A Reconfigurable Parallel Acceleration Platform for Evaluation of Permutation Entropy*

Xiaowei Ren[1,2], Pengju Ren[1], Badong Chen[1], Jose C. Principe[3] and Nanning Zheng[1]

***Abstract*— In recent years, permutation entropy is widely used to characterize the complexity of EEG time series and can be applied to predict the onset of serious brain diseases, such as the epileptic seizure. In many practical situations, the number of EEG time series that need to be analyzed simultaneously is very large, so the computation of the permutation entropy is time-consuming and should be accelerated so that the real-time analysis is possible. Noting that mathematical operations can be sped up effectively with hardware implementation, we design a parallel FPGA platform consisting of 128 reconfigurable pipelines, which are used to calculate the permutation entropy for a single EEG time series. When the platform works at 150MHz and the embedding dimension is 5, an average speedup of 5553 for different window sizes is achieved compared with C codes running on a 3GHz Intel(R) Core(TM) i5-2320 CPU. Meanwhile, the hardware cost is very low.**

I. INTRODUCTION

Nowadays, a growing number of people suffer from serious brain diseases. However, scientists fortunately find that the complexities of EEG signals of patients start to change in advance of the clinical diagnosis. That's to say, if we can record transitions of EEG signals earlier, we could adopt some measures to prevent these diseases from happening. Then, creating a good measure to characterize the complexity of EEG signal is crucial.

During the last two decades, lots of methods have been proposed to measure the complexity of EEG signal, such as Kolmogorov entropy [1], Lyapunov exponents [2], permutation entropy [3] and weighted-permutation entropy [4] etc. Among these measures, the permutation entropy is widely used because of its simplicity and robustness. Benefiting from its advantages, permutation entropy has a great potential to be used in real-time applications. Lots of studies about the complexity analysis of patients' EEG signals with permutation entropy have been conducted [5] [6].

In physical applications, we usually need to analyze EEG data in real time. However, current instruments used to record EEG signals usually have 64 channels, 128 channels or even more. Therefore, if you want to apply permutation entropy to real-time applications, you have to complete all the analysis of these EEG signals in real time simultaneously. Even though permutation entropy is simple and easy to compute, it's impossible to complete all these calculations with software timely. Note the fact that mathematical operations could be accelerated in orders of magnitude by hardware implementation [7], we explore the permutation entropy acceleration through hardware platform, instead of software implementation, so as to satisfy the real-time demand. Meanwhile, EEG data recording of patients is a long-term process. This requirement also drives us to design a simple wearable device which could collect EEG data without disturbance for our daily lives [8].

In this paper, we design a 128-way reconfigurable parallel hardware acceleration platform for the real-time calculation of permutation entropy with FPGA. When it works at 150MHz and the embedding dimension is 5, an average speedup of 5553 for different window sizes is achieved versus C implementation. The paper is organized as follows. A brief description of permutation entropy is shown in section II. Section III elaborates the architecture of hardware design. In section IV, performance evaluation and implementation results are shown. At last, section V concludes our work.

II. PERMUTATION ENTROPY

Considering the time series $\{x_t\}_{t=1}^{T}$ and its time-delayed embedding representation $X_t^{n,\tau} = \{x_t, x_{t+\tau}, ..., x_{t+(n\ \ 1)\tau}\}$ for $t = 1, 2, ..., T-(n-1)\tau$, where n and τ denote the embedding dimension and time delay respectively. In order to simplify the hardware design, we set the time delay to 1. Then, time series $\{x_t\}_{t=1}^{T}$ has $T-(n-1)$ subvectors expressed as $X_t^n = \{x_t, x_{t+1}, ..., x_{t+n\ \ 1}\}$. To compute permutation entropy, every subvector is assigned a single permutation type out of $n!$ possible ones (representing all unique orderings of n different real numbers). Then the permutation entropy of embedding dimension $n \geq 2$ could be calculated with the following formula:

$$H(n) = -\sum_{i=1}^{n!} p(\pi_i) \log p(\pi_i) \qquad (1)$$

*This work is partially funded by NSFC grant No.61372152, No.610303036 and No.61231018, China Postdoctoral Science Foundation No.2012M521777, Specialized Research Fund for the Doctoral Program of Higher Education of China No.20130201120024, Natural Science Basic Research Plan in Shaanxi Province of China No.2013JQ8029, the Fundamental Research Funds for the Central Universities, the program of introducing talents of discipline to university and the Open Project Program of the State Key Laboratory of Mathematical Engineering and Advanced Computing.

