# 隠れマルコフモデルによる時系列データ分類器のアクセラレータ自動生成

藤岡裕展[1,a)]　趙　謙[2,b)]　久我 守弘[2,c)]　尼崎 太樹[2,d)]　飯田 全広[2,e)]　末吉 敏則[2,f)]

**概要**：近年，隠れマルコフモデル (HMM）を用いた時系列データ解析器のハードウェア（HW）アクセラレータの研究が盛んに行われている．しかし，HMM では時系列データの学習結果によって分類器の構造が異なるため，対象データ毎に HW を設計する必要があった．そこで，本研究では HMM の FPGA（Field Programmable Gate Array）による HW アクセラレータを自動生成するシステムを提案する．提案システムでは，まず入力された学習用データに対して学習を行い，高位合成可能な C++ソースコードを生成する．次に高位合成により IP 化を行う．そして，我々の研究室で開発している設計環境である hCODE（heterogeneous Computing Oriented Development Environment）を用いて IP と PCIe や AXI バスとのインタフェースを接続し，FPGA 用構成情報ファイルを生成する．評価では，SW 実行に比べ約 12 倍の高速化を実現した．

**キーワード**：HMM，異常検出，FPGA，高位合成

# Automatic Accelerator Generation for Time Series Data Analysis using Hidden Markov Model

Fujioka[1,a)]　Cho Ken[2,b)]　Kuga Morihiro[2,c)]　Amagasaki Motoki[2,d)]　Iida Masahiro[2,e)]
Sueyoshi Toshinori[2,f)]

***Abstract:*** In recent years, design and implementation of the HW accelerators for time series data analysis using hidden Markov models (HMM) has been actively reserching. However, in the HMM, the structure of the classifier differs depending on the learning result of time series data, so it was necessary to design HW accelerator for each target data. Therefore, in this research, we propose a system which automatically generates HMM HW accelerator using Field Programmable Gate Array (FPGA) . In the proposed system, HMM learning is first performed on input learning data, and C ++ source code capable of high-level synthesis is generated. Next, modularized IP is performed by high-level synthesis. Then, by using hCODE (heterogeneous Computing Oriented Development Environment), the interface from IP to PCIe and AXI bus is connected, and a bit stream file for FPGA configureation is generated. In the evaluation, we achieved 12 times faster than SW execution.

***Keywords:*** HMM, Anomaly Detection, FPGA, High Level Sinthesis

[1] 熊本大学 大学院自然科学研究科
2-39-1 Kurokami, Chuo, Kumamoto 860–8555, Japan
[2] 熊本大学 先端科学技術研究部
2-39-1 Kurokami, Chuo, Kumamoto 860–8555, Japan
[a)] fujioka@arch.cs.kumamoto-u.ac.jp
[b)] cho@cs.kumamoto-u.ac.jp
[c)] kuga@cs.kumamoto-u.ac.jp
[d)] amagasaki@cs.kumamoto-u.ac.jp
[e)] iida@cs.kumamoto-u.ac.jp
[f)] sueyoshi@cs.kumamoto-u.ac.jp

## 1. はじめに

近年，Internet of Things（IoT）に代表される情報化社会において，大規模センサネットワークが構築されビッグデータを扱う需要が急激に増加している．大量のセンサノードから収集した膨大なデータを解析し，実世界の多様な問題解決に役立てている．その中でも，時系列データに

対して統計的手法を用いて異常検知を行う研究が盛んに行われれいる．時系列データに対する統計的異常検出手法として主に，外れ値検出，変化点検出および異常行動検出の３つがある．本研究では特に異常行動検出に着目をする．異常行動検出とは，時系列データの中から異常なセッションや行動パタンを検出することであり，代表的な手法として HMM（Hidden Markov Model）[1] がある．HMM とは時系列データの中に隠れている特定パタンを検出する確率モデルであり，事前学習によりパラメータを決定する教師有り機械学習の手法の一種である．古くから音声認識の分野で用いられてきたが，近年はタンパク質の塩基配列の解析や時系列センサデータの解析等幅広い分野で応用されている．

