令和3年度 修士論文

FPGAを用いたコンシューマーゲーム機拡張システムの開発
～リングフィットアドベンチャーを例として～

電気通信大学　情報理工学研究科
情報・ネットワーク工学専攻
コンピュータサイエンスコース

学籍番号　2031096
氏名　滝川 潤
指導教員　成見 哲
副指導教員　佐藤 証

令和4年1月28日

概 要

コロナ禍もあり体を動かすフィットネスゲームに一定の需要があり，その中でも任天堂リングフィットアドベンチャーは専用のリングを用いることで豊富な運動種目に対応していることで人気がある．しかし，他のフィットネスゲームと比較すると運動モチベーションを向上させる部分で不十分なところがある．本研究では，本来機能拡張が難しいコンシューマーゲーム機である任天堂 Switch のリングフィットアドベンチャーを機能拡張して，運動モチベーションを向上させる機能を追加する．本システムでは，FPGA を用いてゲーム機からの HDMI 映像出力を解析し，文字認識機能，コンボ数表示機能，ピンチ応援機能を実装した．文字認識機能ではゲーム中に表示される運動量などを認識する．コンボ数表示機能では，正しい姿勢で運動が行えた際にコンボ数をゲーム画面内に表示する．ピンチ応援機能では，ピンチ時に指定した音声を再生する．本システムには CPU が搭載された FPGA ボードを使用しているが，リアルタイム処理を求められないが複雑な処理が必要な文字認識は CPU で行っており，エフェクト検出やコンボ数表示の映像効果付与は FPGA で行っている．

ピクセル単位の遅延評価を行ったところ，本システムでは元のゲーム機の映像出力から 13 ピクセル分しか遅延していないことが分かった．同様のシステムは USB カメラやキャプチャユニットなどを PC に接続しても実現出来るが，数フレーム分 (少なくとも数百万ピクセル分) の遅延が発生していた．遅延が大きいとゲーム自体が成り立たないため低遅延であることは重要である．

このように映像を出力する機器に対する機能拡張は，コンシューマーゲーム以外でも有用であると考えられる．仮に FPGA 付テレビが将来普及した際には，既存の映像コンテンツを後からプラグインのような形で修正することで新たな楽しみを提供出来る可能性がある．

目 次

1　はじめに

1.1　研究背景

コロナウイルス感染症の影響から手軽に行えるフィットネスの需要が高まっている [1]. 本研究で着目した任天堂リングフィットアドベンチャーは任天堂 Switch[2] 向けのフィットネスゲームである [3]．リングフィットアドベンチャーでは任天堂 Switch の 3D センサコントローラと専用のリングを組み合わせることで，腕，お腹，足など体全体のトレーニングを可能にしている．しかし，他のフィットネスゲームと比較すると．まだまだ運動効果を高められる可能性がある．任天堂 Switch はコンシューマーゲーム機であることもあり，機能の拡張が難しい．

1.2　目的

そこで本研究では，映像信号を介した機能拡張を行うことで，フィットネスゲーム向けの運動モチベーション向上システムを提案する．既存の映像コンテンツ（ゲーム機やテレビなど）をリアルタイムに修正するためには，高速に映像をキャプチャして映像信号を変化させなければならない．そのため，本研究では FPGA(Field Programmable Gate Array) を用いたシステムを作成する．また同様のシステムを実現する他の技術との比較評価を行い，本システムの利点について調査する．

1.3　本論文の構成

1. はじめに
   本研究の背景，目的について述べる．
2. フィットネスゲームとリングフィットアドベンチャー
   一般的なフィットネスゲームとリングフィットアドベンチャーの比較について述べる..
3. ゲームとアドオン
   電子ゲームの構成と拡張機能を追加するアドオンについて述べる．
4. 既存研究・技術
   リングフィットアドベンチャーなどのゲームに対する既存技術や映像信号を用いた機能拡張に関した研究ついて述べる．
5. システム構成
   本研究で作成するシステムについて述べる．本システムで作成した映像効果付与機能，文字認識機能，ピンチ時応援機能について述べる．

6. 動作と評価
   本システムの動作と評価について述べる．評価で使用したデバイス構成についても述べる．

7. おわりに
   本システムのさらなる応用例について提案し，まとめを述べる．

2　フィットネスゲームとリングフィットアドベンチャー

2.1　一般的なフィットネスゲーム

2.1.1　FiNC HOME FiT

FiNC HOME FiT[4] は任天堂 Switch のフィットネスゲームである．FiNC HOME FiT はカンフーからボクシングなどの様々な格闘技の動きを使って，音楽に合わせてリズムよく動くリズムフィットネスゲームである (図 1)．図 1 では FiNC HOME FiT での運動の様子を示しており，下部ではユーザーの動くタイミングを示していて，右下では連続して正しく動いた回数をコンボ数として表示している．また，FiNC HOME FiT では日々の運動データを時系列のグラフにして表示できる (図 2)．時系列データにすることで，ユーザーは毎日の運動の変化を視覚的に確認することが出来る．この機能はユーザーの継続的な運動を促している．

図 1: FiNC HOME FiT（参考文献 [5] より引用）

図 2: FiNC HOME FiT での運動データの可視化（参考文献 [4] より引用）

2.1.2 Fit Boxing

Fit Boxing[6] も任天堂 Switch 向けのフィットネスゲームである．Fit Boxing はボクシングの動きを使ったリズムゲームである．Fit Boxing は FiNC HOME FiT と異なり，上半身を中心に運動することができる．Fit Boxing では複数人での対戦や協力プレイをサポートしている (図 3)．複数人での運動は運動モチベーションを高めることも分かっているため [7]，複数人プレイでは運動効果の向上が期待できる．また，Fit Boxing の後続ソフト Fit Boxing 2[8] では専用のスマートフォンアプリ [9] と連動することも可能になった．ユーザーは専用のアプリを使用することで，Fit Boxing 2 での運動データをスマートフォンからアクセスすることが可能になる (図 4)．

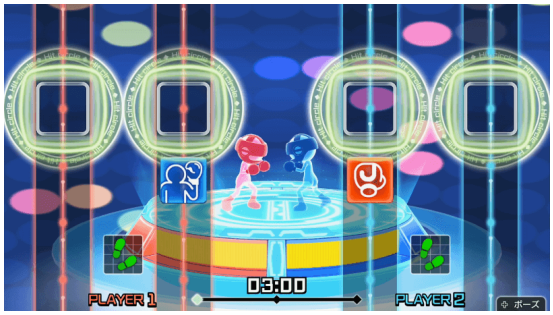


図 3: Fit Boxing での協力プレイ（参考文献 [6] より引用）

図 4: Fit Boxing 公式アプリと運動データ（参考文献 [9] より引用）

2.2 リングフィットアドベンチャーと課題

リングフィットアドベンチャーは任天堂 Switch の JoyCon と呼ばれる 3D センサコントローラと専用のリングを組み合わせたフィットネスゲームである．リングフィットアドベンチャーでは腕，お腹，足など体全体のトレーニングを可能にしている．FiNC HOME FiT や FitBoxing では JoyCon の 3d センサから運動を判定していたのに対し，リングフィットアドベンチャーは専用のリングを使っている．そのため，リングを押し込む動作や引っ張る動作もトレーニングに使うことが可能になり (図 5)，より豊富な運動種目を行える．

運動では単に運動量を増やすよりも正しい姿勢で運動を行い，継続して運動をこなすことで運動効果が向上する．しかし，リングフィットアドベンチャーは FiNC HOME FiT や FitBoxing などの他のフィットネスと比べて以下の機能がない．

(1). 運動時にコンボ数を表示する機能
　　連続して正しく運動を行うことを促進する．

(2). 時系列の運動データを表示する機能
　　日々の運動データの変化を確認することで，継続的な運動を促す．

(3). 複数人でゲームをプレイする機能
　　運動のモチベーションを向上させる．

リングフィットアドベンチャーは複数人の協力・対戦プレイをサポートしていない．複数人の対戦プレイに近いものに，運動データと使って他ユーザーとランキング形式で比較するものがある．リングフィットアドベンチャーでも図 6 のように運動の累計データを使うものがあるが，一日単位のデータを使ったランキング機能はなくランキング機能自体の自由度は高くない．

そのため，仮にリングフィットアドベンチャーに (1)(2)(3) の機能を追加できれば，ユーザーの運動効果の向上が期待できる．以上からリングフィットアドベンチャーは他のフィットネスゲームと比較して独自のゲーム性で様々な種類の運動を行える強みがあるものの，まだまだ運動効果を向上させられる可能性がある．

図 5: リングフィットアドベンチャー（参考文献 [10] より引用）

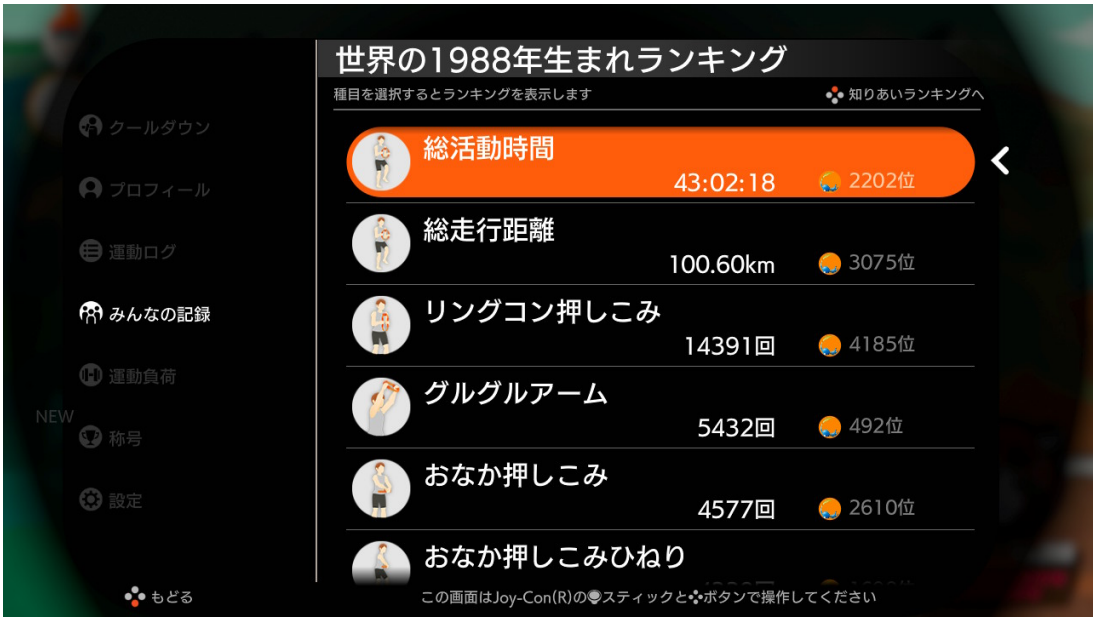


図 6: リングフィットアドベンチャーでのランキング表示

3　ゲームとアドオン

3.1　電子ゲームの構成

電子ゲームにはPCゲームやスマートフォンゲーム，モニターにつなげて遊ぶコンシューマーゲームや，モバイル型コンシューマゲームなど様々な種類のものがある．ここで電子ゲームの構成を考えると，電子ゲームは図7のように大きく入力装置，内部装置，出力装置の3つの装置に分類することが可能である．内部装置とはゲームのプログラムやそのプログラムを実行するOS，ハードウェアである．入力装置はユーザーが何らかの信号を送りゲームの操作を意味する信号を送るための装置である．出力装置はゲーム内の状態をユーザーに知らせるための装置である．例えば，PCゲームの場合，キーボードやマウスが入力装置であり，PC本体やOS，ゲーム自体のプログラムが内部装置であり，映像を映すモニターなどの液晶パネルと，音を伝えるスピーカーが出力装置であると考えられる．スマートフォンゲームの場合はタッチパネルが入力装置であり，スマートフォン内部の基板や，OS，ゲームプログラムが内部装置であり，液晶パネルが出力装置である．