[1,2]Xiaowei Ren is with the Institute of Artificial Intelligence and Robotics, Xi'an Jiaotong University, Xi'an 710049, China and the State Key Laboratory of Mathematical Engineering and Advanced Computing, Wuxi 214125, China (email: renxiaowei66@gmail.com).

[1]Pengju Ren, Badong Chen and Nanning Zheng are with the Institute of Artificial Intelligence and Robotics, Xi'an Jiaotong University, Xi'an 710049, China (email: pengjuren@gmail.com, {chenbd, nnzheng}@mail.xjtu.edu.cn).

[3]Jose C. Principe is with the Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL 32611 USA (email: principe@cnel.ufl.edu).

978-1-4244-7929-0/14/$26.00 ©2014 IEEE

$$H(n,i+1) = H(n,i) - \frac{\|\pi_{increase}\|}{T-n+1}\log\frac{\|\pi_{increase}\|}{T-n+1} + \frac{\|\pi_{increase}\|+1}{T-n+1}\log\frac{\|\pi_{increase}\|+1}{T-n+1} - \frac{\|\pi_{decrease}\|}{T-n+1}\log\frac{\|\pi_{decrease}\|}{T-n+1} + \frac{\|\pi_{decrease}\|-1}{T-n+1}\log\frac{\|\pi_{decrease}\|-1}{T-n+1} \quad (3)$$

$$H(n,0) = H(n,0) - \frac{\|\pi_{increase}\|}{T-n+1}\log\frac{\|\pi_{increase}\|}{T-n+1} + \frac{\|\pi_{increase}\|+1}{T-n+1}\log\frac{\|\pi_{increase}\|+1}{T-n+1} \quad (4)$$

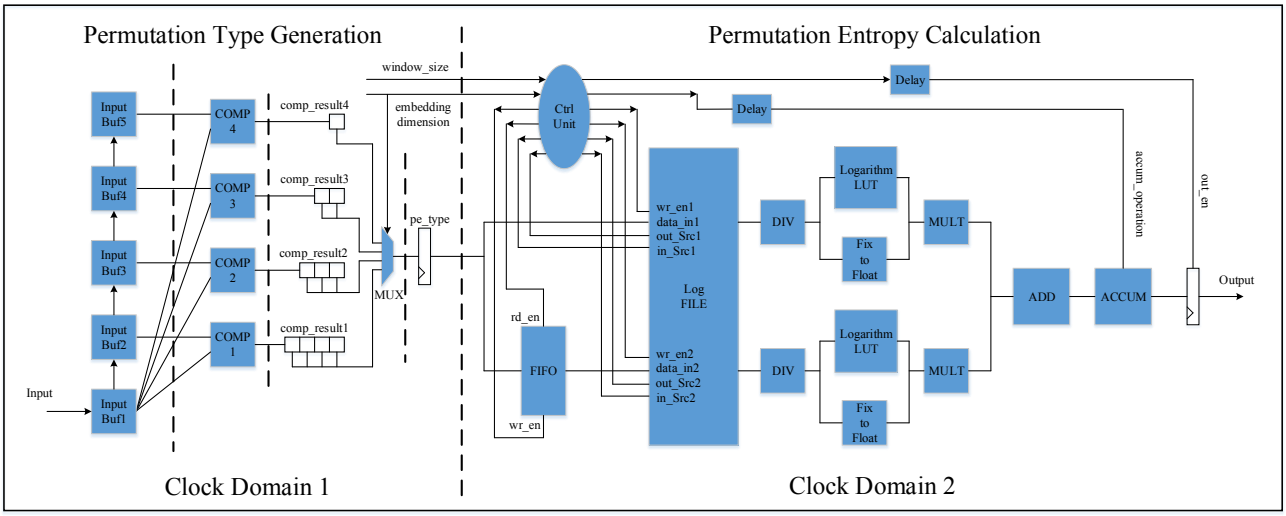

Fig. 1. The Microarchitecture of Pipeline

Where log is with base 2, π_i is one of the $n!$ permutation types $\{\pi_i\}_{i=0}^{n!}$ and $p(\pi_i)$ is defined as:

