実現方法として，Application Specific Integrated Circuit (ASIC) と，Field Programmable Gate Array (FPGA) が挙げられる．ASIC は専用回路を実装するもので，回路の修正は不可能である．また，新規開発の初期投資も大きい．それに対して FPGA は，設計者が内部の回路を自由自在に書き換えることができ，初期投資も小さい．それゆえ，FPGA は製品の試作段階などにおいて特に有用なデバイスである．

しかしながら，従来のハードウェア開発は C などの高級言語でアルゴリズムを設計した後，それをハードウェア記述言語へ手作業で変換する必要があった．これには多くの時間と労力が必要である．それらの問題点を解決するためのツールが高位合成ツールである．これは C などの高級言語を VHDL などのハードウェア記述言語へ自動で変換するものである [4], [5], [6]．このツールを用いることで，

[1] 九州工業大学 (Kyushu Institute of Technology)
1-1 Sensui-cho, Tobata-ku, Kitakyushu-shi, Fukuoka, 804-8550, JAPAN
[a)] yama@ecs.kyutech.ac.jp

アルゴリズムの変更に柔軟に対応することができ，開発期間も大幅に削減できる．ただし，高位合成ツールにも弱点は存在する．ハードウェアの特性を考慮したソフトウェア記述を行わないなら，非常に大規模で低速なハードウェアが生成されてしまい，ハードウェア処理の恩恵を受けられない．

本研究で対象とする画像処理は，動的背景差分処理である．背景差分処理は古典的な手法ではあるが，深層学習などの機械学習が適用できない非力な環境における移動物体の検出に用いられている [7]．さらに，深層学習を用いた監視カメラシステムにおいて，学習させるためのデータのための前処理などに用いられている．これらのことから現在でも有効なアルゴリズムであると考えられる．背景差分法は重要なアルゴリズムではあるが，背景の時間的な変化に弱いため，それを解決するために動的背景差分法 [8], [9] が研究されている．しかしながら，動的背景差分法に関しては，高位合成によるハードウェア化が試みられていない．加えて，アルゴリズム通りに単純にプログラムを記述して，高位合成によってハードウェア化したとしても，高性能なハードウェアが生成できるわけではない．

また，著者らの知る限り，動的背景差分自体だけでなく，それを用いた動体検知機能を備えた実組み込み機器も，高位合成によって開発した事例は存在しない．以上のことから，本研究では動的背景差分を用いた動体検知機能を備えたカメラ搭載組み込みシステムの主要部分を高位合成によって開発する．主要部分とは，カメラからの画像をトリミングするモジュール，動的背景差分処理を行うモジュール，動体を検知した際に画面へ表示するモジュールである．これらの処理をすべてまとめたソフトウェアを記述し，1つのハードウェアを生成してしまうと，大きな処理単位でのパイプラインが難しくなり，サブ処理がボトルネックとなりうる．そこで本研究では，3 つのモジュールを高位合成を用いて個別に開発し，それら 3 つのハードウェアをソフトウェアで最適制御する．そしてシステム全体で見たときに，処理ごとにパイプライン化できるように構成する．

以降，高位合成を用いて開発するカメラ搭載組み込み画像処理システムの全体像を示し，所望の機能を実現するためのサブ処理の関係と，どの部分を高位合成により開発するか明らかにする．その後，高位合成による高性能中央値ベース動的背景差分ハードウェアの開発を行う．次に，カメラやディスプレイに対するデータ転送に関するサブモジュールを高位合成によって開発する．そして，それら高位合成によって開発した各ハードウェアモジュールと，実カメラを組み合わせたリアルタイム処理システムを開発する．実験では，動的背景差分ハードウェアの実現にあたって，高性能化を目的に導入した各種工夫の効果や，システム全体での性能や電力効率，およびフレームレートなどを評価する．

2. 提案カメラ搭載組み込み画像処理システム

2.1 動的背景差分法

本設では動的背景差分法について説明する．はじめに，単純な背景差分法の原理について述べ，その後本研究で使用する中央値ベース動的背景差分法について説明する．

2.1.1 単純背景差分法の原理と適用事例

背景差分処理とは，観測画像と事前に取得しておいた背景画像とを比較し，背景画像には存在しない物体を抽出する処理のことを言う．ここで，背景画像を B，処理対象画像を I，差分画像を D とする．ただし，本研究で処理する画像はすべてグレースケール画像とする．画素 p における画素値をそれぞれ $B(p)$, $I(p)$, $D(p)$ と表し，しきい値を th とする．その時 $D(p)$ は以下の式で与えられる．全画素に対して，(1) 式を適用することで，差分画像を取得できる．

$$D(p) = \begin{cases} 1 & abs(B(p) - I(p)) \geq th \\ 0 & abs(B(p) - I(p)) < th \end{cases} \quad (1)$$