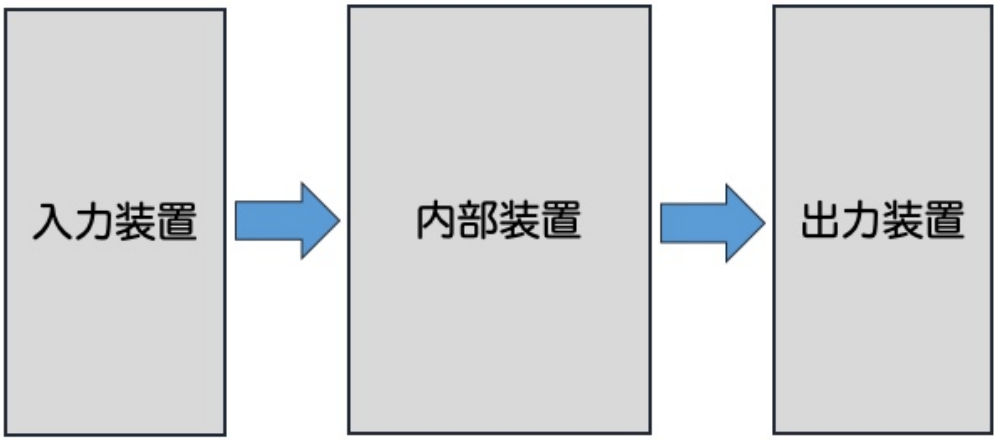


図 7: 電子ゲームの構成図

3.2　アドオンの分類

3.2.1　内部装置を拡張

ゲームの機能追加には様々な方法が考えられる．内部装置に対し機能追加を行う場合，ゲーム内部のハードウェアやゲームプログラムに介入する手法がある．例えば MOD と呼ばれるゲームプログラムに適応する追加のプログラムを実行することで機能を追加するという方法がある．図 8 では仮想空間でものづくりなどを行えるゲームのマインクラフトに対して，ゲーム内空間でのユーザーの位置情報を画面右上に表示する MOD を示している．MOD にはゲームを便利にする機能を拡張するものや，グラフィックを綺麗にするものなどがある．MOD はゲームと同時に外部のプログラムを実行できる PC ゲームなどで多くみられる．しかし，リングフィットアドベンチャーの動く任天堂 Switch などのコンシューマーゲームでは外部プログラムを実行することや，内部ハードウェアに触れることが難しく，これらの手法はコンシューマーゲーム向きではない．

図 8: マインクラフトマップ機能追加 MOD（参考文献 [11] より引用）

3.2.2　入力部を拡張

入力装置を拡張した機能追加を行った例として，連射用コントローラーがある．ユーザーは連射用コントローラに配置された連射用ボタンを押すだけで，内部装置からはユーザーが連射を行ったのと同様の信号を送ることができる．連射用コントローラは FPS などのゲームで使用したり，ゲームの操作を自動化したりすることが可能になる．連射用コントローラはコンシューマーゲームやスマートフォンに対応したものもある．図 9 ではスマートフォン用の連射用ツールを示している．これを使うユーザーは図右下のコントロールボックスに連射速度を入力し，左上のハサミでタッチパネルを挟むことで好きな速度で入力を自動化することができる．連射コントローラは任天堂 Switch にも対応したものもあり，任天堂 Switch のドックの USB コネクタに接続することで連射機能を使用することができる．しかし，リングフィットアドベンチャーのような任天堂 Switch の JoyCon を使っているゲームソフトでは，JoyCon と Switch 本体の通信方式が Bluetooth であることもあり機能拡張をすることが難しい．

図 9: スマートフォン用連射ツール（参考文献 [12] より引用）

3.2.3 出力部を拡張

出力装置を拡張した機能拡張としてはディスプレイに映すゲーム映像に介入する方法が考えられる．ゲーム映像からゲームデータを取得したり，ゲームの内部状態を取得することでゲームを便利にしたり，ゲームの映像信号を上書きすることでゲーム自体を楽しくするような映像処理を行うことが可能になる．任天堂 Switch はドックの HDMI 外部出力コネクタから映像信号を取得することができる．ゲームへの様々な機能追加の手法の中でも，映像信号を介した機能拡張を行うことで，リングフィットアドベンチャーの運動効果を高めることができると考えた．

4　既存研究・技術

4.1　RingFitRanker

リングフィットアドベンチャーに関する既存の取り組みとして，RingFitRanker[13] と呼ばれるものがある．リングフィットランカーは Twitter で稼働するボットである．リングフィットランカーを使うユーザーは，リングフィットアドベンチャー終了時に表示される運動結果画面をスクリーンショット保存し，アルバムアプリから「Twitter へアップロードボタン」を押して Twitter の RingFitRanker アカウント (@ringfitranker) へ画像を含むツイートを送る．RingFitRanker は運動結果画面のカロリーデータを抽出し，毎日 1 回消費カロリーランキング結果の画像 (図 10) をツイートする．こうすることで，RingFitRanker は複数人でカロリーデータを使って競い合うことが可能にした．また，2021 年 9 月にはユーザーが過去の自分のカロリーデータと比較してどの程度運動したか確認できる個人ランキング機能も追加された．リングフィットランカーはカロリーを使った他人との対戦を可能にし，運動モチベーションの向上を実現している．

しかし，リングフィットランカーではカロリー以外のデータを活用していないという課題があった．また，リングフィットランカーは Twitter へアップロードする手間がかかってしまう．

本研究ではカロリー以外の運動種目ごとのデータの記録を取得することや，自動で運動結果データの抽出を行うことを目指す．

図 10: RingFitRanker のランキング画像（参考文献 [13] より引用）

4.2 Ikalog

Ikalog[14] は任天堂 Splatoon 向けのリアルタイム映像読み取りシステムである (図 11). Ikalog では HDMI を介してゲームの勝敗やゲーム内状況を記録している．Ikalog のシステムでは専用 OCR を作成してゲーム内文字を収集することや，ニューラルネットワークを用いた画像認識を使用することでシステムを実現している．Ikalog は HDMI キャプチャデバイスを WiiU に接続し，Ikalog 実行用 PC を使用することで，ゲームをしながらスクリーンショットなどの手間をかけずに自動的にゲーム内データを取得することができる．

本研究では取得したゲーム映像に対しコンボ数などの映像効果付与を行う点で Ikalog とは異なっている．

図 11: Ikalog の取得データ（参考文献 [15] より引用）

4.3 HDMI を介した機能拡張に関する研究

4.3.1 FPGA で手話映像の拡大を行った研究

Auangkun らはテレビのニュースなどで手話映像の表示されているワイプのサイズをリアルタイムに拡大する専用システムを FPGA を使って開発した (図 12[16]). 元の放映映像では手話者が小さくて見にくくても，この機器を通すことで大きく表示され見やすくなる．システムでは Zedboard[17] に HDMI 入力用の FMC カードを追加して映像の入出力を行っている．

本研究では映像内からの運動結果データの取得や，検出したゲームエフェクトに応じた映像処理を行うため既存研究とは異なる．

(a) 適用前　　**(b)** 適用後

図 12: 手話映像拡大システム（参考文献 [16] より引用）

4.3.2　PYNQ-Z1 で映像フィルタリングを行った研究

FPGA を用いて HDMI 信号を入出力することで映像の修正を加えた他の例として，PYNQ-Z1 ボード (図 13[18]) を使用して様々な映像フィルタをその場で切り替えられるシステム [19] がある．PYNQ-Z1 ボードは HDMI 入出力コネクタが最初から搭載されていることや Python 環境が用意されていることなどから高速な映像処理を行う用途によく使用されている [20][21][22].

本研究ではリアルタイム処理が必要なエフェクト検知や映像効果付与を FPGA で行い，リアルタイム処理は必要ないが複雑な処理の必要な文字認識を CPU で分担して行う点で既存研究と異なっている．

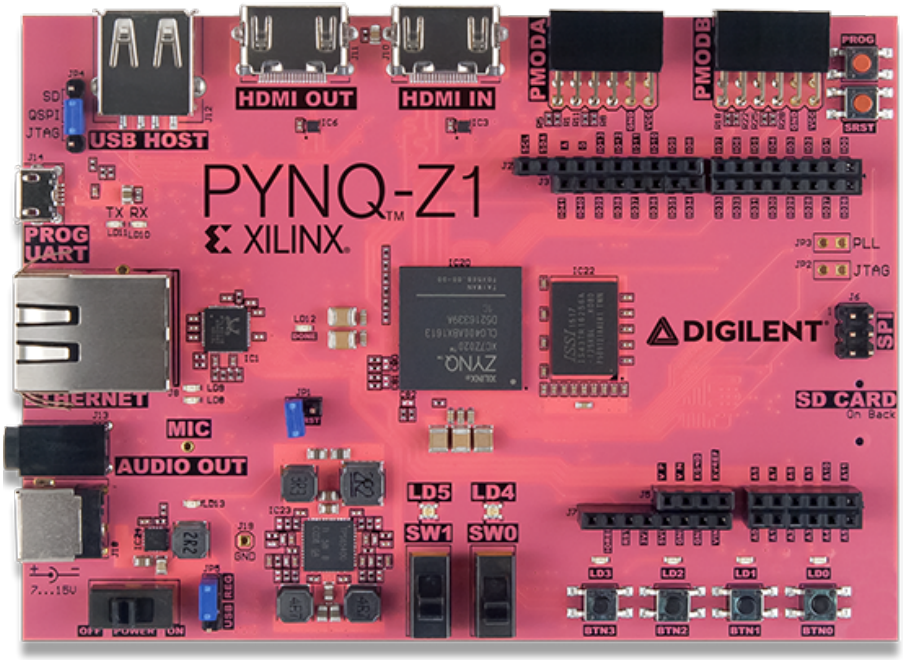


図 13: PYNQ-Z1 ボード（参考文献 [18] より引用）

5　システム構成

5.1　システム設計の方針

本システムでは，リングフィットアドベンチャーのゲームに以下の三つの機能を追加する．

(1). 適切な運動が出来た場合に画面に映像効果を出す

(2). 個別の運動の種類毎の成果を記録したり競い合うための文字認識

(3). ゲーム内でピンチのときに応援音声を出力する

まず (1) のために，毎回の運動が正しく行えたかどうかによって違う Best, Good, 白文字というエフェクト表示を認識する．そして連続して正しい運動が出来た際に映像効果を画面上に表示することで，ユーザーに正しい運動を行うモチベーションを高める．例えば図 14 のように同じスクワット運動だとしてもどの程度屈むかによって運動量は大きく異なり，Best と表示されるスクワットを繰り返すことで運動の効果が期待される．このため，Best を多く繰り返した際にコンボ数を示す映像効果を付加する．元々のゲームではこの三つの差をあまり重視していないため Best を繰り返すモチベーションが高くなりにくい．

次に，(2) のために，図 15 のような運動の種類ごとの回数を文字認識する．文字認識を行うことができれば，結果を記録したり他のユーザーと競い合うことが可能になる．元々のゲームでも累計データで競い合う機能や，簡単にスクリーンショットを Twitter に投稿出来る機能があるが，細かい運動毎には区別していない．投稿された Twitter 画像から自動的にランキングを作るシステム [13] では，ユーザーがツイートするという手間を間に挟む必要がある．本システムでは自動的にそのようなランキングを行えるものを目指している．ただし本研究では文字を FPGA で認識するところまでを研究の範囲としておりその後の処理までは開発しない．

また，(3) のために，ゲーム内で相手からの攻撃を受け体力ゲージが減ったときに表示される図 16 のような画面の外側に赤みがかかるエフェクトを検知してユーザーを応援する音声を出力する．例えば好きなアイドルや声優などの音声をあらかじめ応援音声として登録してゲームのピンチ時に出力することによってゲーム自体にさらなる楽しさを追加できる．

