組込みデバイス向けリアルタイム時系列データ分類器の設計と実装

園田 勇介[1,a)]　久我 守弘[2,b)]　尼崎 太樹[2,c)]　飯田 全広[2,d)]　末吉 敏則[2,e)]

概要：センサベースのデバイスが大量にインターネットに接続されると予測されており，高い処理能力を持つ時系列データ分析技術の要求が高まっている．本研究では時系列データの分類に着目する．Dynamic Time Warping（DTW）は最もよく用いられる時系列データの類似度計測方法の 1 つであり，時系列データ分類の重要なサブルーチンである．DTW 計算の高速化の研究は数多くなされているが，センサネットワークの中継ノード上で DTW を用いたリアルタイム時系列分類を実現した例は少ない．本研究では Xilinx 社のプログラマブル SoC である Zynq に DTW を実装し，リアルタイム時系列データ分類器を構築した．評価により，ソフトウェアのみでの処理と比較して最大 204 倍の実行時間短縮が確認できた．

キーワード：時系列データ，分類，アクセラレータ，FPGA

Design and Implementation of Real-Time time series classification for Embedded Device

Sonoda Yusuke[1,a)]　Kuga Morihiro[2,b)]　Amagasaki Motoki[2,c)]　Iida Masahiro[2,d)]
Sueyoshi Toshinori[2,e)]

Abstract: It is predicted that tremendous amount of sensor-based devices will be connected to the Internet and there is an increasing demand for time series data analysis method with high processing capability. Dynamic Time Warping (DTW) is one of the most popular similarity measure method and it is an important subroutine in time series data classification. Related works have been proposed to speed up DTW, but there is few case of realizing real-time time series classification using DTW on the relay node. In this paper, We propose a real-time time series classification architecture on Xilinx programmable SoC Zynq. Proposed architecture show that the execution time was reduced by up to 1/204 times compared to software.

Keywords: Time series, Classification, Accelerator, FPGA

1. はじめに

Internet of Things（IoT）に代表される情報通信技術の発展により，膨大な量の組み込みデバイスがインターネットに接続されている．2020 年までには 500 億ものデバイスが接続されるという予想もされている [1]．それらのほとんどはセンサベースの組み込みデバイスであり，莫大な量のデータストリームがそれらから生成される．データストリームを活用する多くのアプリケーションがリアルタイムのデータマイニングを必要とする．そのため，センサから得られたデータをそのままサーバに送るのではなく，センサネットワークの中継ノード上で処理のすべて，もしくは一部をオフロードする手法が主流になると考えられる．

本研究では時系列データの分類に着目する．時系列デー

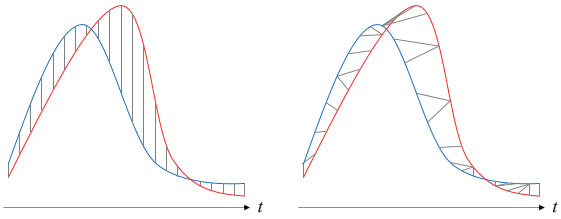
[1] 熊本大学 大学院自然科学研究科
2-39-1 Kurokami, Chuo, Kumamoto 860–8555, Japan
[2] 熊本大学大学院 先端科学研究部
2-39-1 Kurokami, Chuo, Kumamoto 860–8555, Japan
[a)] sonoda@arch.cs.kumamoto-u.ac.jp
[b)] kuga@cs.kumamoto-u.ac.jp
[c)] amagasaki@cs.kumamoto-u.ac.jp
[d)] iida@cs.kumamoto-u.ac.jp
[e)] sueyoshi@cs.kumamoto-u.ac.jp

タの分類の中で重要な処理として，時系列同士の距離計算があげられる．時系列同士の距離の計算方法は様々存在しているが，一般的な方法の 1 つとして Dynamic Time Warping（DTW）である．先行研究 [2] では様々な時系列データの距離計算方法を比較しており，DTW が最も良い方法であることを示している．しかし，DTW の時間計算量は大きく，膨大なデータを対象に計算する場合その計算時間が問題となる．