本研究における画像処理ではグレースケール画像を用いるが，カメラ撮影画像はカラー画像である．そこで，カラー画像からグレースケール画像へ変換する手法について説明する．カメラからのカラー画像は RGB それぞれ 8bit で，深度 24bit である．NTSC 加重平均法を用いてグレースケール化した際の明度を Y とすると，変換式は (2) 式のようになる．この Y で RGB それぞれを置換することによってグレースケール画像が得られる．

$$Y = 0.299R + 0.587G + 0.114B \quad (2)$$

このような差分処理は，監視カメラシステムなどへ応用可能である．しかしながら，単純背景差分法にはデメリットが存在する．単純背景差分法において前景画像は固定である．そのため背景が変化する環境では，正しい差分画像を取得することができない．この問題を解決するために，動的背景差分法が研究されている．

2.1.2 中央値ベース動的背景差分法の原理

動的背景差分法とは，前景画像を動的に更新していき，その画像と入力画像との差分を取る背景差分法のことである．これを用いることで，背景の y 変化が差分画像に与える影響が小さくなる．動的な背景画像の生成手法として，逐次更新法，統計的更新法，中央値法などが存在するが，本研究では中央値法をアルゴリズムとして採用する．本研究においては，中央値法を用いた動的背景差分を，中央値ベース動的背景差分法と呼ぶ．

中央値ベース動的背景差分法について説明する．まず，前景画像生成のために，任意の N 枚の時系列画像をメモリ上に格納しておく．そして同一箇所のピクセル群のデータをソートし，中央値を取得する．これを背景画像のピクセ

ル値として使用する．これらの処理を全ピクセルに対して行うことで，前景画像を生成できる．そして，生成した前景画像と現在画像との差分を 2.1.1 で示したアルゴリズムを用いて処理することで，1 枚の白黒差分画像を取得することができる．先行研究より，本画像処理においては高位合成によって生成されるハードウェアモジュールが高速かつ小回路規模であることがわかっているため，ソートアルゴリズムとしてバブルソートを使用する [10].

2.2 提案システム

2.1 で述べた，中央値ベース動的背景差分法を搭載した組み込み機器を実現するにあたって提案するシステムは，図 1 のような構成になる．本システムはカメラ，各種画像処理関連ハードウェア，ディスプレイから構成される．

2.2.1 カメラインターフェース

本研究で使用するカメラインターフェースは，既に設計されている資産を用いる．以下，インターフェースの動作について説明する．カメラからは，1280 × 1024 のサイズの撮影画像が約 12fps で FPGA に入力される．1 画素あたり RGB がそれぞれ 5bit，6bit，5bit の 16bit カラー画像である．カメラから得られた RGB に対して，下位ビットに 0 を埋め込み，RGB それぞれを 8bit に拡張する．また，本インターフェースは，カメラバッファに画像データを書き込む．カメラバッファは 2 つ用意してあり，カメラは 2 つのバッファに交互に書き込む．このようにすることによって，一方のバッファに書き込みが完了した直後から，そのデータを用いた画像処理を開始することが可能になる．バックグラウンドでもう一方のバッファにカメラからの画像を格納することになるため，カメラ画像の書き込み待ちによる処理速度低下を低減することができる．

2.2.2 提案システムの動作フロー

まず，前述の通りカメラインターフェースが，撮影画像をカメラバッファに格納する．カメラバッファに格納された画像のサイズは 1280 × 1024 である．これを指定のサイズで切り取るためのハードウェアが図 1 中の HLS HW for Trimming である．

トリミングハードウェアは，カメラバッファに格納された画像をトリミングし，リングバッファへ順に格納していく．中央値ベース動的背景差分法では，N 枚の時系列撮影データから背景画像を生成する．ただし本研究で作成する画像処理ハードウェアでは，現在画像を含めた N+1 枚の画像データから背景画像を生成する．しかしながら，これは中央値ベース動的背景差分法の本質には影響を与えないと考えられる．例えば，動体がある場合はその領域近辺の画素値が大きく変化するから，N+1 枚の画素値をソートすれば，動体が含まれるフレームが中央値画像となることはない．

リングバッファの領域は N+2 枚分用意する．まず，N 枚分の領域は過去の時系列撮影データを格納しておくためである．そして 1 枚が現在画像格納用，もう 1 枚はトリミングハードウェアによるバックグラウンド書き込み用である．Buffer_0，Buffer_1，…，Buffer_N+1 の領域に対して，トリミングハードウェア，画像処理ハードウェア，画面表示ハードウェアが適切に入出力を行うようにシステム全体を制御する．また，リングバッファの末尾にデータが格納されたあとは，再び先頭からデータの書き込みを開始する．これにより，使用メモリ量を必要最低限にできる．