このような機能を追加する際には，以下のような制限を前提としている．

(1). 任天堂 Switch には何も手を加えない

図 14: 運動中の姿勢とゲーム内エフェクト

(2). 描画の遅れが 1 フレームの時間よりも十分に短い

(1) のためには，任天堂 Switch からの HDMI 映像出力だけからゲームの状態を認識する必要がある．また，映像効果はユーザーが見ている画面上に現れるべきであるため，映像信号を途中で改変する機能が必要となる．(2) のためには，一旦一フレーム分の映像信号をキャプチャしてから処理した後に映像出力したのでは間に合わない．Youtube のゲーム実況などではパススルー付きキャプチャユニットを用いることが多いが，これは実況者が遅延の無い画面を見ないとゲームをスムーズに行えないためである．今回の場合システムによる映像効果がゲーム画面内に表示される必要があるため，パススルー機能では実現出来ない（Youtube 視聴者側の画面だけに表示出来ればよいのであれば一フレーム程度の遅延は問題とならない）．このため，FPGA 等を用いた専用システムが必要となる．

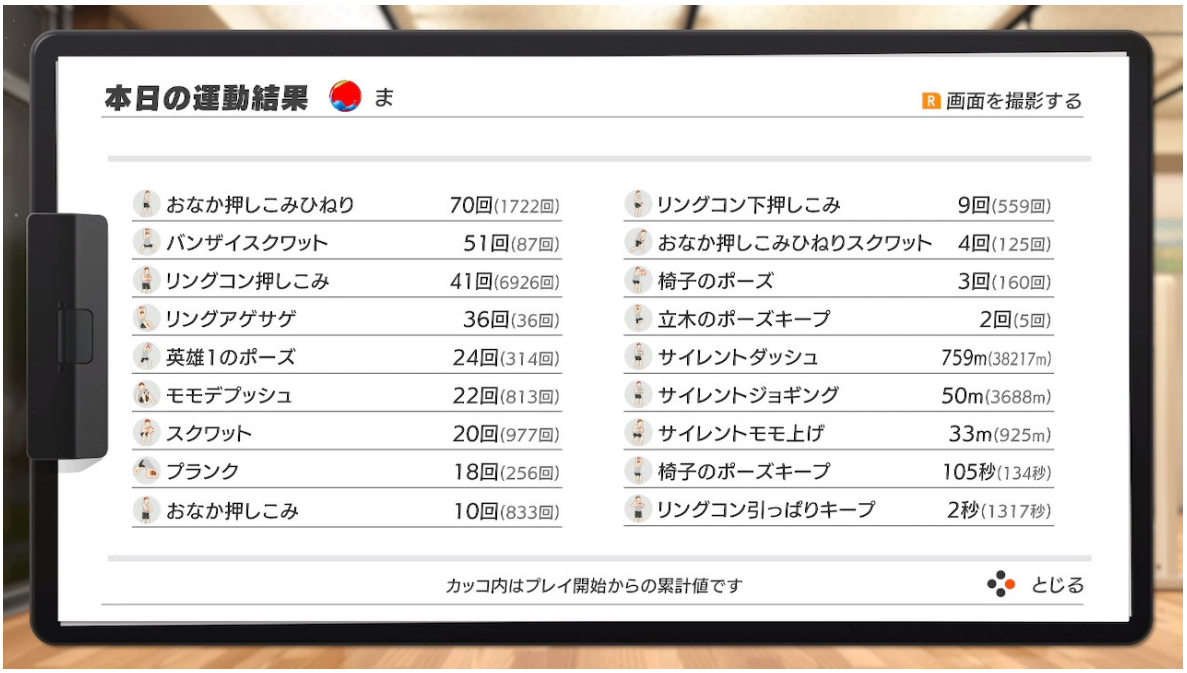


図 15: 運動結果画像

図 16: ゲーム内のピンチ時の様子

表 1: 使用した機材

機材の種類	型番・バージョン
FPGA	PYNQ-Z1
Vivado	2019.2
任天堂 Switch	HAC-001
OS	PYNQ 2.6.0

5.2 使用した機材

表 1 は本研究で使用した機材やソフトウェアを示す．FPGA ボードとして使用した PYNQ-Z1 は，FPGA として XC7Z020，HDMI 入出力コネクタをそれぞれ 1 個ずつ，512 Mbyte の DDR メモリを搭載している．XC7Z020 内には，PS(Processing System) と呼ばれる ARM Coretex-A9 CPU と PL(Programmable Logic) と呼ばれる FPGA が内蔵されている．ARM CPU で Linux を動かすことで高度なソフトウェア環境を使いつつ処理の重い部分は PL 部の専用回路による処理を組み合わせることが出来る．特に PYNQ シリーズのボードでは Python による Jupyter Notebook 環境を最初から提供しており，搭載されている HDMI 入出力回路を用いた機能を Python から手軽に使用することが出来る．ただしユーザーの専用回路と組み合わせる際には PL 部の再構築が必要である．

5.3 コンボ数表示機能

5.3.1 全体構成

システムの構成図を図 17 に示す．任天堂 Switch の HDMI 出力からの映像信号が HDMI Decoder へと入力される．入力信号は三つに分かれるが，一つはキャプチャ用ハードウェアを介し VRAM(DDR メモリ) へと書き込まれる．もう一つはピンチ検出モジュールに直接渡され，最後の一つは Best Good 検出モジュールに渡される．Best Good 検出モジュールではゲーム中のエフェクトを検出し，次に映像処理モジュールで検出したエフェクト数に応じて元の映像信号に特別な映像を付加する．最後に HDMI 信号へと変換して出力する．HDMI 入出力には GitHub で公開されている hamsternz らが作成した HDMI デコーダ，エンコーダハードウェアを使用した [23].

5.3.2 Best Good 検出

ゲーム中に表示されるエフェクト (図 14) を検知する手順は以下の通りである．

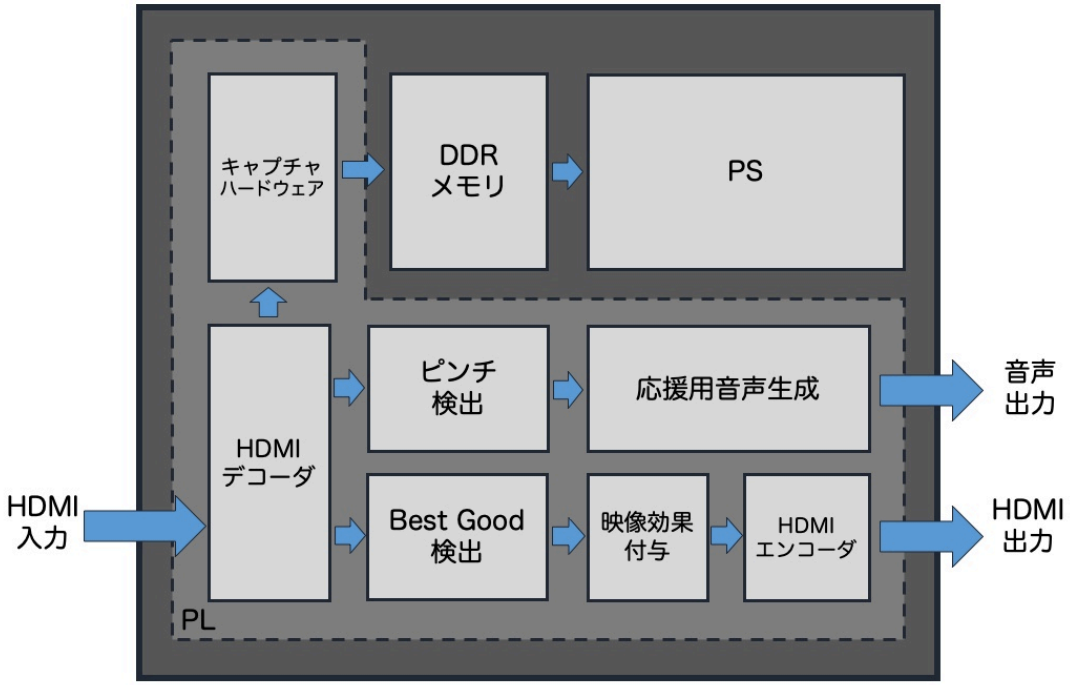


図 17: ハードウェアブロック図

(1). エフェクト毎の色範囲に入っているピクセルを探す

(2). 対象とする水平・垂直のピクセル座標の範囲を絞る

(3). 上記ピクセルの検出領域の大きさを求める

(1) では色の範囲を指定することで Best や Good の文字の候補ピクセルを探す．例えば，Best は赤や黄色が変化しながら表示され，Good は単色の黄緑色で構成されている．閾値の範囲を満たす色を白，それ以外を黒というように色で区別した画像を作ると，Best の場合は図 18 のような画像が得られる．Best，Good などの運動姿勢に対するエフェクトは表 2 の RGB 値で抽出できることを確認した．

(2) では全てのピクセルを検出対象とするのではなくてある程度範囲を絞って次のステップに行く．3 種類のエフェクトが表示されるピクセル位置は大きく変化しないため，水平座標範囲，垂直座標範囲が当該エフェクトの表示される範囲であるかどうかを判定する．

(3) では，それまでに検出されたピクセルの領域の大きさを求めることで本当に Best や Good が表示されたことを認識する．エフェクトはアニメーション形式で表示されるため，アニメーションの中でエフェクトが最も大きく表示されるときの垂直方向，水平方向の

図 18: カラーフィルタリングの適用前 (上) と適用後 (下)

表 2: エフェクトとカラーフィルタリングの閾値

エフェクト種類	Red	Green	Blue	その他
Best	210〜235	120〜215	0〜30	-
Good	190〜230	216〜237	0〜30	Red < Green
白色数字	240〜255	240〜255	240〜255	-

最大ピクセル長を予め求めることが出来る．その大きさを超えた場合にエフェクトを検知したと判断する．

本システムでは 1 フレームよりも十分短い時間で反映することを要件としているため，VRAM に格納された情報を用いてエフェクトを検出している訳ではない．まず，HDMI からデコードされた映像信号は図 19 の走査線と呼ばれる矢印のように水平方向に対し連続したピクセルの画素値を保持している．そのため，カラーフィルタリングされた映像信号から水平方向の最大ピクセル長を計算するには，フィルタリングされた映像信号の中で RGB 閾値範囲内の連続したピクセル数を記録すれば良い．一方，垂直方向の最大長を調べるには水平座標毎に垂直方向の RGB 閾値範囲内の連続ピクセル数をカウントする必要がある．また，本システムでは全ての水平座標毎の垂直方向の最大連続ピクセル数を分散 RAM に記録して，エフェクト検知を行っている．

映像効果付与のために作成した回路構成を図 20 に示している．図 20 では，in_blank 信号から水平方向・垂直方向の座標 HCnt，VCnt を求めている．colorfilter は現時点でモジュールに代入されている RGB 信号が検出対象のエフェクトの RGB 閾値範囲かどうか調べる組み合わせ回路である．HCnt，VCnt と colorfilter の結果を使用して水平方向の最大長 Hmax と垂直方向 Vmax を計算している．HCnt と VCnt がどの値であっても，Hmax と Vmax があらかじめ決めた大きさを超えると，DetectEffect に 1 が代入され，エフェクトを認識したことになる．そのため，1 フレーム全体を読み込まなくても，HDMI デコードされた RGB 値を 1 つずつ読み込みながらエフェクト検知を行える．この回路の検知で発生する最大の遅延は reg 変数にピクセルを保存したことで発生する数ピクセル程度のものである (全体の遅延については 6.3.2 を参照のこと).