近年では，ビッグデータに対する解析処理の高速化が課題となっており，アルゴリズムの改良 [2] や，専用 HW（HardWare）を設計する手法 [2] が提案されている．専用 HW 化する手法の 1 つとして, FPGA（Field Programmable Gate Array）によるアクセラレータ設計が注目を集めている．FPGA は回路構成を再構成可能な集積回路であり，並列処理可能なアプリケーションに対してプロセッサに比べ高速かつ低消費電力に処理が可能である．従来より通信や動画像処理の分野で用いられてきたが，近年ではサーバ用途での使用も広がっている [4]．先行研究として，Chris ら [5] はタンパク質の塩基配列解析専用 HW を開発した．Intel XeonCPU に対し 16 倍の高速化，消費電力性能では 3.8 倍の性能を示した．しかし，FPGA による専用 HW 設計において SW（Software）設計に比べて開発期間が長いことが問題として挙げられる．また，HMM では対象データごとにパタンを分類する構造が異なるため，データ毎に HW を設計しなければならないという問題もある．さらに FPGA では，SW 設計と違い分類器の設計において回路資源が限られているという問題がある．そこで，本研究では HMM の FPGA による HW アクセラレータを自動生成するシステムを提案する．提案システムでは，生成する HW アクセラレータは 2 種類ある．1 つはスループットが高く回路資源消費が多いアクセラレータ，2 つ目はスループットは低く回路資源消費は少ないアクセラレータである．提案システムでは学習データを入力し，FPGA で実行可能な構成情報ファイルを出力する．本提案システムにより，設計者の負担を軽減するとともに，HW 設計者でなくても容易に HMM のアクセラレータを利用することが可能である．

## 2. 隠れマルコフモデル（HMM）

HMM は時系列データをモデル化する統計的手法であり，ノイズに対するロバスト性が強いという特徴がある．本研究では離散量の時系列データを分類するため，Gaussian HMM について取り扱う．HMM は，初期状態確率 $\pi$，状態遷移確率 $A$, シンボル出現確率 $B$ の 3 つのパラメータを持ち，シンボル出現確率は正規分布で与えられる．また HMM では主に 3 つのアルゴリズムがある．モデルの尤度を求める Forward-Backward アルゴリズム，モデルパラメータの学習を行う Baum-Welch アルゴリズムおよび時系列データの分類を行う viterbi アルゴリズムである．本研究では viterbi アルゴリズムの専用 HW の自動生成を行う．

### 2.1 viterbi アルゴリズム

viterbi アルゴリズムは動的計画法の一種で，時系列データから最適な状態（パタン）系列を求める．viterbi アルゴリズムは 4 つのステップからなる. 主な記号と定義を表 1 に示す.

**表 1** 主な記号と定義

| 記号 | 定義 |
|---|---|
| $x_t$ | 観測系列 |
| $\pi_i$ | 時刻 1 で状態 i に遷移する確率 |
| $a_i, j$ | 状態 i から j に遷移する確率 |
| $b_i$ | 状態 i において $x_t$ が出現する確率 |
| $\psi_t(i)$ | 時刻 t で状態 i で $x_t$ が観測される確率 |
| $\omega_i$ | 状態 i |
| $\varphi(i)$ | t-1 の状態のうち $\psi_t(i)$ を最大にするもの |
| $s^*$ | 最終状態 |
| $P(x, s^*)$ | 観測系列 x が得られたとき最終状態が $s^*$ である確率 |
| $i_n$ | 最終状態のラベル |

(1) STEP1：初期化
時刻 1 での観測確率を求める．ここで観測確率とは，ある時刻 t で状態 $\omega_j$ に到達し，かつシンボル $x_t$ が観測される確率のことである．