続いて，リングバッファに格納されたデータを，動的背景差分ハードウェアで処理する．動的背景差分ハードウェアは，リングバッファに格納された N+1 枚のデータを読み取り，ピクセルごとに中央値を計算し，それらを集めた背景画像を生成する．動的背景差分ハードウェアへは，現在画像がリングバッファの何番目に格納されているのかを引数として渡す．それをもとに，リングバッファに格納された現在画像との差分を計算し，その画像データをディスプレイへ表示する．

これまでに説明した，トリミングハードウェアと画像処理ハードウェアのリングバッファへのアクセスについて，図 2 に示す．時系列撮影データ数は N=4 として説明する．まず，画像処理対象領域が Buffer_0〜Buffer_4 のとき，画像処理ハードウェアに渡すリングバッファの先頭番地は Buffer_0，現在画像の先頭番地は Buffer_4 である．また，トリミングハードウェアにわたすバックグラウンド書き込み領域先頭番地は Buffer_5 である．このとき，トリミングハードウェアは，カメラバッファに格納された画像をトリミングし，Buffer_5 へ書き込む．それと同時に，画像処理ハードウェアが Buffer_0〜Buffer_4 までの画像を処理する．この一連の流れを繰り返すことで，効率的にメモリを使用できる．また，ハードウェアを並列動作させることができる．

動的背景差分ハードウェアは，1 枚画像を出力するごとに，画素中の白ピクセルの総数を返すように設計する．前述の通り白ピクセルの総数は差分計算した結果しきい値を超えたもの，すなわち移動物体の有無を表す値である．この値が，あらかじめ設定したしきい値を超えた場合は，ディスプレイ出力ハードウェア（図中 HLS HW for Display Output）を起動させ，リングバッファ内に格納された該当現在画像を出力する．これにより，移動物体が存在するときのみ，カメラからの画像を出力することが可能となり，動体検知機能を実現できる．ディスプレイ出力を行ったのち，移動物体が存在しなくなれば，すなわち白ピクセルの総数が連続してしきい値を下回る場合には，表示を停止させる．

これまでに説明したハードウェア（トリミング，動的背景差分，ディスプレイ出力）はすべて高位合成により作成する．以降の章では，各ハードウェアを高位合成で生成す

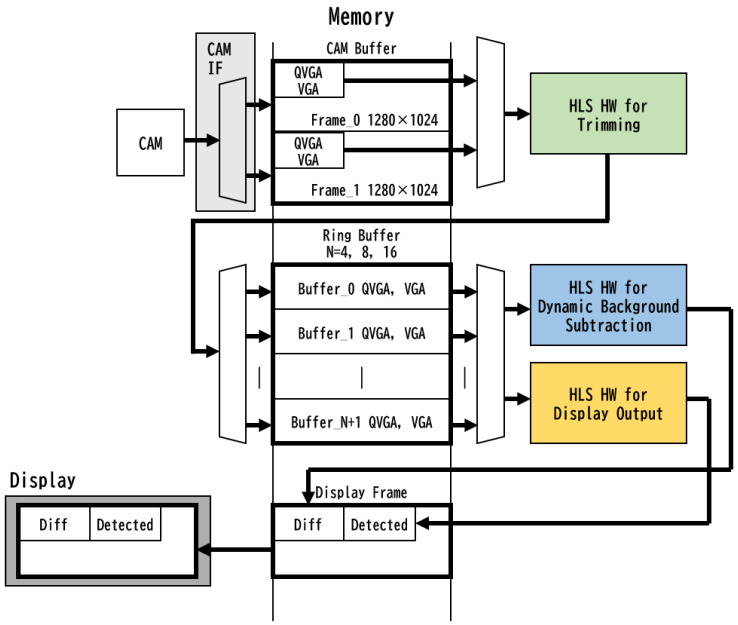


図 1 提案システムの全体構成
Fig. 1 Overall Structure of the Proposed System.

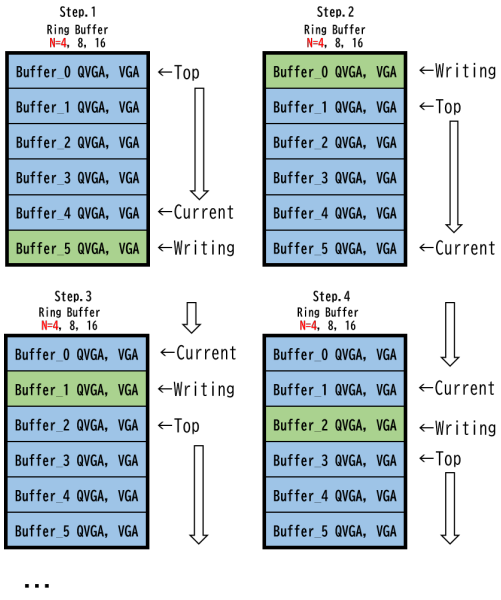


図 2 リングバッファへのアクセス
Fig. 2 Accessing the Ring Buffer.