5.3.3 映像効果付与

エフェクト検知モジュールから伝えられる Best や Good の検出回数が一定回数を超えると，映像効果付与モジュールでは元のゲーム画面に特別な映像を付与する．現時点で

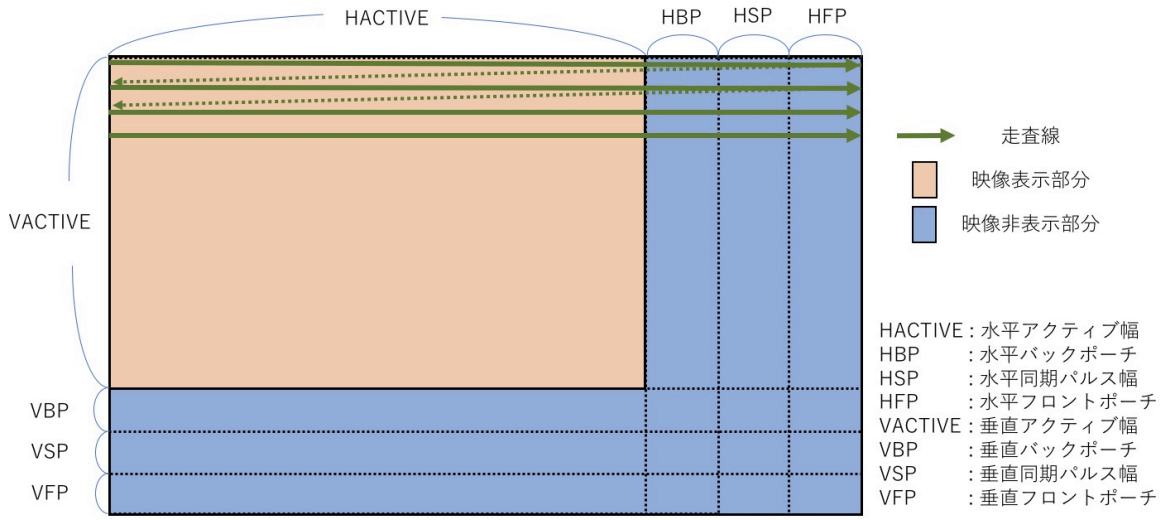


図 19: 映像信号におけるデータの流れ

は二つの映像効果を実装している．

(1). 検出された Best の回数の表示

(2). コンボ数を示す流れるメッセージの表示

図 21 は流れるコンボ数のメッセージの表示方法を表したものである．流れるメッセージに関しては，予め「5x Best」,「10x Best」などの文字をドット絵で表したデータとして用意し，画面右から左まで移動するコンボ数表示エリアに映すことで実現している．コンボ数を表示する処理において，ゲーム映像を表示している部分とコンボ数表示エリアが重なっているところだけ RGB 値の上書きを行い，他のエリアに対しては HDMI デコーダから入力された映像信号をそのまま出力することで，1 ピクセルも遅延を発生することがない．

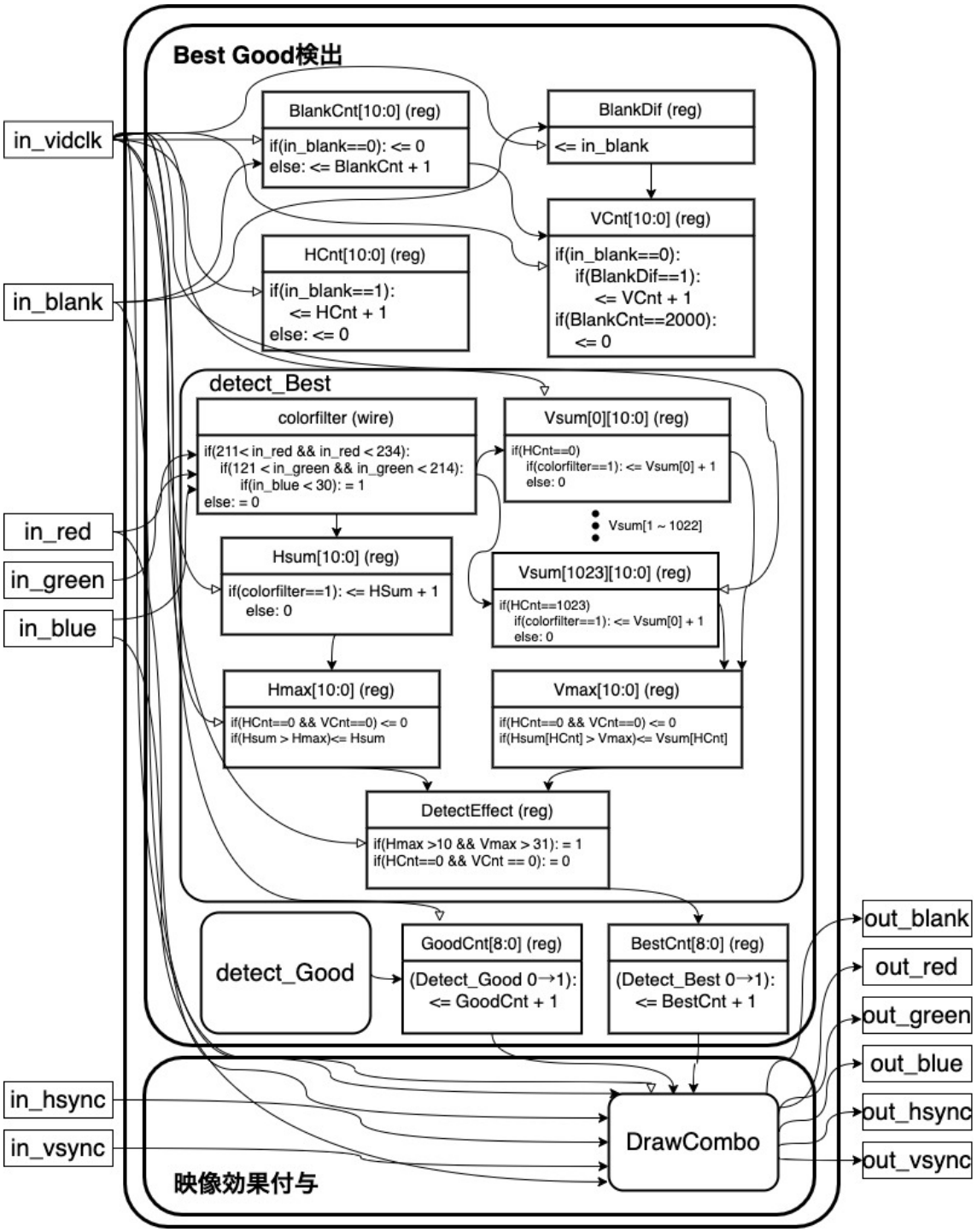


図 20: コンボ数表示のための回路構成

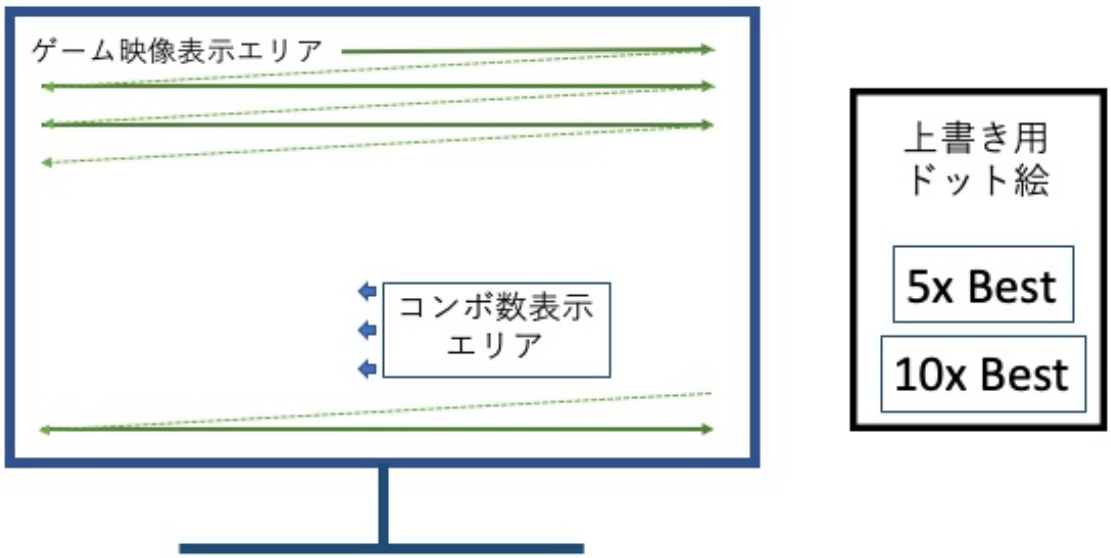


図 21: コンボ数付与のオーバーレイ

5.4 文字認識機能

5.4.1 全体構成

個別の運動記録の成果を他人と共有するために，運動結果画面からの文字認識を行う．ゲーム内でこの画面 (図 15) はしばらく表示しておくことが出来るため，リアルタイムに文字認識する必要はない．しかし認識しなければならない文字数が多いため，エフェクト検知モジュールのような色や大きさだけで検知するのは難しい．そのため，運動結果の文字認識には PYNQ-Z1 ボード上の PS を使った処理を行う．PS は PL よりも速度が遅いため，HDMI 信号の生のタイミングでピクセル情報を処理することは出来ない．このため VRAM に保存された情報にアクセスして文字認識を行う．

5.4.2 キャプチャ回路

図 22 ではキャプチャを実現するための回路構成を示す．HDMI デコーダから出力している映像信号の矢印で示されているように，取り込まれたゲーム画面は PYNQ-Z1 の PS からアクセスするため VRAM に書き込まれる．HDMI デコーダから出力された映像データは VDMA IP コアに対して，AXI-Stream と呼ばれるデータ形式で入力すると DDR メモリに書き込まれる．VDMA IP コアを使用するためには AXI-Stream 形式のデータにする

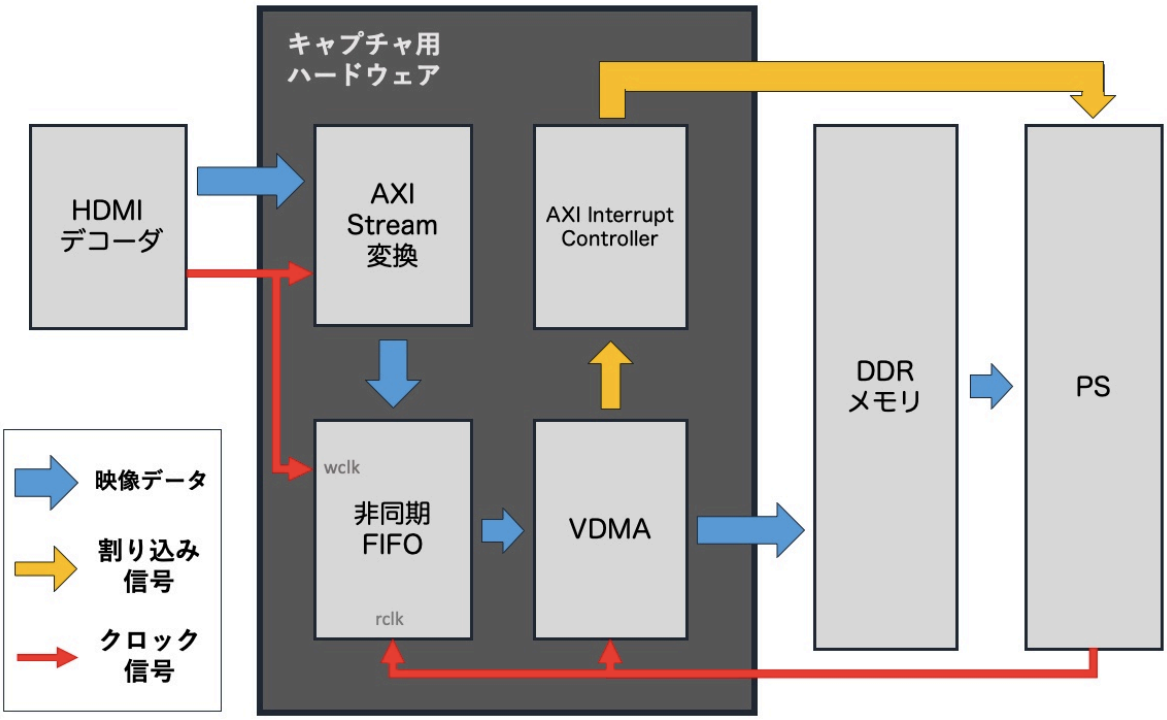


図 22: 文字認識機能のための構成

必要がある．HDMI デコーダから出力された映像データは HDMI デコーダから出力されるピクセルクロックを基準としているが，AXI-Stream では PS から出力されるクロックを基準とする必要がある．これらの周波数の違いを吸収するため，Xilinx の FIFO Generator IP コアを非同期クロックモードにして使用した．非同期 FIFO は BlockRAM を用いて生成するよう指定し，オーバーフローを防ぐため可能な限り多くの BlockRAM を使用した．また，VDMA IP コアから出力された割り込み信号線を AXI Interrupt Controller IP コアを介して，PS に接続する必要があった．