DTW のソフトウェアやハードウェアでの高速化の研究は数多く存在する．Lower Bound（LB）[3] と呼ばれる手法では DTW 計算の多くを削減できる．先行研究 [4] で提案された SPRING では DTW をインクリメンタルに計算できるが，時系列データの正規化ができないという問題がある．さらに時系列データの分類をリアルタイムに行うための手法として先行研究 [5] では分類のエニータイムアルゴリズム化が提案されている．エニータイムアルゴリズムとは，求める解の質が時間に対して単調増加になっており，途中で処理を中断しその時点での解を出力できるアルゴリズムの総称である．これにより継続的に時系列データが生成されるような環境での分類が可能である．これらのようにソフトウェアでの高速化が数多く提案されているが，それでも DTW 計算はアプリケーションの全体の処理時間の 80%を占めている [6]．そのため FPGA を用いたハードウェアによる高速化も提案されている．先行研究 [7] では，FPGA（Field Programmable Gate Array）に DTW 計算を実装することで高速な時系列データの類似度探索を実現している．ほかにも GPU を用いることで高速な DTW 計算を達成している例が存在する [8][9]．

本研究では，時系列データのリアルタイム分類をセンサ側の組み込みデバイスで行うことを想定する．センサベースの組み込みデバイスでは低い消費電力で高い処理能力を持つことが求められるが，本研究では Xilinx 社製プログラマブル SoC を利用する．Zynq はプロセッサを中心とする Processing System（PS）部とユーザが内部の回路構成を変更可能な Programmable Logic（PL） 部から構成される集積回路である．分類の上でボトルネックとなる DTW 計算を PL 部に実装することで高い処理性能を持つ時系列データの分類を達成する．さらにリアルタイムで分類を行うためには大量の時系列データの距離比較が必要になり，PS 部と PL 部のデータの転送が大きな時間のロスになると考えられる．そこで本研究では分類したい時系列を PL 部の BRAM（Block RAM）に保存し，複数の教師データとの比較の際に繰り返し転送しないようにすることで処理時間の短縮を行った．

本論文の構成は以下のとおりである．2 章では時系列データの分類の関連研究を説明する．3 章では提案アーキテクチャについて説明し，4 章ではその評価結果を示す．最後に 5 章で本研究のまとめを示す．



(a) ユークリッド　　(b) DTW

図 1: 時系列データ間の距離計算法

2. Dynamic Time Warping

本章では，DTW の計算方法を説明した後，それを用いた分類手法である K-Nearest Neighbor（KNN）について説明し，さらにそのエニータイムアルゴリズム化に関する先行研究について説明する．

2.1 DTW 計算手法

2 つのシーケンスを以下のように表す．

$$C = c_1, c_2, ..., c_n, \quad T = t_1, t_2, ..., t_m \tag{1}$$

n，m はそれぞれシーケンス C，T の長さである．図 1 にユークリッド距離と DTW 距離の計算方法を示す．ユークリッド距離計算では 2 つのシーケンスの時間軸順に対応する値の差を計算し，それを総和することで求める．しかし図 1 に示すような，2 つのシーケンスの形が似ているが少し時間軸上でずれているような場合，距離が大きくなり 2 つを似ているものとして検出できない．また，数式 (1) で定義されているような 2 つのシーケンスの長さが異なる場合，ユークリッド距離は計算できない．DTW は時系列データの距離を最小化するようにシーケンス長を調節する性質を持っているため，時間軸上でずれている場合でも似たものとして検出できる．さらにシーケンス長が異なる場合でも計算が可能である．

通常，シーケンスの正規化を行わなければシーケンス間の距離は有用ではないとされている [10]．長さ n のシーケンス $S = (s_1, s_2, ..., s_n)$ の正規化は以下の式で求められる．

$$\mu = \frac{1}{n}\sum_{n}^{k=1} s_k \tag{2}$$

$$\sigma^2 = \frac{1}{n}\sum_{k=1}^{n} s_k^2 - \mu^2 \tag{3}$$

$$s'_k = \frac{s_k - \mu}{\sigma} \tag{4}$$

以下より簡単のため，正規化されたシーケンスの要素 s'_k を s_k のように表す．

DTW 距離の計算は図 2 のような行列を用いて行われる．数式 (1) で定義された 2 つのシーケンス長の DTW 距離 $D(C,T)$ は以下のように定義される．