$$p(\pi_i) = \frac{\|\{t|t \le T-n+1, type(X_t^n) = \pi_i\}\|}{T-n+1} \quad (2)$$

Where $\|.\|$ denotes the cardinality of a set. So permutation entropy is the information contained in comparing n consecutive values of the time series and it assumes values in the range of $[0, \log n!]$.

III. HARDWARE IMPLEMENTATION

The microarchitecture of pipeline used to compute the permutation entropy of one-channel EEG signal composes of two parts: permutation type generation and permutation entropy calculation, as shown in Fig.1. They are operated at two different clock rates and the frequency of clock domain 2 is twice as large as clock domain 1, the reason is elaborated in the last paragraph of subsection III.B. Once a sample datum comes, a new permutation type is generated. Then, permutation type is passed into the latter part of this pipeline to update the result of permutation entropy.

A. Permutation Type Generation

The hardware implementation used to calculate the permutation type is reconfigurable so that it could support different embedding dimensions. In our design, the embedding dimension could be 2, 3, 4 or 5 as required. Fig.1 shows that the generation of a permutation type should go through four pipeline stages. Firstly, we need to shift values of all input buffers to their corresponding upper ones to record and update the data sequence. Meanwhile, the new sample datum is buffered in Input Buf1. Then data from Input Buf2 to Buf5 are compared with the new input datum simultaneously. In the third stage, there are four shift registers storing the comparison results. These four registers shift one bit to the right after every comparison operation. At last, the fourth stage outputs the result of permutation type according to the configuration value of embedding dimension.

Let's take a look at how the output is reconfigured in the light of embedding dimension. Assuming a_i is the new coming datum stored in Buf1, then data of Input Buf2 to Buf5 are respectively $a_{i\ 1}$, $a_{i\ 2}$, $a_{i\ 3}$ and $a_{i\ 4}$. If we define $c_{m,n}$ as the comparison result between a_m and a_n, values in the comp_result1 register are $[c_{i\ 1,i}, c_{i\ 2,i\ 1}, c_{i\ 3,i\ 2}, c_{i\ 4,i\ 3}]$ and $[c_{i\ 2,i}, c_{i\ 3,i\ 1}, c_{i\ 4,i\ 2}]$, $[c_{i\ 3,i}, c_{i\ 4,i\ 1}]$, $[c_{i\ 4,i}]$ are values in comp_result2, comp_result2 and comp_result3. The setting value of embedding dimension will determine which element of these four comparison-result registers could go through the multiplexer. When embedding dimensions are 2, 3, 4 and 5, the corresponding permutation type results are $[c_{i\ 1,i}, 0, 0, 0, 0, 0, 0, 0, 0, 0]$, $[c_{i\ 1,i}, c_{i\ 2,i\ 1}, 0, 0, c_{i\ 2,i}, 0, 0, 0, 0, 0]$, $[c_{i\ 1,i}, c_{i\ 2,i\ 1}, c_{i\ 3,i\ 2}, 0, c_{i\ 2,i}, c_{i\ 3,i\ 1}, 0, c_{i\ 3,i}, 0, 0]$ and $[c_{i\ 1,i}, c_{i\ 2,i\ 1}, c_{i\ 3,i\ 2}, c_{i\ 4,i\ 3}, c_{i\ 2,i}, c_{i\ 3,i\ 1}, c_{i\ 4,i\ 2}, c_{i\ 3,i}, c_{i\ 4,i\ 1}, c_{i\ 4,i}]$ respectively. At last, the pipeline implementation and four parallel comparators guarantee that the throughput of the permutation type generation unit is one result per cycle without any stalls.