PS から VRAM へのアクセスには，OS から提供された既存の Python のライブラリを使用した．標準で搭載されていたものは古かったため，表 1 の PYNQ OS を使用した．

5.4.3 文字認識

PS では HDMI 映像のキャプチャデータに対し，文字認識を行う．文字認識には PS で走る Linux 上の Python の環境を使用した．既存のシンプルな文字認識エンジン [24] では，リングフィットアドベンチャー内で使用されるフォントを読み取れないことが分かったため，本システムでは独自の文字認識エンジンを実装した．実装には簡単のため K 近傍法

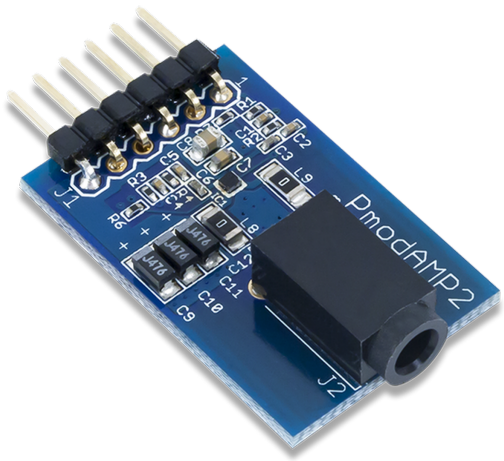


図 23: 音声出力に使用したアンプモジュール（参考文献 [25] より引用）

を用いたが，今後はより高速に認識できるアルゴリズムを実装する予定である．

5.5 ピンチ時応援機能

5.5.1 全体構成

ピンチ検出モジュールでは，ゲーム内でピンチ時に表示される赤っぽさを検知して，あらかじめ録音しておいた「頑張って！」という応援音声を流す．音声の出力には，図 23 のアンプモジュールを PYNQ-Z1 ボードの外部に接続して使用した．外部ピンを使うことや，将来的に HDMI の映像信号に音声を上書きすることでディスプレイから音声を流すことも可能になるため，これらのモジュールは PYNQ-Z1 ボードの PL で実装する．

5.5.2 ピンチ検出

ゲーム中のピンチ状況 (図 16) を検知する手順は以下の通りである．

(1). ピクセルの色空間を変換する

(2). 画面上部 4 ライン全体の赤っぽさの平均値を計算する

(1) では，赤みを抽出するために，YCbCr と呼ばれる色空間を使用した．YCbCr は輝度を表す Y，輝度と青色成分の差を表す Cb，輝度と赤色成分の差を表す Cr で表現される色空間である．ここでは RGB 空間の映像のピクセルから 1 対 1 に変換できる YCbCr444 方式に着目した．RGB 空間と YCbCr444 空間は次の式で対応している．

$$Y = 0.257R + 0.504G + 0.098B + 16$$
$$Cb = -0.148R - 0.291G + 0.439B + 128$$
$$Cr = 0.439R - 0.368G - 0.071B + 128$$

赤っぽさの検出だけを行えばよいため，Cr だけを計算する．ここで，図 24 では PC を使ってピンチ中のゲーム画面を Cr 値でフィルタリングしたものを表している．図 24 ではすべてのピクセルに対して，閾値値を超えるときに白を，閾値を超えないときは黒を出力している．図 24 において Cr>160 でフィルタリングを行った画像に注目すると，画面上部の一部分でピンチの赤みが抽出できていることが確認できる．

次に，FPGA 上のハードウェアで Cr の計算を実装した．Cr の計算には整数部 8 ビット，小数部 8 ビットの固定小数点を使用した．図 25，26 では図 16 のスクリーンショット画像に対してそれぞれ PC と FPGA で Cb>160 のフィルタリングを行ったものを示している (図 26 の画面左下にはデバッグ用の 2 進数が赤色で表示されている)．FPGA でも Cr が正しく計算できていることが分かる．

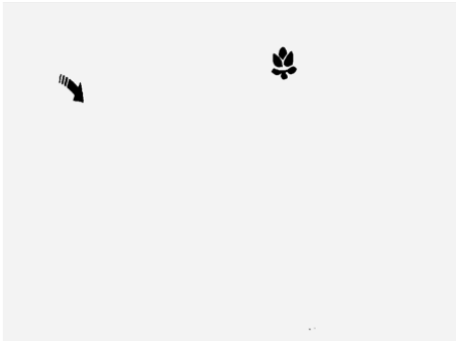
(2) では，映像信号のうち画面上部 4 ラインの Cr 値の平均値の計算を行った．画面上部 4 ラインの Cr 値の平均値が 160 を超えたときにピンチと判定するようにしている．赤みの検出では画面全体のデータを一度に保持しなくても，走査線の最初から 4 ライン目のデータを受け取った直後にピンチの検知を行える．

図 27 ではピンチ時応援機能全体の回路構成を表した図である．Cr_16bit と Cr は先に述べた整数 8bit，小数 8bit の固定小数点を使った Cr 値の計算を表している．ピクセル座標の HCnt，VCnt を使って画面上部 4 ライン分の映像がデコードされている間は Cr が Cr_sum に加算される．ゲームの解像度は 1024x768 であり，上部 4 ライン全ピクセルの Cr 値の平均値を求めるには，Cr_sum を (水平ピクセル数 $*$ 対象ライン数) = $(1024 * 4) = (2^{11})$ で割ればよい．そこで，Cr_mean では Cr を 11bit 右にシフトすることで平均 Cr 値を計算している．平均 Cr 値が 160 を超えたときに detect_weak を 1 にすることでゲーム中のピンチを検知している．

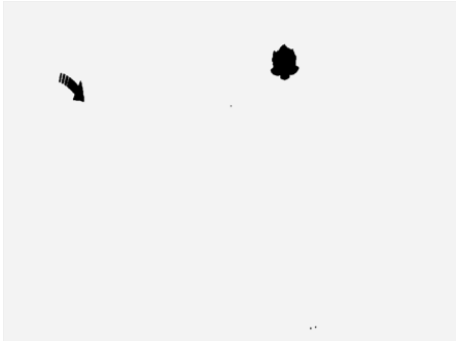
5.5.3 応援用音声出力

ピンチエフェクトの検出後，音声出力モジュールではユーザーを応援する音声を出力する．ここで出力する音声はあらかじめ録音しておいた「頑張って」という音声を 8bit, 8kHz でサンプリングしたものをパルス幅変調 (PWM) し，外部ピンに接続したアンプモジュールから出力している．

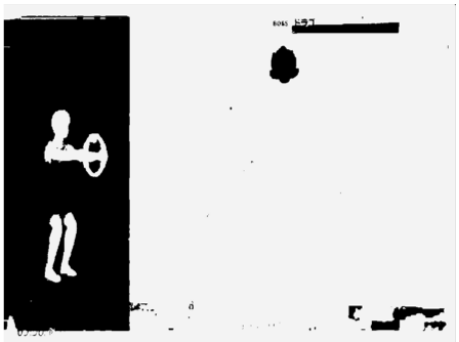
(a) 処理前画像



(b) Cr>110 でフィルタリング



(c) Cr>120 でフィルタリング



(d) Cr>130 でフィルタリング

(e) Cr>140 でフィルタリング

(f) Cr>150 でフィルタリング

(g) Cr>160 でフィルタリング

(h) Cr>170 でフィルタリング

図 24: Cr 値フィルタリングの閾値を変化させたときの様子

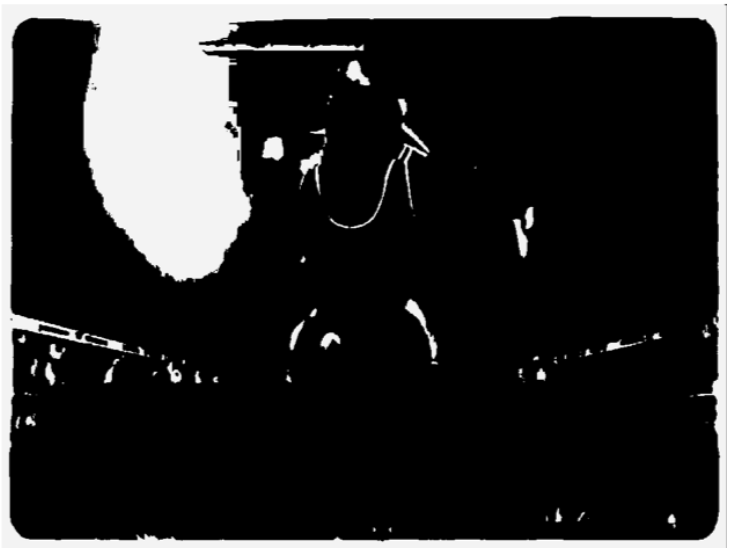


図 25: PC でフィルタリング (Cr>160)

図 26: FPGA でフィルタリング (Cr>160)