$$D(C,T) = f(n,m)$$

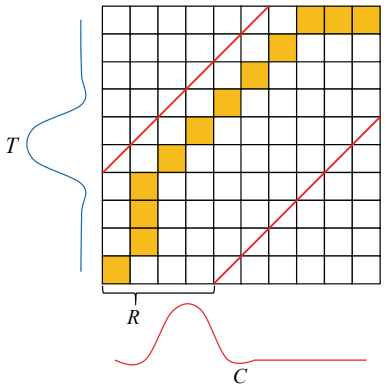


図 2: タイムワーピング行列

$$f(i,j) = |c_i - t_j| + min \begin{cases} f(i,j-1) \\ f(i-1,j) \\ f(i-1,j-1) \end{cases} \quad (5)$$

$$f(0,0) = 0, f(i,0) = f(0,j) = \inf$$

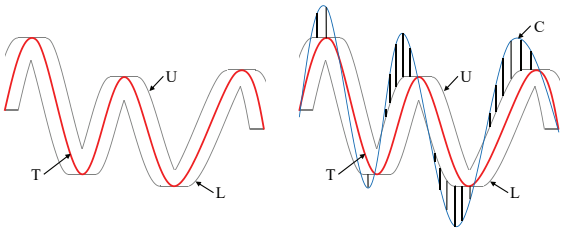
$$(i = 1,...,n; j = 1,...,m)$$

DTW 距離計算は図 2 に示すタイムワーピング行列の左下の要素から，シーケンス間の距離が最短となるような要素同士をマッチングして累積することで求める．この時たどっていった要素がなす経路をワーピングパスと呼ぶ．DTW 距離計算の時間計算量は $O(mn)$ である．

先行研究 [11] では Sakoe-Chiba band と呼ばれる高速化手法を提案している．図 2 に示すように，ワーピングパスを対角線上からある距離までの範囲に制限することで要素の計算を減らすことで高速化を達成している．さらに，片方のシーケンスの 1 つの点がもう片方のシーケンスのほとんどの点と対応している，などといったあまり有用でないマッチを避けることができる．制約 R はシーケンス全体の長さの割合で与えられ，0 から 100%で変化する．R で決定されるワーピングパスの範囲をワーピングウィンドウと呼ぶ．Sakoe-Chiba band は様々なアプリケーションで効果的である．最適な R の値はアプリケーションによって異なる．

さらに Lower Bounds（LB）と呼ばれる高速化手法がある [3]．これはある閾値以上の正確な DTW 距離に関心がない場合に有効である．LB 距離は DTW 距離と比較して時間計算量が小さく，さらに LB 距離は DTW 距離よりも小さいことが保証されている．そのため，LB 距離が閾値を上回る場合，DTW 距離も必ず閾値を上回ることがわかるため，DTW 距離計算を省略することができる．数式 (1) で定義する 2 つの時系列間の LB 距離 $LB_{Keogh}(C,T)$ は以下のように求められる．

$$U_i = \max(t_{i-R} : t_{i+R}), \quad L_i = \min(t_{i-R} : t_{i+R})$$



(a) (b)

図 3: Lower Bound(LB) 距離

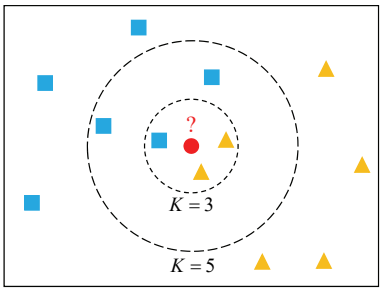


図 4: k-Nearest Neighbor 法

$$LB_{Keogh}(C,T) = \sum_{i=1}^{m} \begin{cases} c_i - U_i & \text{if } c_i > U_i \\ L_i - c_i & \text{if } L_i > c_i \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

ここで R はワーピングウィンドウ制約である．図 3(a) のように，あらかじめシーケンス T からシーケンス U，L を生成しておく．シーケンス C が与えられたら，図 3(b) のように U，L と比較して累積することで LB 距離を求める．時間計算量は $O(n)$ であり，DTW 距離よりも高速に計算することができる．

2.2 K-Nearest Neighbor 法

KNN 法は最も盛んに研究されているパターン分類手法の 1 つである．図 4 に KNN の分類手法を示す．KNN 法では分類したいデータと，あらかじめ分類された教師データとの距離を計測し，教師データの中から最も近い K 個のデータを探索する．その K 個のデータの中で多数決を取り，最も多い分類クラスに分類する手法である．最適な K の値はアプリケーションにより異なる．