B. Permutation Entropy Calculation

Once we get a permutation type, it can be used to update the value of permutation entropy. Considering an EEG time series $\{x_1, x_2, ..., x_i, x_{i+1}, ...\}$, it is analyzed using a sliding window contains T samples and the embedding dimension is n, where $n \leq T$. So the i_{th} window contains sample data $\{x_i, x_{i+1}, ..., x_{i+T\ \ 1}\}$ and $\{x_{i+1}, x_{i+2}, ..., x_{i+T}\}$ is the sample data of the $(i+1)_{th}$ window. It's obvious that only x_i and x_{i+T} can lead to the difference of permutation entropy between the i_{th} window and the $(i+1)_{th}$ window, because x_i makes the occurrence number of permutation type of subvector X_i^n decrease by 1 and x_{i+T} increases the occurrence number of permutation type of subvector $X^n_{i+T\ \ (n\ \ 1)}$ by 1. We define the permutation type of X_i^n and $X^n_{i+T\ \ (n\ \ 1)}$ as $\pi_{decrease}$ and $\pi_{increase}$. If $\pi_{increase}$ and $\pi_{decrease}$ are different, the permutation entropy of $(i+1)_{th}$ window could be updated by formula (3). Otherwise, $H(n)$ remain unchanged. Furthermore, there is no $\pi_{decrease}$ in the first window, so the permutation entropy of the first window should be calculated by formula (4).

The latter part of Fig.1 shows the hardware structure which could complete the update of permutation entropy as described above. The control unit is used to schedule different steps of update. All control signals are generated according to the configured values of embedding dimension and window size. The FIFO is used to record $\pi_{decrease}$ and the occurrence number of every permutation type is logged in the LogFILE. Four fractions of formula (3) are all calculated with fix-point divider DIV. Fix-to-float unit transforms the fix-point results of divider into float-point format. In order to save hardware cost, the logarithm function defined in $[0, 1]$ is quantized to a finite LUT (look-up table). Although the quantized (2048 values) logarithm approximation can lead to some error, experimental results of subsection IV.A show that the calculation error is negligible. The first product item of formula (3) adds with the third one with an adder, so do the second item and the fourth one. Accumulator performs addition and subjection alternatively so that the permutation entropy is updated as formula (3).

As shown in formula (3), two product terms need to be calculated for each $\pi_{decrease}$ and $\pi_{increase}$, so the clock of permutation entropy calculation unit should have twice higher frequency than permutation type generation unit. Meanwhile, the adoption of a dual-port LogFILE enables operations of $\pi_{decrease}$ and $\pi_{increase}$ perform in parallel, accelerating the computation rate.

C. Structure of LogFILE

In our design, LogFILE is a very important component, because it records the occurrence number of every permutation type in real time during the analysis of EEG time series and its excellent hardware structure makes the parallel operations of $\pi_{decrease}$ and $\pi_{increase}$ possible. Fig.2 shows the detailed structure of LogFILE. The storage body is a read-first SRAM which has two input ports and two output ports. $\pi_{increase}$ and $\pi_{decrease}$ are connected to addr1 and addr2 respectively. Therefore, data_in1 and data_out1 are responsible for the operations needed by $\pi_{increase}$. Similarly, data_in2 and data_out2 belong to $\pi_{decrease}$.