$$\psi_1(i) = \pi_1 b(\omega_1, x_1) \tag{1}$$

(2) STEP2：再帰的計算
時刻 t における観測確率を各状態において求める．その際，各状態における観測確率の最大値を保存する．

$$\psi_t(j) = max\{\psi_{t-1}(i)a_{i,j}\}b(\omega_j, x_t) \tag{2}$$

$$\varphi(j) = argmax\{\psi_{t-1}(i)a_{i,j}\} \tag{3}$$

(3) STEP3：終了
STEP2 で求めた最終状態から最終状態を決定する．

$$P(x, s^*) = max\psi_n(i) \tag{4}$$

$$i_n = argmax\psi_n(i) \tag{5}$$

(4) STEP4：経路探索
STEP3 で求めた最終状態を起点に STEP2 で保存していた最大パスを辿る．t=1 までパスを辿れば終了する．

$$i_t = \varphi_{t+1}(i_{t+1}) \tag{6}$$

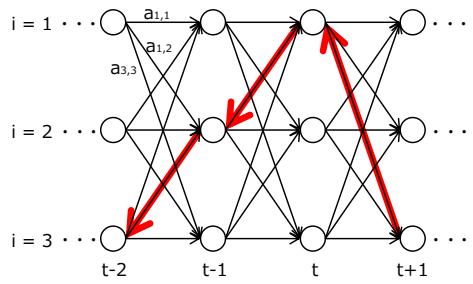


図 1　HMM 状態遷移

$$s_t^* = \omega_{i_t} \tag{7}$$

図 1 に縦軸を各状態，横軸を時間で並べた viterbi アルゴリズムの構造を示す．図 1 の例は，状態数 3 の HMM モデルである．各状態にはそれぞれ 3 本の矢印が伸びており，これらは STEP2 における観測確率を求める状態遷移確率とシンボル出現確率の積の計算を表す．従って，各時刻 t で 9 回観測確率を計算する．また，その際に各状態における観測確率の最大値を保存しておく．その後 STEP3 で求めた最終状態を起点に，STEP4 では太矢印で表すようにその最大値を逆に辿っていく．これにより，時系列データがどのような状態系列を生成しているか求める．

### 2.2　関連研究

HMM はセンサデータ処理の手法として様々な分野において広く活用されている．Jay ら [3] は音声認識の分野で HMM を使用し，吐息等の雑音が混じっても音声の高い認識率を実現した．Wang ら [6] は，HMM を改良した pHMM（patern-based hidden Marlcov Model） 提案しセンサデータの動的パターンを表現している．Nico ら [7] は，HMM を用いて異常検知に応用した HMAD を提案しており，生物学における実データに対して高い検出性能を示した．Chris らは HMM によるたんぱく質の塩基配列解析を FPGA にオフロードし，CPU や GPU に対し速度と消費電力共に高い性能を示した．

従来の FPGA による HMM アクセラレータ設計では，特定の時系列データ分類の高速化に焦点がしぼられてきた．しかし，本研究ではデータの学習から IP 生成までを行うシステムを提案することで，多様なデータに対するアクセラレータを作成することを目指す．

## 3.　提案システム

本章では，提案システムについて述べる．

### 3.1　提案システムの概要

図 2 に提案システムの設計フローを示す．提案システムでは時系列データを入力し，その時系列データを分類可能な HW アクセラレータのための FPGA の構成情報ファイルを出力する．具体的には，まず（1）入力データに対し学習を行い分類器パラメータや構造を決定し，HW 化に必要な情報を得る．次に，（2）学習で得られた情報をもとに，高位合成可能な C++ソースコードを生成する．そして，（3）生成されたソースコードから高位合成ツールである Vivad HLS[9] を用いて IP 化を行う．IP 化した分類器は，hCODE[11] を用いて HW システムに組み込む．最後に，（4）論理合成を行い構成情報ファイルを作成する．本研究では，（1）～（3）の工程を自動化することでアクセラレータの作成を容易にしている．また，アクセラレータ作成において 2 種類の HW を作成可能としている．通常はスループット重視の HW 用コードを生成する．しかし，（2）のコード生成の際にパラメータを 1 部変更することで回路資源使用率を重視した HW 用のコード生成を選択可能としている．

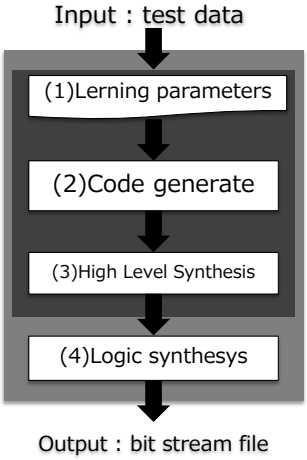


図 2　提案システムのフロー

### 3.2　hCODE

hCODE は，本研究室で提案しているヘテロジニアスな環境における FPGA を用いた HW アクセラレーション開発を行うことができるプラットフォームである．図 3 に機能スタックを示す．ハードウェア階層は，アプリケーションに依存しない Shell と，特定のアプリケーション専用の IP から構成される．Shell は通信やメモリといったアプリケーションに共通する機能を持つ．ハードウェアを Shell と IP に分けて設計することで，様々な場面に対応して実装が可能である．ソフトウェア階層では，Shell と IP 開発者が作成したドライバを介してアプリケーションから IP を利用することが可能である．本研究では，HMM の IP，IP ドライバ，IP を利用するアプリケーションを作成する．Shell と Shell ドライバに関しては，hCODE で利用可能であるサードパーティのフレームワークであり，Xillybus[10] を使用する．Xillybus は，FPGA で作成した IP コアを DMA（Direct Memory Access）や PCI-Express