2.3 Anytime Classification アルゴリズム

KNN 法をリアルタイムで行うためには，計算の途中でも中断可能で，ある程度の分類の正確性を持つ結果を出力できるように改良する必要がある．先行研究 [5] では KNN 法のエニータイムアルゴリズム化を提案している．図 5 に示すように，エニータイムアルゴリズムは計算時間と解の品質がトレードオフになっているアルゴリズムである．次のデータが継続的に生成される環境の場合，現在のデータの処理を中断，結果を出力し，次のデータの処理を開始す

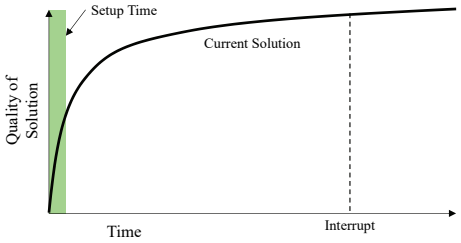


図 5: Anytime Classification

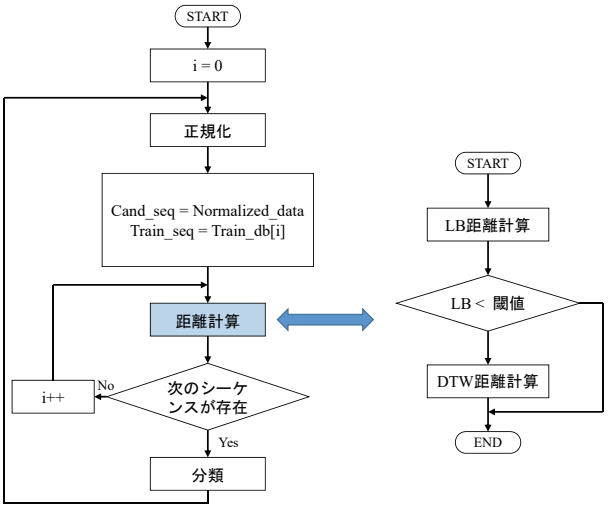


図 6: 分類のフロー

ることでリアルタイムで処理を行うことができる．分類の場合，解の品質とは分類の正答率である．先行研究 [5] では KNN 法で比較する教師データの並び替えをすることで分類のエニータイムアルゴリズム化を実現している．具体的には，それぞれのデータごとに分類の貢献度を計算し，最も悪いものを最後列に並べるのを繰り返していくことで並び替えていく．貢献度は以下の式で求める．

$$rank(x) = \sum_j \begin{cases} 1 & \text{if class(x) = class(x}_j\text{)} \\ -2/(num\_of\_class - 1) & \text{otherwise} \end{cases} \quad (7)$$

x_j は x を最も距離の近いデータとするデータである．最も悪い貢献度を持つものが複数現れた場合はそれらの中で同じように貢献度を計算して 1 つに絞っていく．

3. 提案アーキテクチャ

本章では実際に実装した時系列データ分類のアーキテクチャについて述べる．実装には Xilinx 社のプログラマブル SoC である Zynq を用いた [12]．

3.1 ハードウェアアーキテクチャ

図 6 にシーケンス分類のフローチャートを示す．センサは継続的にデータを発行し，分類が可能なほどデータが蓄積されたら分類を開始する．最初にシーケンスの正規化を行った後，正規化されたシーケンスと教師データのシーケンスの距離計算を行う．教師データは複数あるが，2.3 節で説明したように並び替えされており，途中で中断してもそれまでの最善の結果が得られるようになっている．距離計算後，もしセンサから発行されたデータが次のシーケンスとして十分なほど蓄積されていたらそこで教師データとの距離計算を中断し，現在のシーケンスの分類結果を出力する．そうでなければ分類を継続し，現在のシーケンスと次の教師データのシーケンスとの距離計算を行う．