るにあたり工夫したソフトウェア記述法とその効果などを論ずる．また，これらのハードウェアの協調は FPGA ボード上に組み込まれたプロセッサによって行われる．これを実現するために，ハードウェアに渡す引数の設定や，ハードウェア実行順序の制御プログラムなどを別途作成する．

3. システム各部の高位合成用ソフトウェア開発

本章では，提案システム各部の高位合成向けソフトウェア開発に関して説明する．

3.1 中央値ベース動的背景差分法

中央値ベース動的背景差分法は，過去の N 枚の撮影画像における各画素を取り，N 個の画素群の中央値を取ってそれを新しい背景画像とし，現在の撮影画像との差分を計算するアルゴリズムである．以降，中央値ベース動的背景差分法のハードウェア化について検討する．

3.1.1 純粋ソフトウェア

純粋ソフトウェアを高位合成ツールによりハードウェア化するための記述について説明する．ただしリングバッファを想定して，内部でインデックス計算を行っている．純粋ソフトウェアプログラムを図 3 に示す．

3.1.2 ラインバッファを導入した記述

純粋ソフトウェアの問題点として，時系列の N 枚の画像データから中央値を用いた背景画像を生成する際に，不連続なメモリアクセスを行っている．そのため，バースト転送と呼ばれる，データを 1 クロックで一括転送する機能は使用できない．この問題を解決し，連続したメモリアクセスを可能にするために，ラインバッファを導入する [11]．これにより，時系列画像を行ごとに転送できるようになり，バースト転送が可能になる．再構築したソフトウェアを図 4 に示す．

3.2 画像データ転送機構の高位合成ソフトウェア

3.1 では，提案システムの画像処理部を高位合成により開発した．本節では，画像データ転送機構の高位合成向けソフトウェア開発について説明する．

3.2.1 トリミング部

カメラがメモリに書き込む画像のサイズは 1280 × 1024

```
uint32_t bgs_med
(u32 *data_base, u32 *disp_frame, int top,
 u8 th_bgs, int x0, int y0){
#pragma … // Port Config

  u8 buff_sort[FRAME_NUM];

  u8 pix_diff;

  int i, j, k;
  int idx;

  u32 cnt = 0;

  for(i=0; i<IMG_H; i++){
    for(j=0; j<IMG_W; j++){
    #pragma HLS PIPELINE II=1
      for(k=0; k<FRAME_NUM; k++){
        idx = (top+k)%(FRAME_NUM+1);
        buff_sort[k] = PastPIX
      }
      pix_diff = CurrentPIX - GetMedian(buff_sort);

      if(pix_diff >= th_bgs){
        pix_diff = 0xff; cnt++;
      }else{
        pix_diff = 0x00;
      }

      disp_frame[(i+y0)*DISP_W + j + x0]
        = To_u32(pix_diff);
    }
  }

  return cnt;

}
```

図 3 動的背景差分 純粋ソフトウェア
Fig. 3 Pure Software of Dynamic Background Subtraction.

ピクセルである．この画像から任意の領域を切り出して，リングバッファ部へ格納するためのハードウェアがトリミング部である．本ハードウェアを高位合成で作成するにあたり使用したソフトウェアを図 5 に示す．

3.2.2 ディスプレイ出力部

動的背景差分ハードウェアは差分結果をディスプレイに表示すると同時に，白ピクセルの総数を制御プログラムへ通知する．その値がしきい値を超えた場合には，本ハードウェアを起動させ，リングバッファ内に格納されている現在画像を表示する．本ハードウェアを高位合成で作成するにあたり使用したソフトウェアを図 6 に示す．