を通じて Windows/Linux PC からアクセス可能にしたインターフェースである．ホスト PC からのアクセスはデバイスファイルを通じて行われ，様々なプログラミング言語から利用可能である．本研究では，IP コアと Xillybus とのインタフェーイスは AXI stream を利用し，ホスト PC と Xillybus のインタフェースは hCODE を用いる．

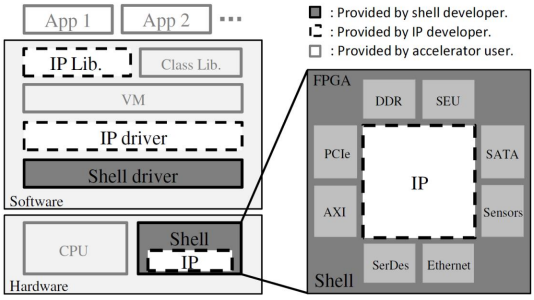


図 3　hcode の機能スタック

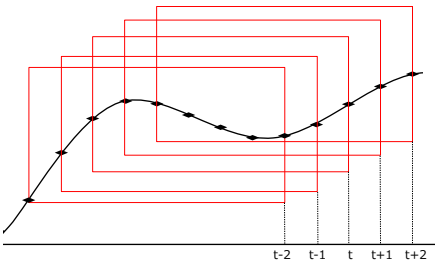


図 4　ウインドウ移動

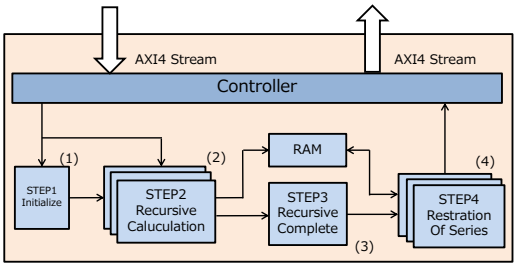


図 5　HMM のブロック図

### 3.3　hmmlearn

hmmlearn[12] は python のオープンソースライブラリであり，GaussianHMM をサポートしている．本研究では HMM の学習に利用しており，Baum-welch アルゴリズムでパラメータを推定している．Baum-welch アルゴリズムとは HMM におけるパラメータ推定のアルゴリズムで，山登り法の 1 種である．Baum-welch アルゴリズムでは局所解に陥ってしまう可能性があるが，hmmlearn ではパラメータ推定の際に初期値を自由に設定できる．そのため，初期値を変更しながら学習を行うことで最適なパラメータを推定することが可能である．

### 3.4　ソースコード生成

ソースコード生成の説明にあたり，今回設計する HW の説明を行う．設計する HW は 2 種類ある． 1 つはスループットが高く回路資源消費が多いアクセラレータ，2 つ目はスループットは低く回路資源消費は少ないアクセラレータである．基本的に設計手順は同じであるため，HW 化の最後に両者の違いを説明する．

#### 3.4.1　HW 化

viterbi アルゴリズムの HW 化にあたり，スループット向上のためウインドウサイズを固定とした．ウインドウとは 1 度に分類するデータの数を意味する．図 4 に示す通りウインドウを 1 ずつスライドさせながら処理をしていく．スライドする度にウインドウの右端のデータを入力し，探索し終わった最後の状態を出力する．