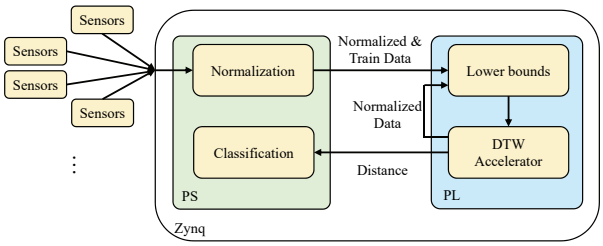


図 7: ハードウェア全体のブロック図

KNN 法ではすべてのデータとの正確な距離を計測する必要はなく，最も近い K 個のデータがわかればよい．そのため，距離計算では K 個目に近いデータとの距離を閾値とし，LB 距離の後に DTW 距離を計算する．LB 距離が閾値以上の場合，DTW 距離の計算をせずに距離計算を終了する．

本研究では距離計算の部分をアクセラレータとして Zynq 内の PL 部に実装する．図 7 にハードウェア全体のブロック図を示す．センサから生成されたデータはソフトウェアにより正規化され，正規化されたシーケンスと最初の教師データがハードウェアに転送される．最初に LB 距離が計算され，LB 距離が閾値を下回った場合，さらに DTW 距離を計算する．さらに DTW 距離が閾値を下回った場合，その距離がソフトウェアに通知され，閾値が更新される．もし LB 距離または DTW 距離が閾値を上回った場合，その結果がソフトウェアに通知される．その時，センサから発行されたデータが十分に蓄積されていたらソフトウェアでその時点での分類結果を出力する．そうでない場合，ソフトウェアから次の教師データのシーケンスが送られる．分類したいシーケンスはハードウェアで再度 LB 距離計算モジュールに送られ，さらに距離計算を行う．

3.2 距離計算アクセラレータのブロック図

図 8 に距離計算アクセラレータのブロック図を示す．PS 部と PL 部のシーケンスデータの転送は AXI4 プロトコルで転送される．距離計算アクセラレータでは LB 距離と DTW 距離を実装する．さらに分類したいシーケンスを BRAM に保存することで，複数の教師データと比較する際に繰り返しの転送を避け，処理時間の短縮が可能になる．LB 距離計算モジュールは対応する教師データから生成されたシーケンス U，L と分類したいシーケンス C を入力とし，対応する点同士を比較して累積していく．累積してい

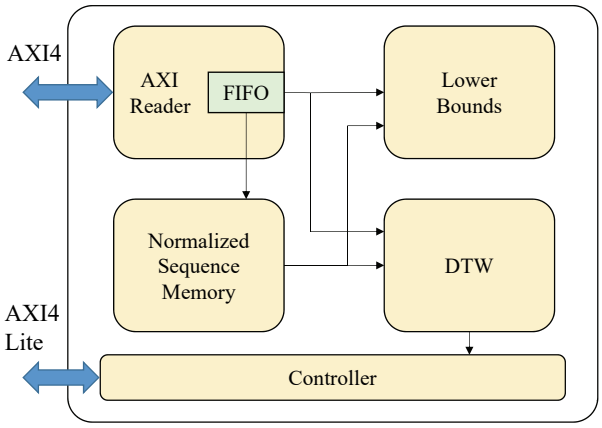


図 8: 提案システムのアーキテクチャ

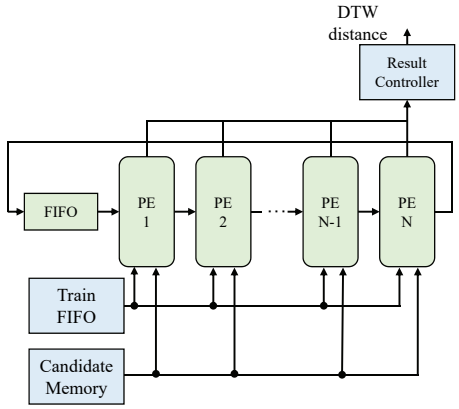


図 9: PE ring

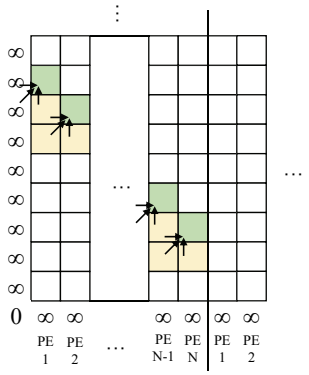


図 10: タイムワーピング行列のデータ依存性

る途中で閾値を上回った場合，その時点で LB 距離計算を中断し，その結果を通知する．LB 距離計算が終了した時点で閾値を下回っていた場合，DTW 距離計算モジュールで DTW 距離を計算する．計算した結果は AXI4-Lite を用いて PS 部に転送する．