4. 実験及び考察

4.1 実験環境

今回開発した実カメラシステム写真を図 7 に示す．使用した FPGA ボードは Digilents 社の ZYBO Z7-10 である．このボードに搭載されている FPGA は Xilinx 社の Zynq-7010 で，ARM 社の組み込みプロセッサ Cortex-A9 プロセッサが統合されている．図 1 のメモリは FPGA はボード上の DDR3 SDRAM(1GB) に置かれる．

```
uint32_t bgs_med
(u32 *data_base, u32 *disp_frame, int top,
 u8 th_bgs, int x0, int y0){
#pragma … // Port Config

  u8 buff_sort[FRAME_NUM];
  u8 buff_line[FRAME_NUM][IMG_W];

  #pragma HLS ARRAY_PARTITION var=buff_sort comp dim=0
  #pragma HLS ARRAY_PARTITION var=buff_line comp dim=1

  u8 pix_diff;

  int i, j, k;
  int idx;

  u32 cnt = 0;

  for(i=0; i<IMG_H; i++){
    for(k=0; k<FRAME_NUM; k++){
      idx = (top+k)%(FRAME_NUM+1);
      for(j=0; j<IMG_W; j++){
      #pragma HLS PIPELINE II=1
        buff_line[k][j] = PastPIX_line;
      }
    }

    for(j=0; j<IMG_W; j++){
    #pragma HLS PIPELINE II=1
      for(k=0; k<FRAME_NUM; k++){
        buff_sort[k] = buff_line[k][j];
      }
      pix_diff = CurrentPIX - GetMedian(buff_sort);

      if(pix_diff >= th_bgs){
        pix_diff = 0xff; cnt++;
      }else{
        pix_diff = 0x00;
      }

      disp_frame[(i+y0)*DISP_W + j + x0]
        = To_u32(pix_diff);
    }
  }

  return cnt;

}
```


図 4 動的背景差分 再構築ソフトウェア
Fig. 4 Restructured Software of Dynamic Background Subtraction.

使用したカメラは Omnivision 社の CMOS カメラモジュール OV9655 である．OV9655 からは 1280 × 1024 のサイズの撮影画像が，2.2.1 で述べたカメラインターフェースを介して約 12fps で FPGA に入力される．FPGA ボードには HDMI 経由でディスプレイが接続され，図 1 のディスプレイ出力部を介して，画像処理結果を目視確認できる．

FPGA の動作周波数は 100MHz で，組み込みプロセッサの動作周波数は 650MHz である．本研究で使用する高位合成ツールは Xilinx 社の Vivado HLS 2018.3，FPGA の実装ツールは Vivado 2018.3 である．

4.2 評価手法

評価手法は，処理速度，ハードウェア規模を考慮した電

```
void trim_cam_img
( u32 *src_cam, u32 *dst_trim, int size ){
#pragma … // Port Config

  int i, j;
  int trim_w, trim_h;
                    Select Image Size
switch(size ){
  case 1: // 1 -> QVGA
    trim_w = QVGA_W;
    trim_h = QVGA_H;
    break;
  case 2: // 2 -> VGA
    trim_w = VGA_W;
    trim_h = VGA_H;
    break;
  default: // default -> QVGA
    trim_w = QVGA_W;
    trim_h = QVGA_H;
}

  // Copy trimmed image to "dst_trim"
  for( i = 0; i < trim_h; i++ ){
  #pragma HLS LOOP FLATTEN OFF
    for( j = 0; j < trim_w; j++ ){
    #pragma HLS PIPELINE II=1
      dst_trim[ j + i*trim_w ]
        = src_cam[ j + i*CAM_W ];
    }
  }          Write Trimmed Image
             to Ring Buffer
}
```

図 5 トリミングハードウェア生成用ソフトウェア
Fig. 5 Software for Generating Trimming Hardware.

```
void disp_ring_buf
(u32 *p_data, int size, u32 *disp_frame, int x0, int y0){
#pragma … // Port Config
  int i, j;
  int w, h;
                    Select Image Size
switch( size ){
  case 1: // 1 -> QVGA
    w = QVGA_W;
    h = QVGA_H;
    break;
  case 2: // 2 -> VGA
    w = VGA_W;
    h = VGA_H;
    break;
  default: // default -> QVGA
    w = QVGA_W;
    h = QVGA_H;
}

  for(i=0; i<h; i++){
  #pragma HLS PIPELINE II=1
    for(j=0; j<w; j++){
      disp_frame[ ( i + y0 ) * DISP_W + j + x0 ]
        = p_data[ i * w + j ];
    }
  }          Write to Display Frame
}
```

図 6 ディスプレイ出力ハードウェア生成用ソフトウェア
Fig. 6 Software for Generating Display Output Hardware.

力効率，最大フレームレートである．

処理速度，すなわち実行時間は以下の式によって定義される．

$$Exec\,Time[s] = \frac{Total\ number\ of\ Clocks\ [clk]}{Clock\ Frequency\ [Hz]} \quad (3)$$

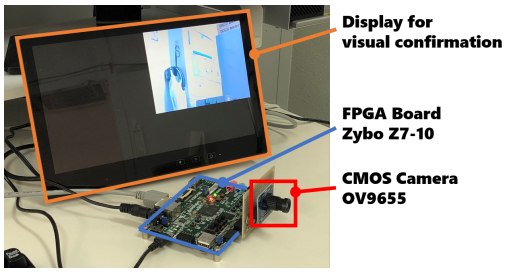


図 7 開発カメラ組み込みシステム
Fig. 7 Developed Embedded Camera System.