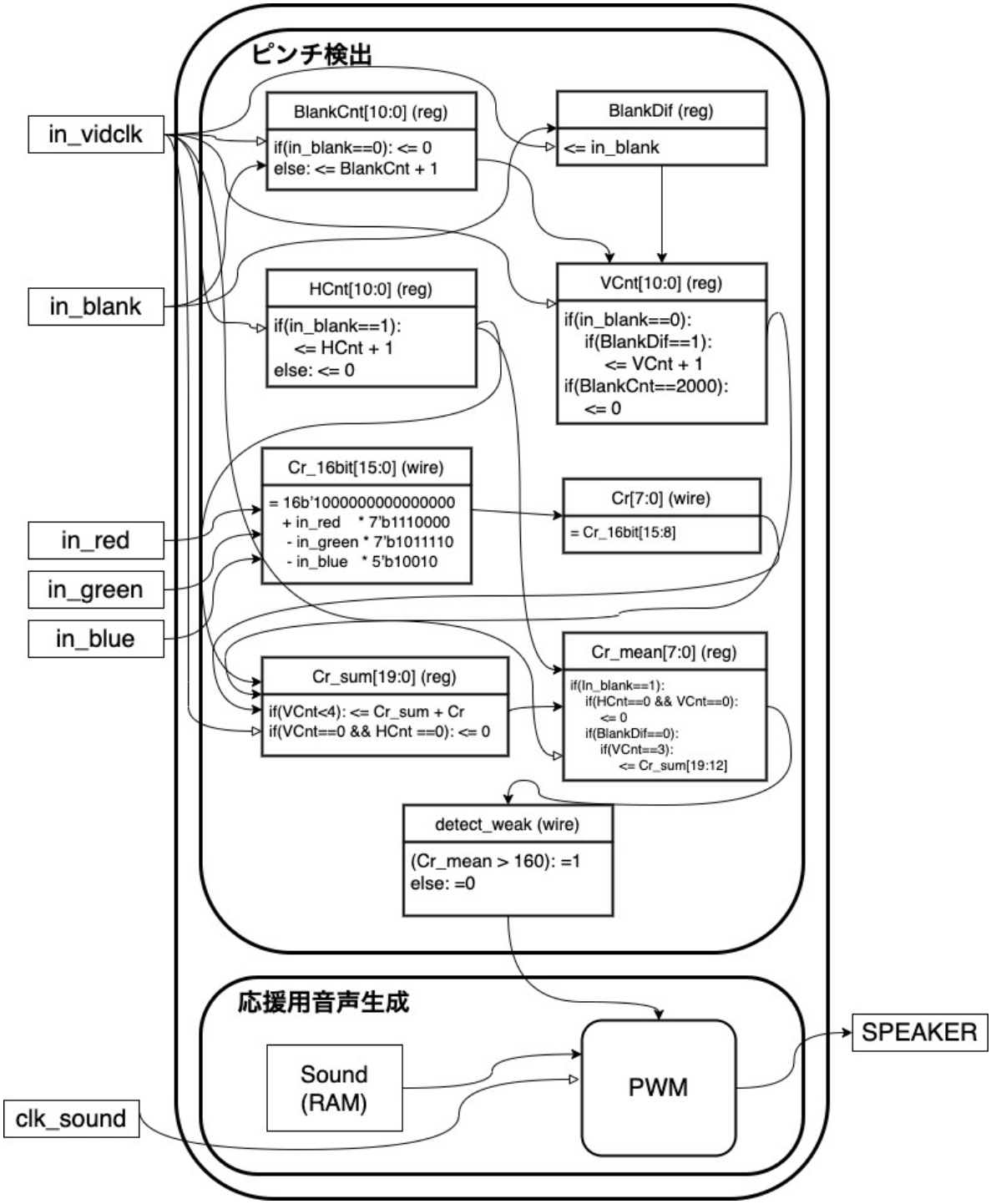


図 27: ピンチ時応援機能のための回路構成

6　動作と評価

6.1　動作結果

6.1.1　コンボ数表示機能

リングフィットアドベンチャーの運動時の映像を本システムに入力すると，映像効果が組み込まれていることを確認できた (図 28)．左上の緑丸で囲んだ部分には赤い四角が二つ表示されているが，これは Best を 5 回検出したことを 2 進数で表示している．また，右下の緑丸で囲んだ部分には Best x5 という文字が表示されているが，これはコンボ数を示す映像効果の部分であり，画面の右端から左端まで文字が移動しながら表示されることを確認した．

現時点ではカラーフィルタリングで用いる閾値や大きさが固定値になっているため，リングフィットアドベンチャー専用となっているが，今後はこの値を外部から変更できるようにすることで汎用性を高めることも想定している．

6.1.2　文字認識機能

図 29 は PC 上の Python で実装した文字認識エンジンへ図 15 を入力として与えた際の，出力結果のテキストデータを示している．正しいテキストデータが読み取れていることから今回作成した K 近傍法による判定がうまく行っていることが分かる．ただし PS での実行はまだ出来ていない．

6.1.3　ピンチ時応援機能

リングフィットアドベンチャーでピンチになっているときの映像をシステムに入力すると，アンプモジュールに接続したイヤホンから「頑張って！」という音声が流れることを確認した．

6.2　フレーム単位の遅延評価

6.2.1　評価方法

本システムと同等のシステムを構成するためには任天堂 Switch からの映像を受け取り，受け取った映像に応じた映像処理を行う必要がある．本システムで仮に映像処理自体には時間がかからなかったとしても，HDMI デコーダ・エンコーダを含むシステム全体では遅延が発生する．そこで，映像のキャプチャを行う他のデバイスとの比較を行う．

(1). 本システムでの描画の遅れ

(2). キャプチャデバイスでの描画の遅れ

図 28: ゲーム内エフェクトに応じた映像処理の様子 (緑で囲った部分)

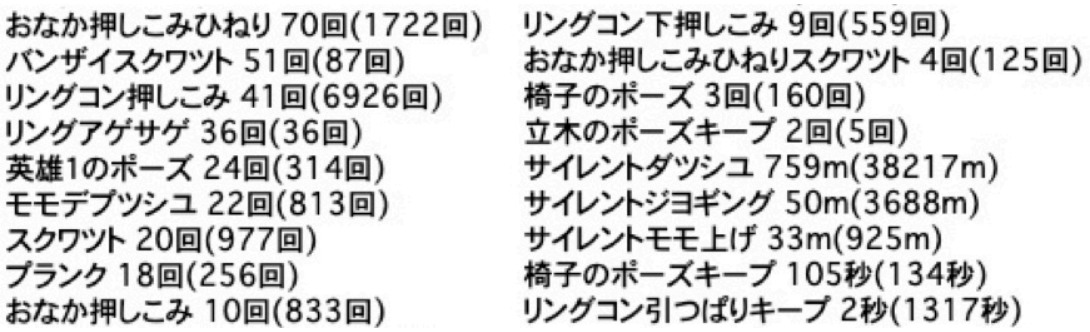


図 29: 文字認識エンジンの認識結果

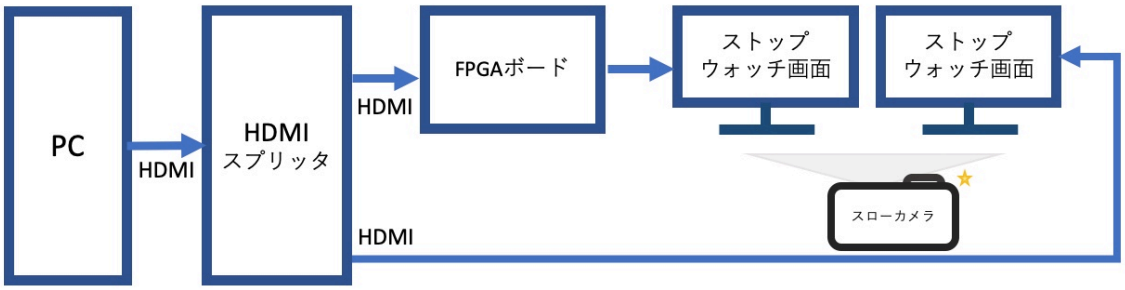


図 30: フレーム単位遅延評価時の機器構成（FPGA）

(1) の評価のために，PC でストップウォッチ画面 (1920x1080@60Hz) を出し，図 30 のように 1 つはそのままモニターに接続し，もう片方は FPGA を経由して，モニターに接続する．モニターに写った画面をスローカメラで写真を取ることで遅延時間を記録する．

(2) の評価において，キャプチャデバイスには USB カメラや USB キャプチャユニットが考えられる．これらのキャプチャデバイスの評価のために，それぞれ図 31，32 のように PC から出力されモニターに映ったストップウォッチ画面をキャプチャデバイスで読み込み，モニターに出力する．モニターに写った画面をスローカメラで記録する．

6.2.2 評価結果

本システムと USB カメラ，USB キャプチャユニットでストップウォッチ画面を読み取り，スローカメラで記録した写真を図 33-35 に示す．USB カメラと USB キャプチャユニッ

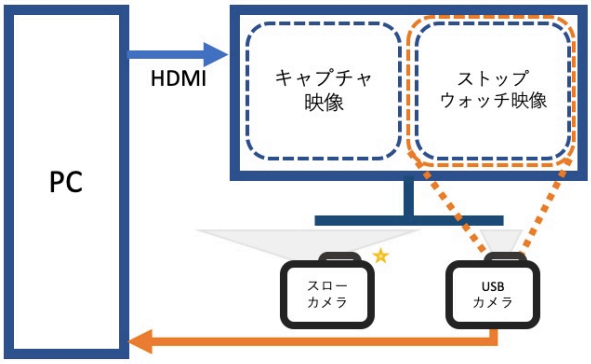


図 31: フレーム単位遅延評価時の機器構成（USB カメラ）

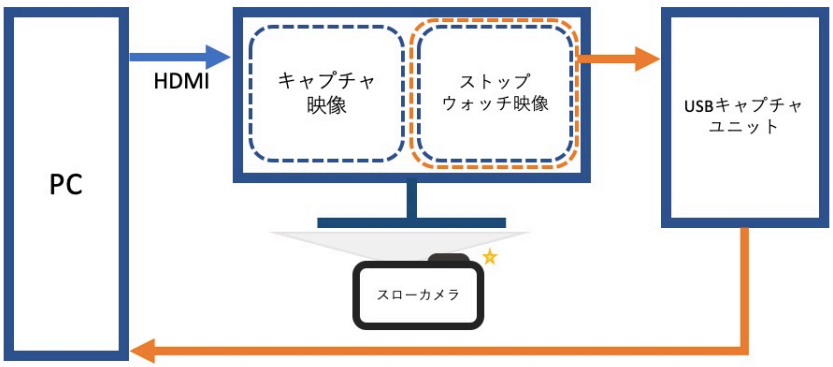


図 32: フレーム単位遅延評価時の機器構成（USB キャプチャユニット）

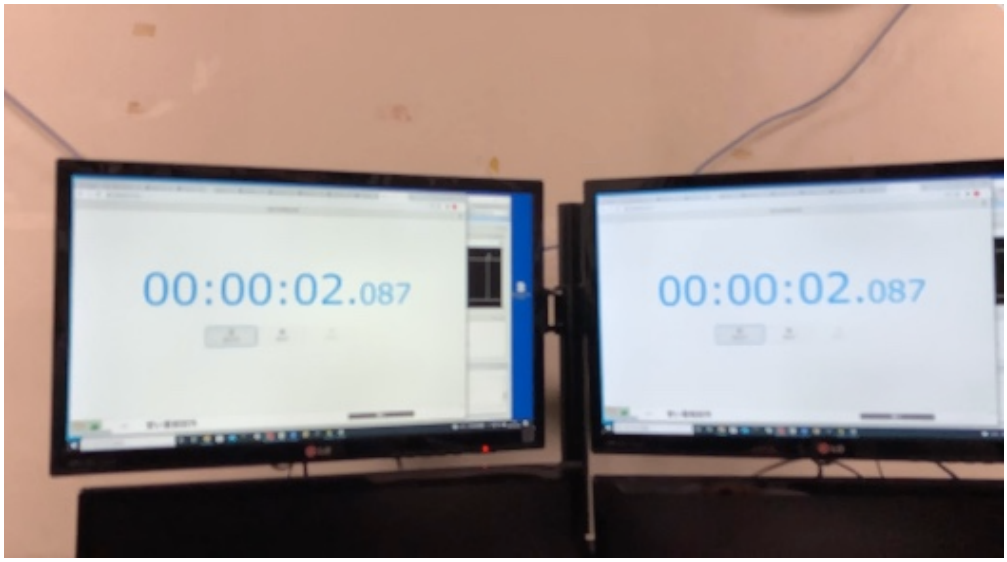


図 33: スローカメラの撮影結果 (左:FPGA，右:オリジナル)

トでは数フレーム単位の遅延が発生しているのに対し，本システムではフレーム単位での遅延が発生していないことが分かる (1 フレームは 1/60 秒).