DTW 距離計算モジュールのアーキテクチャを図 9 に示す．この構造は先行研究 [7] を参考にしている．Processing Element（PE）がリング状になった構造になっている．図 10 にタイムワーピング行列計算のデータ依存性を示す．行列内の要素の計算の際に依存しているのはその要素の周辺の 3 つの要素のみである．そのため，PE を十分な数だけ実装すればクロックサイクルレベルのパイプラインが可能である．しかし，シーケンス長が変更されるたびに PE 数を変更し PL 部を再構成しなければならず，柔軟性がない．さらに組込みデバイスなどに用いられる SoC は使用できるハードウェアリソースが少なく，実装できる PE の数が制限されるという問題もある．そのため，PE の数をシーケンス長ではなく固定にし，リング状にして PE を使いまわすことで柔軟性を持たせる．N 個の PE はパイプライン的に並列に計算され，N 個目の計算結果は FIFO に保存される．1 個目の PE の計算が終わった時点で FIFO に保存されている計算結果を読出し，続いて計算を開始する．最後の列まで計算が終わったら DTW 距離をコントローラが読み出して出力する．シーケンス長を N，PE の数を W としたとき，DTW 距離計算の実行サイクル数は $\lceil N/W \rceil * N$ である．

4. 評価

4.1 評価環境

使用したボードは Xilinx 社製 Zynq-7000 XC7Z010-1CLG400C を搭載する Digilent 社製の ZYBO[13] である．アクセラレータの開発には，Xilinx 社が提供している Vivado 2016.4[14] を使用した．ARM の動作周波数は 650MHz，FPGA 部の動作周波数は実装結果から 50MHz に設定した．なお実装した回路は PE の数を 48 に設定している．実装した際の FPGA 上のリソース利用量を表 1 に示す．

表 1: リソース利用量

リソース	使用数 [個]	利用可能数 [個]	使用率 [%]
LUT	11,561	17,600	65.69
LUTRAM	409	6,000	6.82
FF	9,193	35,200	26.12
BRAM	19	60	31.67

4.2 実行時間

ARM のみでの動作と提案した ARM+FPGA の動作の実行時間を比較した．データセットには UCR time series classification archive[15] のページからダウンロードしたものを用いた．このデータセットはデータマイニングや機械学習のコミュニティから提供された現実世界のデータから成っている．実行時間データの概要を表 2 に示す．

表 2: データセット概要

データセット名	教師データ数	テストデータ数	データ長
Beef	30	30	470
Synthetic Control	300	300	60
CBF	30	900	128

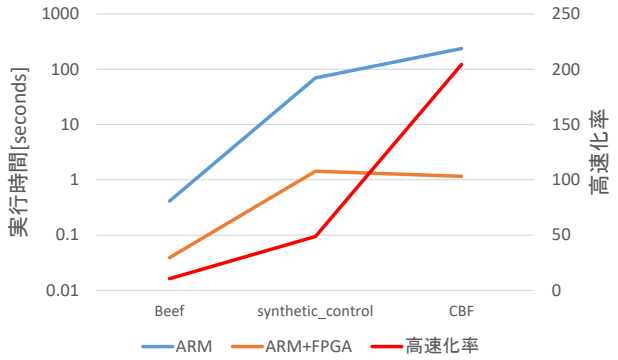


図 11: 実行時間

実行時間を図 11 に示す．3 つのデータセットすべてで提案システムでの高速化が見られた．高速化率は Beef，synthetic control，CBF でそれぞれ約 10 倍，49 倍，204 倍の高速化が確認できた．データセットごとに高速化率が大きく異なるのは，データセットごとにテストデータ数が異なり，テストデータ数が多いほど PL 部への転送時間が削減され，ARM のみの実行時間と差がついたためと考えられる．