動作電力向上比 (Operating Power Improvement Ratio, OPIR) は以下の式によって定義される．

$$OPIR = \frac{CPU\ Exec\,Time\ [s] \times CPU\ Freq\ [Hz]}{HW\ Exec\,Time\ [s]\ \times HW\ Freq\ [Hz]\ \times HW\ Scale} \quad (4)$$

ただし $HW\ Scale$ は以下のように計算される．

$$HW\ Scale = \frac{SLICE(*)}{SLICE(Ref)} \times \frac{BRAM(*)}{BRAM(Ref)} \quad (5)$$

- SLICE(*) : SLICE of Hardware for comparison
- SLICE(Ref) : SLICE of Reference Hardware
- BRAM(*) : BRAM of Hardware for comparison
- BRAM(Ref) : BRAM of Reference Hardware

動作電力向上比は，HW 処理が SW 処理に対して，電力効率の観点でどの程度優れているかを表す指標である．また，この式の分母にはハードウェア規模が組み込まれている．ハードウェア規模が大きくなると，回路はより多くの電力を消費するため，ハードウェア規模の小さいハードウェアを生成すれば電力効率が向上する．上記の式ではハードウェア規模として，SLICE と BRAM を採用した．

次に，カメラの性能を無視した，システムが達成しうる最大フレームレートは，以下の式によって定義される．式 6 の T_0 はシステムが 1 枚の画像を出力するまでの時間であり，式 3 を用いて計算できる．T_0 の逆数を求めることにより，システムが達成しうる最大フレームレートを計算 できる．

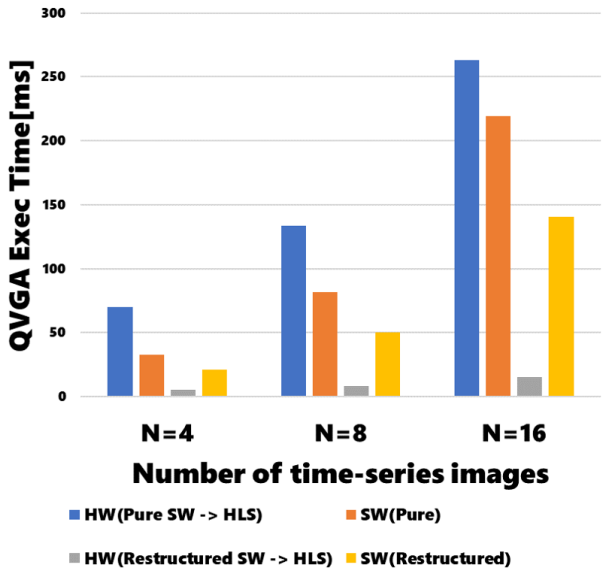
$$FR_{MAX} = \frac{1}{T_0} \quad (6)$$

4.3 中央値ベース動的背景差分法のハードウェア評価

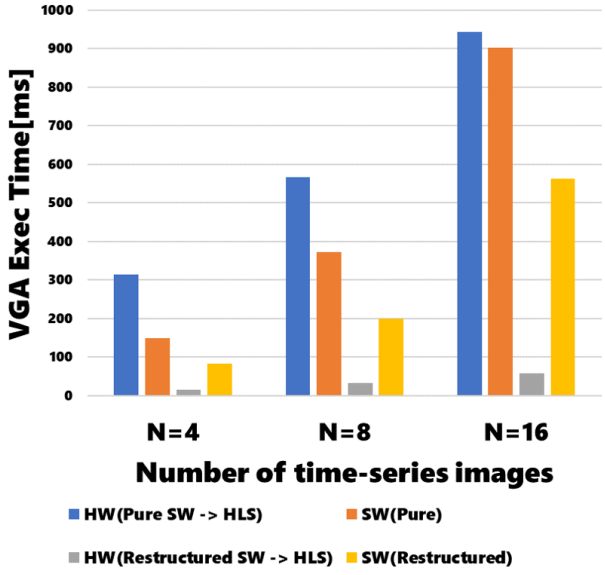
実行時間に関して，図 8 に結果をまとめる．

まず，純粋ソフトウェアに関する比較を行う．純粋ソフトウェアから生成したハードウェアの実行時間と，組み込みプロセッサによる純粋ソフトウェアの実行時間を比較すると，ソフトウェア処理のほうが高速であった．これは，純粋ソフトウェアがハードウェア化に適していなかったからだと考えられる．

次に，ハードウェア構成を考慮して再構築したソフト



(a) QVGA.