Because clock domain 2 is twice faster than clock domain 1, the values of $\pi_{increase}$ and $\pi_{decrease}$ are kept constant during two clock cycles. In the first cycle, data_out1 and data_out2 are assigned by the values of SRAM outputs, namely $\|\pi_{increase}\|$ and $\|\pi_{decrease}\|$. At the same time, $\|\pi_{increase}\|+1$ and $\|\pi_{decrease}\|-1$ are assigned to data_in1 and data_in2. During the second clock cycle, $\|\pi_{increase}\|+1$ and $\|\pi_{decrease}\|-1$ are outputted thorough data_out1 and data_out2. Meanwhile, $\|\pi_{increase}\|+1$ and $\|\pi_{decrease}\|-1$ are written into SRAM. However, if $\pi_{increase}$ is equivalent to $\pi_{decrease}$, it's unnecessary to perform the above operations because the state of SRAM does not change any after these effort. Therefore, we can keep SRAM unchanged and output $\|\pi_{increase}\|$ and $\|\pi_{decrease}\|$ directly in two clock cycles when $\pi_{increase}$ and $\pi_{decrease}$ are identical. At last, we have to pay more attention to the first window where operations of $\pi_{decrease}$ don't exist. Hence, wr_en2, data_in2 and data_out2 should all keep "0" in the first window, but operations of $\pi_{increase}$ are the same as above description.

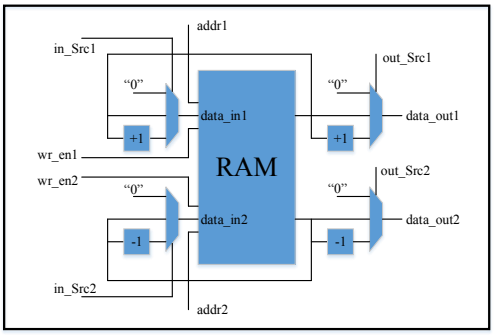

Fig. 2. Hardware Structure of LogFILE

IV. EVALUATION

In physical applications, we usually need to collect EEG data from 64, 128 channels or even more. Therefore, we should compute permutation entropies for all of these EEG time series simultaneously. Given this, by simply instantiate the pipeline of Fig.1 128 times, we implement a 128-way parallel hardware acceleration platform with FPGA. Through parallel computing, we could get the permutation entropies of 128 EEG time series in real time simultaneously. The FPGA platform we used is Xilinx Virtex-7 XC7V2000T, the state-of-the-art FPGA device.

A. Accuracy of Hardware Design

Given the quantized logarithm LUT could bring some computation error, the accuracy of our hardware platform is testified firstly. We use our FPGA platform to calculate permutation entropies for 128-way EEG time series that come from the Computational NeuroEngineering Laboratory of UFL. Each EEG time series contains 4200 sample data which are collected when people are required to watch a face picture. We compare our results with C implementation. The maximum error and average error for different window sizes and embedding dimensions are recorded in table I. It

shows that nearly every average error is less than 0.15%. Furthermore, all of the maximum errors are also so small that they are negligible. Therefore, we can definitively conclude that all permutation entropy results are calculated precisely with our parallel FPGA platform.

TABLE I

CALCULATION ERROR OF FPGA PLATFORM

Window size	Embedding dimension	Maximum error(%)	Average error(%)
256	3	0.150	0.070
	4	0.229	0.147
	5	0.349	0.212
512	3	0.115	0.070
	4	0.174	0.116
	5	0.211	0.158
1024	3	0.057	0.034
	4	0.097	0.058
	5	0.105	0.078

B. Speedup Analysis

Our FPGA platform is designed to accelerate the computation of permutation entropy so that it could be applied in real-time applications. Therefore, we test its speedup versus C codes. In the experiment, the FPGA platform works at 150MHz, namely that the clock domain 1 is 150MHz. No matter what the values of window size and embedding dimension are set to, the FPGA platform accepts one sample datum per clock cycle, so the execution time of FPGA is a constant. As for the C implementation, we also use formula (3) and (4) to compute the permutation entropy. Because the number of sample data is fixed at 4200, with the increase of window size, formula (4) is computed more times and less times for formula (3). What's more, the amount of computation in formula (4) is less than formula (3). Therefore, the bigger the window size, the shorter the execution time for C implementation. The C code runs on a desktop which is configured with 4GB main memory and a 3GHz Intel(R) Core(TM) i5-2320 CPU. During the experiment, window size and embedding dimension are set to different values. Table II shows that when embedding dimension is 3, 4 and 5 respectively, corresponding average speedup for different window sizes are 4555, 4965 and 5553. So the computation rate is sped up considerably.