次に分類器のスループットを計算する．ここで，N をシーケンス長，C をテストデータ数，T を教師データ数とする．1 つのテストデータを分類するのに転送するデータは，そのテストデータとそれと比較する教師データである．そのため，すべてのデータの転送量は $(N + N * T) * C$ である．これを各データセットの実行時間で割ると，Beef は 11,206,081 ワード/s，Synthetic control は 26,936,485 ワード/s，CBF は 3,084,992 ワード/s だった．

5. まとめ

本研究では，センサベースの組み込みデバイス向けのリアルタイム時系列データの分類を実現するシステムを提案した．実装には Xilinx 社のプログラマブル SoC である Zynq を使用し，PL 部の BRAM に分類したいシーケンスを保存し転送時間を省略することで短い実行時間を達成した．評価の結果，ソフトウェアのみと比較して最大 204 倍の実行時間の短縮を確認できた．現在はシーケンスデータの転送が完了するまでアクセラレータの動作が停止しており，完全なストリーミング処理になっていない．ストリーミング処理で実装を行うことにより更なる実行時間の短縮が可能であると考えている．

参考文献

[1] Dave Evans. “The Internet of Thigs: How the Next Evolution of the Internet Is Changing Everything,” CISCO white paper, 2011.
[2] Ding, H., Trajcevski, G., Scheuermann, P., Wang, X. and Keogh, E. “Querying and mining of time series data: experimental comparison of representations and distance measures,” PVLDB Endowment, Vol.1, No.2, pp.1542-1552 (2008).
[3] Keogh, E. J., Wei, L., Xi, X., Vlachos, M., Lee, S. H. and Protopapas, P. “Supporting exact indexing of arbitrarily rotated shapes and periodic time series under Euclidean and warping distance measures,” VLDB, Vol.18, No3, pp.611-630 (2009).
[4] Sakurai, Y., Faloutsos, C. and Yamamoto, M. “Stream monitoring under the time warping distance,” Proc. ICDE, pp1046-1055 (2007).
[5] Ueno, K., Xi, X., Keogh, E. and Lee, D. “Anytime classification using the nearest neighbor algorithm with applications to stream mining,” Proc. ICDM, pp.623-632 (2006).
[6] Zhang, Y., Adl, K. and Glass, J. “Fast spoken query detection using lower-bound Dynamic Time Warping on Graphical Processing Units,” Proc. ICASSP, pp.5173-5176 (2012).
[7] Wang, Z., Huang, S., Wang, L., Li, H., Wang, Y. and Yang, H. “Accelerating subsequence similarity search based on Dynamic time Warping distance with FPGA,” Proc. FPGA, pp.53-62 (2012).
[8] Sart, D., Mueen, A., Najjar, W., Keogh, E. and Nienattrakul. “Accelerating Dynamic Time Warping subsequence search with GPUs and FPGAs,” Proc. ICDM, pp.1001-1006 (2010).
[9] Hundt, C., Schmidt, B. and Schomer, E. “Cuda-accelerated alignment of subsequences in streamed time series data,” Proc. ICPP, pp.10-19 (2014).
[10] Rakthanmanon, T., Campana, B., Mueen, A., Batista, G., Westover, B., Zhu, Q., Zakaria, J. and Keogh, E. “Searching and mining trillions of time series subsequences under Dynamic Time Warping,” Proc. KDD, pp.262-270 (2012).
[11] Keogh, E. “Exact indexing of Dynamic Time Warping,” Proc. VLDB, pp.406-417 (2002).
[12] Xilinx Inc. “Zynq - 7000 All Programmable SoC 概要，” 入手先 <https://japan.xilinx.com/support/documentation/data_sheets/j_ds190-Zynq-7000-Overview.pdf>(参照 2017-1-19).
[13] Digilent Inc. “ZYBO FPGA Board Reference Manual,” available from<https://reference.digilentinc.com/_media/zybo:zybo_rm.pdf>(accessed 2017-1-19).
[14] Xilinx Inc. “Vivado Design Suite ユーザー ガイド，” 入手先 <https://japan.xilinx.com/support/documentation/sw_manuals_j/xilinx2014_1/ug910-vivado-getting-started.pdf>(参照 2017-1-19).
[15] Chen, Y., Keogh, E., Hu, B., Begum, N., Bagnall, A., Mueen, A. and Batista, G. “UCR Time Series Classification Archive,” available from<http://www.cs.ucr.edu/~eamonn/time_series_data/>(accessed 2016-12-16).