(b) VGA.

図 8 動的背景差分ハードウェア 実行時間
Fig. 8 Execution Time of Dynamic Background Subtraction HW.

ウェアに関する比較を行う．再構築した記述はソフトウェアレベルでも高速化の効果が見られた．ハードウェアレベルでは，顕著な高速化の効果が見られた．純粋ソフトウェアから生成したハードウェアと，再構築したソフトウェアから生成したハードウェアの処理時間を比較すると，画像サイズや時系列画像数にもよるが，約 13〜17 倍程度の性能向上が見られた．

次に，動作電力向上比に関して図 9 に結果をまとめる．これは，ソフトウェア処理に対するハードウェア処理の電力効率を示すものである．基準となるハードウェアは，QVGA，VGA ともに時系列画像 N=4 のハードウェアである．グラフからわかるように，すべてのケースにおいて

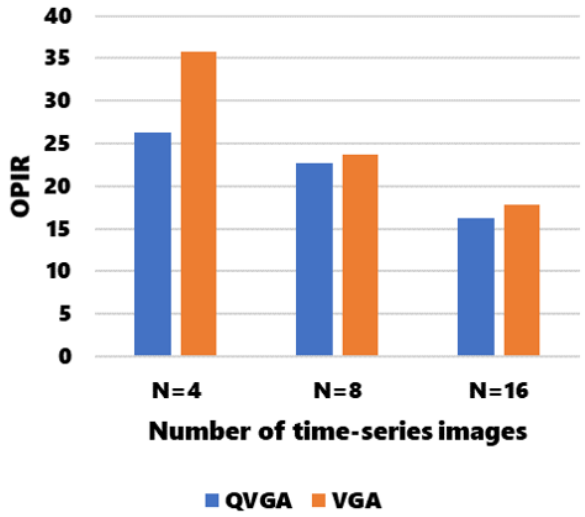


図 9 動作電力向上比
Fig. 9 Operationg Power Improvement Ratio.

ハードウェアは非常に優れた電力効率となっている．画像サイズ VGA，N=4 のときは電力効率が約 35 倍となった．

4.4 リアルタイム処理システム

実カメラの性能を無視した，最大フレームレートを図 10 に示す．第 2 章で述べたように，トリミングハードウェア，動的背景差分処理ハードウェア，画面表示ハードウェアが同時並列に動くシステムを構築したため，最も処理時間の長い動的背景差分ハードウェアの実行時間がシステム全体の実行時間となる．画像サイズが小さく，時系列画像数が少ないと処理は高速になる．最も高速な場合は，画像サイズ QVGA，時系列画像数 N=4 のときに，194fps となった．最も低速な場合は，画像サイズ VGA，時系列画像数 N=16 のときに，17fps となった．

実際に本システムが動作している様子を図 11 に示す．画像サイズは VGA，時系列画像数は 8，差分のしきい値は 30，動体検出のしきい値は 10（総ピクセル数の 10%が白くなったら表示）とし，実際に人物がカメラの前を横切った場合の動作を検証した．図 11(a) のように白ピクセルの総数が少ない，すなわち動体であると検知されない場合は，右側の検知領域には何も表示されない．しかし，図 11(b) のように白ピクセルの総数が多い，すなわち動体であると検知される場合は，右側の検知領域にリングバッファ内の現在カラー画像が表示された．以上の結果は，動的背景差分処理を用いて動体検知機能が実現できており，本システムが正しく動作しているといえる．