6.3 ピクセル単位の遅延評価

6.3.1 評価方法

ストップウォッチ画面を写した画面をキャプチャして，キャプチャ後の映像をモニターに映すことでフレーム単位の遅延を確認することができた．しかし，それより小さなピクセル単位での遅延は詳細に測定できない．そこで，ピクセル単位での映像の遅延を測定する方法を提案する．

図 36 では測定に使用した機材の接続図を示している．任天堂 Switch から出力された映像信号は HDMI スプリッタに入力され，HDMI デコードを行う 2 つのデコード用 FPGA ボードで映像同期信号の HSync，VSync に変換される．それぞれの HSync，VSync は測定用 FPGA ボードで受け取り，ロジックアナライザ (ILA) で遅延の大きさを測定する．スプリッタから出力した片方に遅延の大きさを測定したい機器を挟み込み，波形のズレを確認することでピクセル単位で発生する遅延を計測できると考えた．

HDMI デコード用 FPGA として，HDMI 入出用力端子と VGA 端子のついた ZYBO[26] を使用した．また，最終的な波形のズレを確認する測定用 FPGA として NexysVideo[27] を使用した．VGA は HSync と VSync がそれぞれ 13 番ピンと，14 番ピンに対応している

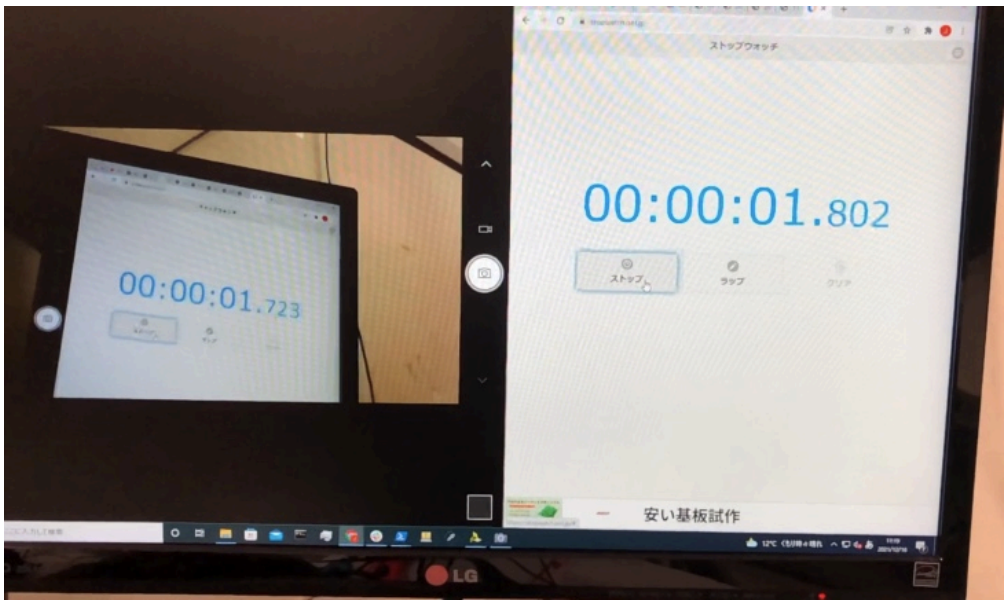


図 34: スローカメラの撮影結果 (左:USB カメラ，右:オリジナル)

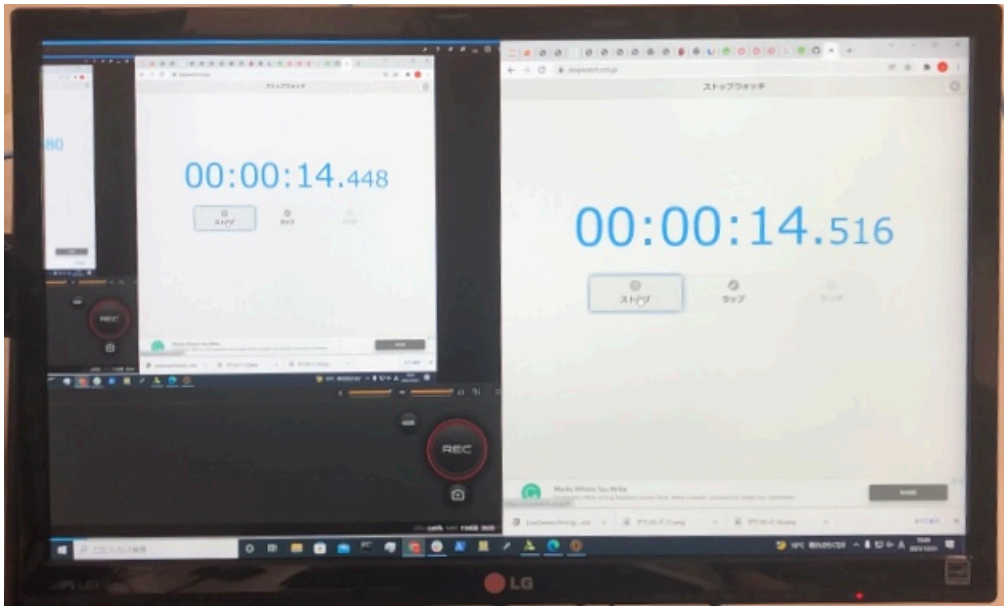


図 35: スローカメラの撮影結果 (左:USB キャプチャユニット，右:オリジナル)

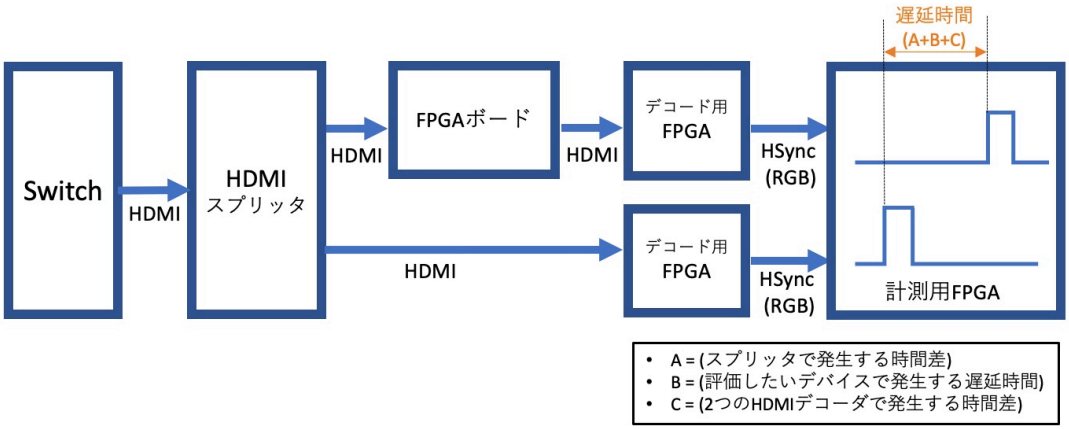


図 36: ピクセル単位の遅延測定時の機器構成

(図 37[28]). ZYBO の VGA ピンのうち，HSync と VSync に対応している 13 番ピンと，14 番ピンを NexysVideo の外部ピンに接続して入力を受け取る．2 つのデコード用 FPGA はそれぞれ非同期で動いているため，ピクセルクロックと同周波数のクロックを別途生成し，そのクロックに応じたサンプリングを行った．図 38 は実際の測定の様子を示している．

ここで図 36 で発生する全体の波形のズレについて考える．もし，スプリッターで出力される 2 つの HDMI 信号でズレが発生したり，2 つの HDMI デコーダの遅延が異なっていると，全体の波形のズレは測定したいデバイスの遅延時間に一致しない．ここで

$$A = (\text{スプリッターで発生する遅延時間差}) \tag{1}$$

$$B = (\text{評価したいデバイスの遅延時間}) \tag{2}$$

$$C = (HDMI\ \text{デコーダで発生する遅延時間差}) \tag{3}$$

とすると

$$(\text{全体の波形のズレ}) = A + B + C \tag{4}$$

と表せる．そのため，評価したいデバイスでの遅延時間 B を評価するためには A と C も測定する必要があると考えた．そこで，図 39 のように評価したいデバイスを取り除くことで遅延時間 A+C を求め，次に図 40 のようにスプリッタから HDMI デコーダまでの信号経路を構成する HDMI ケーブルを入れ替えることによって遅延時間 (-A)+C を求めることができる．これらの連立方程式を解くことによって，A と C の値をあらかじめ求めることが可能になる．