一般的に viterbi アルゴリズムの実装においては，観測確率の計算の際に桁落ちや精度の低下を防ぐため対数をとる．しかし，HW での対数演算はスループットおよび回路資源の面で好ましくない．そこで本研究では，桁落ちが発生しない範囲でウインドウサイズを固定とすることで，対数演算器を用意せずに観測確率の計算を行う．そこで，対数をとる場合およびとらない場合での精度の比較の予備実験を行った．hmmlearn での分類結果を正解データとし，C++で作成した対数有りと対数なしの Viterbi アルゴリズムでの精度を比較した．その結果，分類結果の一致率は対数有の場合が 89%，対数なしの場合が 87%であった．しかし，分類結果では，波形の立ち上がりおよび立下りの特徴点はすべて分類できていた．このことから，精度は少し落ちるものの HW 化において高速化を見込めると判断した．

図 5 にアクセラレータのブロック図を示す．図中の矢印はデータの送信方向を表し，四角はモジュールを表す．アクセラレータの処理の流れについて説明する．処理の順番は，2.2 節で説明した viterbi アルゴリズムのステップ（1）〜（4）に対応している．

まず，AXI STREAM インタフェースからデータを 1 サイクル毎に入力し，シフトレジスタに格納する．次に，（1）初期化と（2）再帰的計算モジュールにデータを転送し，それぞれ観測確率を計算する．この時，GaussianHMM ではシンボル出現確率が正規分布で与えられている．そのため FPGA における HW 化において正規分布の計算はレイテンシが長くなってしまう上に，DSP（Degital Signal Processer）や FF（Flip Flop）を大量に消費してしまう．そこで，本研究では予め正規分布のテーブルを LUTRAM（Look Up Table RAM）に格納し同時アクセスすることで，

計算回数と回路資源の削減を行った．また，各状態における観測確率の最大値となるパスを RAM に保存しておく．そして，（3）シーケンスの最終状態を導出後に（4）経路探索を行う．経路探索の際には，RAM に保存された各状態の最大値のパスを辿っていく．HW 化にあたり，STEP2 の再帰的計算と STEP4 の経路探索をウインドウサイズ分並列化することでスループット向上を実現する．

設計する 2 種類の HW においては，観測確率の最大値となるパスを保存しておく RAM に違いがある．まず，スループット向上を重視した HW では，RAM に LUTRAM を用いる．LUTRAM は多数の同時アクセスが 1 サイクルで可能であるため，処理全体をパイプライン化することが可能である．しかし，ウインドウサイズを拡大していくと DSP の資源使用率がネックとなる．また，回路資源消費を重視した HW では BRAM（Block RAM）を用いる．BRAM は FPGA 内部のメモリ領域を利用して構成される大規模なメモリである．しかし，BRAM の入出力ポートには制限があり，多数の同時アクセスでは数サイクルかかる．そのため，処理全体をパイプライン化できない．しかし，BRAM へのアクセスの間に複数の演算で 1 つの DSP を共用して演算を行えるため，DSP の回路資源消費を抑えることができ，LUTRAM を用いた HW に比べ大きなウインドウサイズで分類することが可能である．

#### 3.4.2 高位合成

高位合成とは C/C++などの高級言語で HW の機能設計を行う技術であるしかし，通常のソフトウェア設計と異なり HW を意識した設計が求められるため，専用のディレクティブ等を用いて明示的な記述を行う必要がある．加えて，高位合成では HDL 設計に比べサイクル単位での制御や回路資源を最適に管理する設計を行うことは難しいという課題がある．今回は，Xilinx 社が提供している高位合成ツール Vivado HLS を使用する．C/C++で記述されたファイルから HDL コードを出力し，IP としてパッケージ化を行う．

#### 3.4.3 コード生成

HMM の学習の際に得られるパラメータと状態数から高位合成可能な C++ソースコードを出力する．コード生成では，あらかじめ C++で HMM のモデルの構造を用意しておき，各パラメータや系列長などの情報を定数マクロを書き換えることで生成する．その際に，HLS のディレクティブや HW 構成を一部明示的に記述することで HW での高速化を実現する．また，コード生成の際にパラメータを 1 部変更することで BRAM もしくは LUTLAM の使用を選択可能としている．

## 4. 評価

HMM アクセラレータの性能評価については，hCODE を用いて実装したホスト PC-FPGA のシステムとホスト PC で行う．ホスト PC では python の hmmlearn ライブラリを評価する．評価項目については，ホスト PC-FPGA のシステムとホスト PC のスループットおよび PC-FPGA のシステムにおけるインターバルサイクルと回路資源の関係である．今回評価を行った HMM の HW アクセラレータについて，以下の表 2 に示す．作成したアクセラレータは 2 種類あり，HW1 がインターバルサイクル 1 の HW，HW2 がインターバルサイクル 4 の HW である．また，レイテンシに関しては，高位合成で合成後の見積りである．