5. 結論

本研究では，中央値ベース動的背景差分機能を備えたカメラ搭載組み込みシステムを，高位合成により開発した。高位合成により，迅速で柔軟な開発を可能とした。

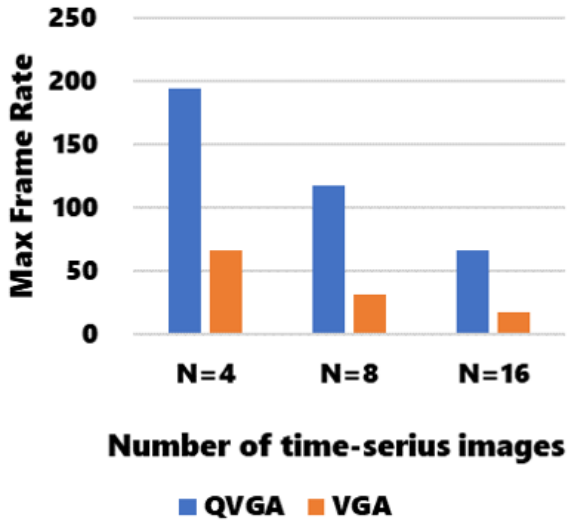
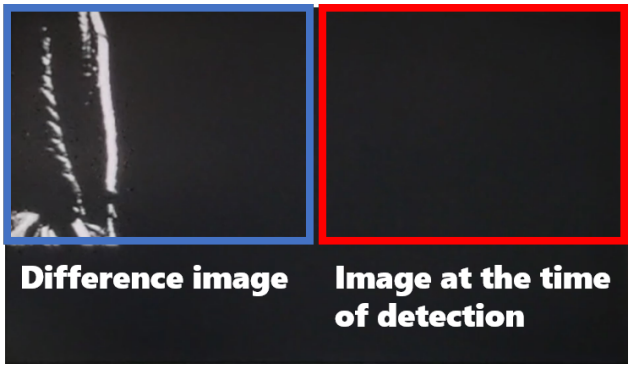


図 10 実カメラの性能を無視した最大フレームレート
Fig. 10 Maximum Framerate Ignoring the Performance of the Actual Camera.



(a) Undetected.

(b) Detected.

図 11 動作デモ
Fig. 11 Operation Demo.

まず，リングバッファという概念をシステムに導入し，メモリの使用効率向上と，ハードウェアの並列動作を可能にした。

次に，これまでに我々が開発してきた，中央値ベース動的背景差分法の高位合成向きプログラムをリングバッファ用に最適化し，システムに組み込んだ。高位合成向きソフトウェアによるハードウェアは，処理速度や電力効率という観点で，画像サイズや時系列画像数に関わりなく組み込みプロセッサよりも優れていた。画像サイズ QVGA，時系列画像数 4 のときに最大フレームレートが 194fps となった。また，画像サイズ VGA，時系列画像数 4 のときに電力効率が 35 倍となった。

ただし，今回の実験で使用したカメラの性能が約 12fps であるため，本試作機の最大フレームレートは約 12fps となる。本システムの性能を最大限引き出すことのできる，高速なカメラを使用すれば，さらなるフレームレートの向上が見込まれる。

参考文献

[1] “Computer Vision Market Size, Share & Forecast 2020-2026”, kbv research, Oct. 2020, https://www.kbvresearch.com/computer-vision-market/, Accessed 1 Feb. 2022
[2] “Stacking Up Software to Drive FPGAs Into the Datacenter”, Next Platform, https://www.nextplatform.com/2016/11/20/stacking-software-drive-fpgas-datacenter/, Accessd 1 Oct. 2019
[3] “GPU vs FPGA Performance Comparison”, Berten, https://www.bertendsp.com/gpu-vs-fpga-performance-comparison/, Accessd 1 Oct. 2019
[4] “High-Level Synthesis with LegUp”, http://legup.eecg.utoronto.ca/, Accessd 10 Nov. 2021
[5] “High-Level Synthesis - MATLAB and Simulink”, Mathworks, https://jp.mathworks.com/discovery/high-level-synthesis.html, Accessd 10 Nov. 2021
[6] Yuto Ishikawa, etc. : “Hardware Acceleration with Multi-Threading of Java-Based High Level Synthesis Tool”, HEART2017 Proceedings of the 8th International Symposium on Highly Efficient Accelerators and Reconfigurable Technologies, No.8, pp.1-6, 2017
[7] 初田慎弥，大野真史，孟林，泉知論，“獣害対策のための監視カメラ向けアライグマ検出器の構築と評価”, The Journal of the Institute of Image Electronics Engineers of Japan, Vol.48, No.2, pp.237-246, 2019
[8] M. P. A. P. Rita Cucchiara, “Detecting Moving Objects, Ghosts, and Shadows in Video Streams”, IEEE Irans-actions on Pattern Analysis and Machine Intelligence, Vol.25, No.10, pp.1337-1342, 2003
[9] 篠崎眞太郎，中島克人，“実時間物体追跡に適した動的背景推定法と背景差分法”, 知能と情報（日本知能情報ファジィ学会誌）, Vol.24, No.2, pp.637-647, 2012
[10] Kohei Shinyamada, Akira Yamawaki : “Effect of Sorting Algorithms on High-level Synthesized Image Processing Hardware”, Proceedings of the 8th IIAE International Conference on Intelligent Systems and Image Processing 2021, pp. 16-20, 2021
[11] Kohei Shinyamada, Akira Yamawaki : “Development of High Performance Hardware by High-Level Synthesis of Median-Based Dynamic Background Subtraction Method with Multiple Line Buffers”, 6.Dec.2021 Technical Meeting on Systems, pp. 23-28, 2021