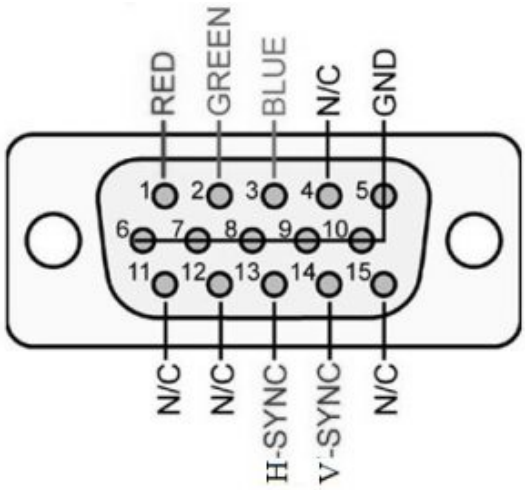


図 37: VGA コネクタのピン配置（参考文献 [28] より引用）

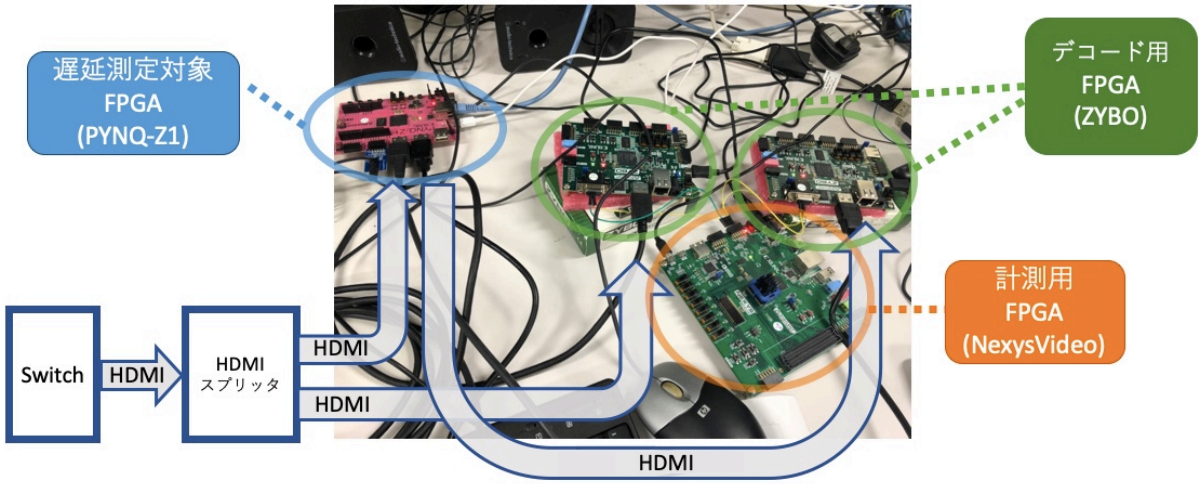


図 38: 実際の測定

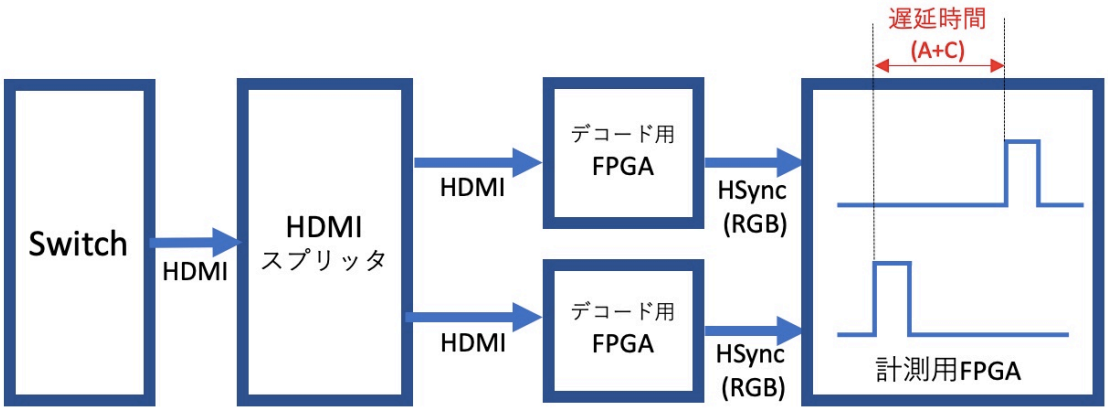


図 39: 接続切り替え前の遅延時間

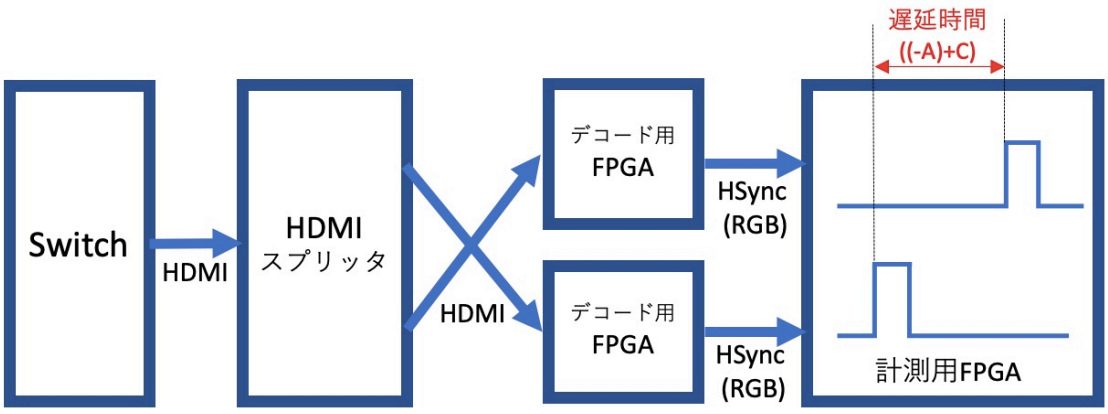


図 40: 接続切り替え後の遅延時間

6.3.2 評価結果

ケーブルの接続の仕方を変えて測定した A+C，(-A)+C，A+B+C とそれより求めた A, B，C を表にすると以下の表 3 となった．本システムのピクセル単位の遅延 (B) は 13 ピクセルとなった．また，スプリッタでの遅延時間 (A) は 1 ピクセル発生している．これはスプリッタから出力される HDMI 信号を伝達する片方の HDMI ケーブルがもう片方のケーブルよりも長さが異なることに起因していると考えている．

表 3: 遅延時間の評価結果

評価対象	遅延時間 (ピクセルクロック)
$A + C$	1
$(-A) + C$	-1
$A + B + C$	14
A	1
B	13
C	0

6.4 本資源

システムの回路資源の使用率を表 4 で示す．Block RAM だけは多くのリソースを使用しているが，他の LUT や FF はほとんど使用していない．VRAM に書き込む際の非同期 FIFO で可能な限り多くの Block RAM を使用したため，BRAM の使用率は全体の回路資源の 93.2%を占めている．

Best Good 検出・映像効果付与モジュールでのエフェクト検出では，色範囲のピクセルの検出後に水平・垂直方向の最大長を計算している．垂直方向のピクセル最大長を計算するために，水平ピクセル毎の垂直方向連続ピクセル数を記憶する必要があった．検出したいオブジェクトごとに水平ピクセル数のカウンタ 1 つと垂直ピクセル数のカウンタ×(水平ピクセル数) を分散 RAM を使用して作成したが，LUT は全体の 3.1%しか使用されていない．そのため回路資源の観点から本システムはさらに多くのオブジェクトを検知することが可能であることが分かった．

ピンチ検知・応援音声出力モジュールでは LUT と FF の消費率は 1%以下でほとんど使用されていないが，代わりに 2.9%の Block RAM を使用している．これはピンチ時に再生される音声データが書き込まれているからである．

表 4: 回路資源の使用状況

モジュール名	LUT (%)	FF (%)	BRAM (%)
HDMI 入出力	724 (1.4)	576 (0.5)	1.0 (0.7)
キャプチャモジュール	3,711 (7.0)	5,281 (5.0)	130.5 (93.2)
Best Good 検出・映像効果付与	1,643 (3.1)	152 (0.1)	0.0 (0.0)
ピンチ検出・応援用音声生成	155 (0.3)	106 (0.1)	4 (2.9)
使用可能な回路資源	53,200 (100)	106,400 (100)	140.0 (100)

7　おわりに

7.1　まとめ

本論文では任天堂リングフィットアドベンチャー向けの運動モチベーション向上システムを提案した．機能の拡張が本来難しかったコンシューマーゲームに対し，まず，映像信号を介したデータの取得，映像処理を行うことでコンボ数の付加表示を行った．連続して Best となる運動をしても元々のゲームではあまり変化がなかったが，分かり易い表示をすることでユーザーの正しい運動を行うモチベーションの向上を狙った．次に，運動の種類ごとの結果を文字認識して取得できることを示した．これらのデータをネット上で共有するなどすれば，ユーザー間での運動モチベーションを向上させる新たな仕組みを開発することも容易になると考えられる．また，ピンチ時に応援音声を出力する機能を追加した．好きなアイドルや声優の声を出力することによってゲーム自体にさらなる楽しさを追加できると考えた．

今回の仕組みを用いて FPGA 付テレビを開発すれば，ゲーム機の画面やテレビ番組の映像など，既存の映像ソースを後から修正して付加価値を付けることが可能になる．FPGA の利点を利用してプラグインのような形で映像ソース毎に提供すれば，同じ映像でも新たな楽しみを提供出来る可能性がある．

7.2　ユースケースの提案

FPGA 付テレビが普及した際のプラグインのアイデアを二つ挙げておく．一つ目は，天気予報における猫のじゃれつき防止機能である．テレビの天気予報では気象予報士が支持棒を使用して天気の説明をする．家に猫がいる家庭では，気象予報士の棒の素早い動きに反応し，猫が支持棒の映っているディスプレイを叩いたり，傷をつけることが考えられる．本システムを用いて天気予報士の棒の色を検知し，棒を周辺の色と同化させ棒を消すことによって，猫が画面を傷つけることを防止することが可能になる．

二つ目は，球技スポーツの観戦支援である．サッカーやテニスなどの球技スポーツにおいて，あらかじめボールの色の範囲を指定してボールを検知させる．検出したボールに対し色の変化をさせボールを見やすい色に変化させたり，数秒前のボールの位置を表示させてボールの軌道を確認したりすることで，スポーツ観戦の支援を行う．

7.3 今後の課題

7.3.1 協力プレイ

運動では他人と競争するよりも協力することによって運動モチベーションが向上する[7]．そのため，コントローラーを改造して同時に複数人でリングフィットアドベンチャーをプレイできるようにすることで，よりリングフィットアドベンチャーにとって効果的な機能拡張を行える可能性がある．

7.3.2 CEC を使ったモニター操作

CEC(Consumer Electronics Control) は HDMI 信号の中に流れるモニターの制御信号である．FPGA から CEC の信号を操作することによって，ゲーム映像の音量や明るさ，モニターの電源の操作が可能になる．

参考文献

[1] Hasan Erdem Mumcu. Fitness-Related on Health Mobile Applications during COVID-19: Case of Turkey. Vol. 23 No. S1 (2021): Supplement 1/2021: Physical Activity, Health and Sports. 2021.

[2] 任天堂. Nintendo Switch. https://www.nintendo.co.jp/hardware/switch/. (最終アクセス日 2022/01/27)

[3] 任天堂. リングフィット アドベンチャー. https://www.nintendo.co.jp/ring/. (最終アクセス日 2022/01/27)

[4] 株式会社ポケット. FiNC HOME FiT（フィンクホームフィット）. https://www.pckt.co.jp/homefit/. (最終アクセス日 2022/01/27)

[5] ポケット公式チャンネル. 『FiNC HOME FiT』 プロモーションムービー（発売前 90 秒）- YouTube. https://youtu.be/0wcW6x-q0Ac. (最終アクセス日 2022/01/27)

[6] Fit Boxing（フィットボクシング）. https://fitboxing.net/. (最終アクセス日 2022/01/27)

[7] Yu Chen, Pearl Pu. HealthyTogether: exploring social incentives for mobile fitness applications. Proceedings of the Second International Symposium of Chinese CHI (Chinese CHI). (pp.25-34). 2014.

[8] Fit Boxing 2（フィットボクシング 2）. https://fitboxing.net/2/. (最終アクセス日 2022/01/27)

[9] Fit Boxing 公式アプリ. https://fitboxing.net/2/app/. (最終アクセス日 2022/01/27)

[10] Nintendo 公式チャンネル. リングフィット アドベンチャー 紹介映像 - YouTube. https://youtu.be/qXfpVh7VI_c. (最終アクセス日 2022/01/27)

[11] Mapwriter 2 - Mods - MinecraftCurseForge. https://www.curseforge.com/minecraft/mc-mods/mapwriter-2. (最終アクセス日 2022/01/27)

[12] スマッチサイレント - 株式会社 ZAURUS（ザウルス）. https://www.zaurus.jp/products/zaurus/2135.php. (最終アクセス日 2022/01/27)

[13] RingFit_Ranker. RingFit_Ranker(リングフィットランカー) (@RingFitRanker) - Twitter. https://twitter.com/ringfitranker. (最終アクセス日 2022/01/27)

[14] IkaLog. hasegaw/IkaLog - GitHub.
https://github.com/hasegaw/IkaLog. (最終アクセス日 2022/01/27)

[15] Takeshi HASEGAWA.「スプラトゥーン」リアルタイム画像解析ツール「IkaLog」の裏側 - SlideShare. https://www.slideshare.net/TakeshiHasegawa1/20151016ssmjpikalog. (最終アクセス日 2022/01/27)

[16] Auangkun Rangsikunpum, Ekachai Leelarasmee, Suree Pumrin. A Design of Sign Video Image Expander for HDMI Source using Bicubic Interpolation. 2017 14th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON). (pp.171-174). 2017.

[17] Digilent. ZedBoard - Digilent Reference. https://digilent.com/reference/programmable-logic/zedboard/start. (最終アクセス日 2022/01/27)

[18] Digilent. PYNQ-Z1 - Digilent Reference. https://digilent.com/reference/programmable-logic/pynq-z1/start. (最終アクセス日 2022/01/27)

[19] Andrew E. Wilson, Michael Wirthlin. Reconfigurable Real-Time Video Pipelines on SRAM-based FPGAs. 2019 International Conference on ReConFigurable Computing and FPGAs (ReConFig). 2019.

[20] Han Sung Lee, Jae Wook Jeon. Accelerating Image Processing on FPGAs using HLS and PYNQ. 2020 IEEE International Conference on Consumer Electronics - Asia (ICCE-Asia). 2020.

[21] T. Fang, X. Huang, J. Saniie. Design Flow for Real-Time Face Mask Detection Using PYNQ System-on-Chip Platform. 2021 IEEE International Conference on Electro Information Technology (EIT). (pp.378-382). 2021.

[22] D. Tsiktsiris, D. Ziouzios, M. Dasygenis. HLS Accelerated Noise Reduction Approach Using Image Stacking on Xilinx PYNQ. 2019 8th International Conference on Modern Circuits and Systems Technologies (MOCAST). 2019.

[23] hamsternz. hamsternz/Artix-7-HDMI-processing - GitHub.
https://github.com/hamsternz/Artix-7-HDMI-processing. (最終アクセス日 2022/01/27)

[24] Tesseract OCR. tesseract-ocr/tesseract - GitHub.
https://github.com/tesseract-ocr/tesseract. (最終アクセス日 2022/01/27)

[25] Pmod AMP2: Audio Amplifier. - Digilent.
https://digilent.com/shop/pmod-amp2-audio-amplifier/. (最終アクセス日 2022/01/27)

[26] Digilent. Zybo - Digilent Reference.
https://digilent.com/reference/programmable-logic/zybo/start.
(最終アクセス日 2022/01/27)

[27] Digilent. NexysVideo - Digilent Reference. https://digilent.com/reference/programmable-logic/nexys-video/start. (最終アクセス日 2022/01/27)

[28] What is a VGA Connector : Pin Configuration & Its Applications - ElProCus.
https://www.elprocus.com/vga-connector/. (最終アクセス日 2022/01/27)

謝辞

本研究を進めるにあたり，ご指導いただいた指導教員の成見先生，副指導教員の佐藤先生に感謝いたします.