**表 2** 評価する HW の仕様

| アクセラレータ名 | HW1 | HW2 |
|---|---|---|
| 状態数 | 3 | 3 |
| インターバルサイクル | 1 | 4 |
| 動作周波数 | 150[MHz] | 150[MHz] |
| レイテンシ | 375[cycle] | 415[cycle] |
| RAM 構成 | LUTRAM | BRAM |

### 4.1 評価環境

今回使用した FPGA は Xilinx 社製 Vertex-7 XC7VX485T-2FFG1761C を搭載する VC707 評価キット [13] を用いた．アクセラレータの開発には，Xilinx 社が提供している Vivado 2015.4[14] を使用した．動作周波数は 150MHz とし，ホスト PC とは PCI Express 2.0（Gen2）で接続する．

ホスト PC は，プロセッサに Intel Core i7-2600@3.4GHz，メモリ 12GB を搭載し動作 OS は Ubuntu 14.04LTS（64bit）である．

ホスト PC-FPGA のシステムの評価の際には，C 言語で記述されたプログラムから FPGA にデータを転送し評価を行う．ホスト PC での SW 実行の際には，python の hmmlearn ライブラリを用いて評価を行う．

評価に用いたデータセットは，HMM を用いた時系列データ解析に用いられる Google Trends[13] のデータとした．データセットが評価を行うには短かったため，同一のデータセットを連結してテストデータを作成した．

### 4.2 評価結果

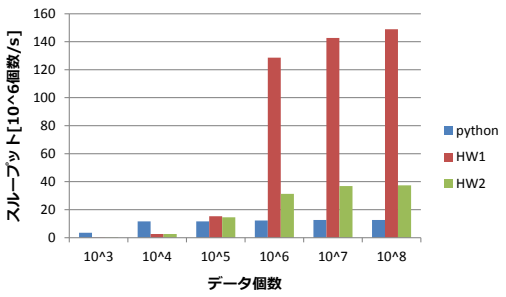


**図 6** PC-FPGA のシステムとホスト PC のスループット

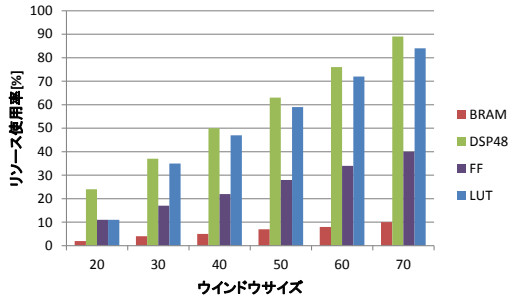


図 7　インターバルサイクル 1 の HW の回路資源使用率

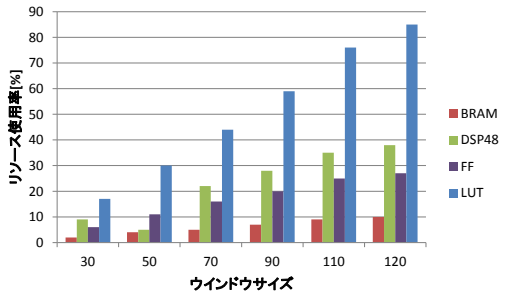


図 8　インターバルサイクル 4 の HW の回路資源使用率

図 6 に PC-FPGA のシステムとホスト PC のスループットについて示す．横軸はデータ個数，縦軸は 1 秒あたりに処理するデータ個数を対数で表現している．PC-FPGA のシステムは python の SW 実行に比べ，インターバルサイクル 1 の HW で約 12 倍，インターバルサイクル 4 の HW で約 3 倍の性能を実現した．

また，図 7 に HW1 の HW 回路資源使用率，図 8 に HW2 の HW 回路資源使用率を示す．インターバルサイクルが 1 である HW の回路資源使用率は DSP 使用率がいちばん高く，ウインドウサイズ拡大のネックになっていることがわかる．また，インターバル 4 である HW の回路資源使用率では LUT が一番回路資源使用率が高いことがわかる．これは，複数の演算で 1 つの DSP を共用するため，演算の前後の値を保持しておくメモリが増加したためであると考えられる．