TABLE II

SPEEDUP VERSUS C LANGUAGE

Window size	Embedding dimension	FPGA (μs)	C (ms)	Speedup
256	3	28.203	140.947	4998
	4	28.203	151.994	5389
	5	28.203	165.935	5884
512	3	28.203	132.090	4684
	4	28.203	142.492	5052
	5	28.203	156.014	5532
1024	3	28.203	112.316	3982
	4	28.203	125.584	4453
	5	28.203	147.830	5242

C. Implementation Result

The parallel FPGA platform is implemented in Verilog and synthesized with Xilinx EDA tool named Vivado. Hardware resource cost is summarized in table III. All functional units perform operations in IEEE single-precision format, and 47.41% DSP IPs is needed. 49.54% BRAM (Block RAM) is used to implement the FIFO, LogFILE and LUT of logarithm function. As for the FF (Flip-flop) and LUT, two main resources of FPGA, the utilization rate are only 17.88% and 25.21% respectively. Therefore, we can claim that the FPGA platform is implemented at a very low hardware cost.

TABLE III

IMPLEMENTATION RESULT OF FPGA PLATFORM

Resources	Utilized	Available	Utilization Rate(%)
FF	436736	2443200	17.88
LUT	307968	1221600	25.21
Memory LUT	1280	344800	0.37
I/O	67	1200	5.58
BRAM	640	1292	49.54
DSP48	1024	2160	47.41
BUFG	3	128	2.34
MMCM	1	24	4.17

V. CONCLUSIONS

In this paper, a hardware acceleration platform which could complete the real-time calculation of permutation entropies for 128 different EEG time series in parallel is achieved at a very low hardware cost. It consists of 128 pipeline circuits, which are used to compute the permutation entropy for a single EEG channel. The pipeline is reconfigurable so as to support different window sizes and embedding dimensions as demanded. Compared with C implementation, the FPGA platform which works at 150MHz is 4555, 4965 and 5553 times faster for different window sizes when the embedding dimension is 3, 4 and 5 respectively.

ACKNOWLEDGMENT

We thank all members of our team for their precious discussion and we express sincere appreciation to Bilal Fadlallah of UFL for his provision of EEG data.

REFERENCES

[1] W. v. Drongelen, S. Nayak, D. M. Frim, M. H. Kohrman, V. L. Towle, H. C. Lee, A. B. McGee, M. S. Chico, and K. E. Hecox, "Seizure anticipation in pediatric epilepsy: use of kolmogorov entropy," *Pediatric neurology*, vol. 29, no. 3, pp. 207–213, 2003.

[2] J. Gao and Z. Zheng, "Local exponential divergence plot and optimal embedding of a chaotic time series," *Physics Letters A*, vol. 181, no. 2, pp. 153–158, 1993.

[3] C. Bandt and B. Pompe, "Permutation entropy: a natural complexity measure for time series," *Physical Review Letters*, vol. 88, no. 17, p. 174102, 2002.

[4] B. Fadlallah, B. Chen, A. Keil, and J. Príncipe, "Weighted-permutation entropy: A complexity measure for time series incorporating amplitude information," *Physical Review E*, vol. 87, no. 2, p. 022911, 2013.

[5] D. Labate, I. Palamara, G. Occhiuto, G. Morabito, F. La Foresta, and F. Morabito, "Complexity analysis of alzheimer disease eeg data through multiscale permutation entropy," *public health*, vol. 1, p. 2, 2012.

[6] X. Li, G. Ouyang, and D. A. Richards, "Predictability analysis of absence seizures with permutation entropy," *Epilepsy research*, vol. 77, no. 1, pp. 70–74, 2007.

[7] H. Shojania and B. Li, "Parallelized progressive network coding with hardware acceleration," in *Quality of Service, 2007 Fifteenth IEEE International Workshop on*. IEEE, 2007, pp. 47–55.

[8] G. Danese, F. Leporati, A. Majani, G. Matrone, and E. Merlino, "A wearable intelligent system for the health of expectant mom's and of their children," in *Digital System Design (DSD), 2011 14th Euromicro Conference on*. IEEE, 2011, pp. 757–763.