まず，DSP 使用率については HW1 にに比べ HW2 は 25%程度に削減できていることがわかる．また，LUTRAM の使用率に関してはインターバルサイクル 4 の HW の方が多くなっている．これは，複数の演算で 1 つの DSP を共用するため，演算の前後の値を保持しておくメモリが増加したためであると考えられる．

## 5. まとめ

本研究では，HMM を用いた時系列データ分類のアクセラレータ自動生成システムを提案した．提案システムでは，スループットおよび回路資源の使用率を重視した 2 種類の HW を生成した．実装に関しては，hCODE を用いてホスト PC-FPGA のアクセラレータを実装した．スループットは python の SW 実行に比べ最大 12 倍の高速化を達成した．回路資源使用率を重視した HW においても最大 3 倍の高速化を達成した．このことから，多様なデータに対しても学習から IP 化まで自動化し，スループット重視の HW と回路資源使用率重視の HW の 2 種類を選択できるようにすることでアクセラレータを容易に作成可能であることを示した．

## 参考文献

[1] 石井健一郎，上田修功，“続・わかりやすいパタン認識教師なし学習入門，” Ohomsha，2014.8.
[2] 松原靖子，櫻井保志，吉川正俊，“隠れマルコフモデルに基づくストリーム処理，” 情報処理学会論文誌，データベース　Vol4 No.4 pp48-62 2011.
[3] Richard Veitch, Louis-Marie Aubert, RogerWoods, and Scott Fischaber,“FPGA Implementation of a Pipelined Gaussian Calculation for HMM-Based Large Vocabulary Speech Recognition,” Hindawi Publishing Corporation International Journal of Reconfigurable Computing ,Volume 2011.
[4] Microsoft，“Large-Scale Reconfigurable Computing in a Microsoft Datacenter，” https://www.microsoft.com/en-us/research/wp-content/uploads/2014/06/HC26.12.520-Recon-Fabric-Pulnam-Microsoft-Catapult.pdf，2017.1 閲覧.
[5] Altera.，“ OpenCL と FPGA によるゲノミクス研究のアクセラレーション，” https://www.altera.co.jp/content/dam/altera-www/global/ja_JP/pdfs/literature/wp/wp-01262-accelerating-genomics-research-with-opencl-and-fpgas_j.pdf，2016.12 閲覧.
[6] J. G. Wilpon, L. R. Rabiner, C. H. Lee, and E. R. Goldman，“Automatic recognition of keywords in unconstrained speech using hidden Markov models,” IEEE Transactions on Acoustics, Speech, and Signal Processing, 1990.
[7] P. Wang, H. Wang, and W. Wang. “Finding semantics in time series,” In SIGMOD Conference, pp 385-396，2011.
[8] Nico GNornitz,Mikio Braun,Marius Kloft,“Hidden Markov Anomaly Detection,” International Conference on Machine Learning, Lille, France, 2015.
[9] Xilinx Inc.，“Vivado HLS ユーザーガイド，”
[10] XILLIBUS，IP cores and desighn services，http://xillybus.com/，2017.1 閲覧
[11] 中道拓也，趙謙，尼崎太樹，飯田全広，久我守弘，末吉敏則，“hCODE：FPGA アクセラレータのためのオープンソースプラットフォーム，” 信学技報 RECONF2016-34，vol.116，no.210，pp45-50，Sep. 2016.
[12] hmmlearn,“Simple algorithms and models to learn HMMs（Hidden Markov Models）in Python，” http://hmmlearn.readthedocs.io/en/latest/index.html，2016.12 閲覧.
[13] Xilinx Inc.，“VC707 Evaluation Board for the Virtex-7 FPGA User Guide，” https://www.xilinx.com/support/documentation/boards_and_kits/vc707/ug885_VC707_Eval_Bd.pdf，2017.1 閲覧.
[14] Xilinx Inc.，“Vivado Design Suite ユーザーガイド 入門，” https://japan.xilinx.com/support/documentation/sw_manuals_j/xilinx2015_4/ug910-vivado-getting-started.pdf，2017.1 閲覧.
[15] Google，Google Trends，“https://www.google.co.jp/trends/，” 2016.12 閲覧.