Hardware Implementation of Autoregressive Model Estimation Using Burg's Method for Low-Energy Spectral Analysis

Koichi Kajihara, Shintaro Izumi, Seiya Yoshida, Yuji Yano, Hiroshi Kawaguchi and Masahiko Yoshimoto
Graduate School of System Informatics
Kobe University
Kobe, Japan
Email: kajihara.koichi@cs28.cs.kobe-u.ac.jp

Abstract— **We present a hardware implementation of Burg's method, which is used for autoregressive (AR) model estimation. The AR model is a linear predictive modeling technique. It assumes that the current value of a signal can be described by a finite linear aggregate of the previous values. The AR model can be used for spectral analysis as an alternative to the Fourier transform. This approach is a parametric method, and it can yield higher resolutions than nonparametric methods in cases when the signal length is short. Although Burg's method requires a large computational capacity, especially with higher model orders, a fast Burg's method has been proposed for improving this drawback. In this study, we evaluate the influence of the order and the data length of Burg's method on the computational capacity. The hardware implementation method of the fast Burg's method including a two-stage pipeline architecture and a parallelization technique for autocorrelation calculations is proposed. The proposed method is implemented using Verilog HDL and its energy consumption is estimated with the 65-nm CMOS process. The evaluation result shows that the proposed method achieves an energy consumption of 21.6–361.4 nJ for the spectral estimation with a data length of 128–2048 points when the model order is 5.**

Keywords—AR model, Burg's method, hardware implementation, parametric method, spectral analysis

I. Introduction

Spectral analysis is widely used as digital signal processing for time series data such as vital signal analysis and speech analysis. Discrete Fourier transform (DFT) is generally used for spectral analysis because the fast Fourier transform (FFT) algorithm can obtain an accurate spectrum with a small amount of calculation. Furthermore, as most of the calculation is composed of multiplication and accumulation, it can be easily implemented in a dedicated hardware.

However, the frequency resolution of the DFT depends on the sampling rate and the number of input data length. A larger number of input points is required to obtain a higher frequency resolution. In actual applications, cases exist where the data length cannot be increased. For example, in biological signal measurement that requires real-time data analysis, the frequency characteristics are required to be instantaneously extracted from the measured data in a short time. The short data length is also effective for reducing the energy consumption of the processor and memory.

We herein focus on an autoregressive (AR) model to realize an accurate and low-energy spectral analysis with short data length. The AR model is a linear predictive modeling technique, which assumes that the current value of a signal can be described by a finite linear aggregate of the previous values. The AR model can be used for spectral analysis as an alternative to the Fourier transform [1].

In this method, the time series data is input and a specific linear system called the AR model is output. Subsequently, we obtain the frequency spectrum by comparing the AR model with the variance of white noise. Using this method, the frequency resolution can be arbitrarily determined. This approach is a parametric method, and it can yield higher resolutions than nonparametric methods such as DFT in cases when the data length is short.

However, the AR-model-based spectral analysis requires a larger number of calculations than FFT, especially with the higher model order. Furthermore, in contrast with FFT, an efficient hardware design of the AR model estimation has not been studied. These drawbacks increase the energy consumption. Therefore, this method is rarely used in systems requiring low energy consumption such as wearable devices.

To overcome this problem, we propose an efficient hardware implementation of the AR model estimation method, called Burg's method. The details of the AR model and Burg's method is introduced in section II. The computational amount of Burg's method is discussed and compared with FFT in section III. The proposed architecture and its implementation result are respectively described in sections IV and V.

II. AR Model and Spectral Estimation

A. AR Model Overview

The AR model predicts the time series data at a certain time from the linear sum of the previous data. Therefore, when the time series data $\mathrm{x}_1, \dots, x_{t-1}$are input, the expression of the AR model is defined as follows:

$$x_t = \sum_{k=1}^{m} a_k x_{t-k} + w_t \tag{1}$$

where a_k is the AR coefficient; it is a weight that indicates how much data of a certain time in the past influences the current data.

This paper is based on results obtained from a project commissioned by the New Energy and Industrial Technology Development Organization (NEDO).

m is the AR order, which is a parameter that determines the past search period in the AR model. Further, w_t indicates the prediction error between the linear sum of the past data and the actual current data.

At this time, the power spectral density (PSD) of the time series data is obtained by the following calculation:

$$PSD_{AR} = \frac{1}{Fs} \frac{\sigma^2}{\left|1 + \sum_{k=1}^{m} a_k e^{-\frac{2\pi ikf}{Fs}}\right|^2} \tag{2}$$

where Fs and σ^2 , represent the sampling frequency and variance in prediction error w_t, respectively. The variance in prediction error is the normal white noise.

To perform spectral analysis using (2), two parameters: AR order and AR coefficient must be accurately estimated. The Akaike information criterion (AIC) [2] is a typical example of the AR order determination method. However, even with such a method, the unique AR order cannot be determined. In fact, it is necessary to adjust the AR order according to input data. In this research, the AR order was designed to be variable with values up to 10. Many methods exist, such as the Yule–Walker method, Burg's method, and covariance and modified covariance methods, to estimate the AR coefficient [3]. In this work, we employed the Burg's method because it has better accuracy and stability. This method can detect clear peaks even with fine frequency resolutions compared with other AR coefficient estimation methods. In addition, it always generates a stable model. Furthermore, a fast Burg's method [4] is proposed to reduce the computational amount. The details of these methods are described in the following sections.

B. *AR Coefficients Estimation Using Burg's Method*

Burg's method predicts the AR coefficient using the least-squares method for the sum of the forward and backward linear prediction error energies of the AR model. N and n respectively represent the input data length and the order of the AR model. The sum of the squares of the forward linear prediction error energy f_i and the backward prediction error energy b_i is expressed by the following equation.

$$E_n = \sum_{k=1}^{N-n} \{f_i^2 + b_i^2\} \tag{3}$$

Where

$$f_i = \sum_{k=0}^{n} a_k X_{i+n-k}$$
$$b_i = \sum_{k=0}^{n} a_k X_{i+k} \tag{4}$$

At this time, the AR coefficient a_i is restricted by the Levinson–Durbin recursion shown below:

$$a_k = a_{k-1} + a_{nn} a_{n-l,n-l} \quad l = 1,2,\dots,n-1 \tag{5}$$

By placing this constraint equation, if the AR coefficient of the (n–1)-th order is known, the n-th order AR coefficient can also be obtained. Using these processes, the value of AR coefficient is determined by minimizing E_n. Therefore, the coefficient is determined by applying the reflection coefficient k_i obtained by differentiating E_n to the following expression (6).

$$a_{i+1} = \begin{bmatrix} a_i \\ 0 \end{bmatrix} + k_i J \begin{bmatrix} a_i \\ 0 \end{bmatrix} \tag{6}$$

where

$$k_i = -\frac{2b_i f_i}{f_i f_i + b_i b_i} \tag{7}$$

$$J = \begin{bmatrix} 0 & \cdots & 0 & 1 \\ \vdots & \iddots & 1 & 0 \\ 0 & \iddots & \iddots & \vdots \\ 1 & 0 & \cdots & 0 \end{bmatrix} \tag{8}$$

Here, J represents an N × N size matrix that inverts the matrix of N × N size vertically and horizontally.

C. *Fast Burg's Method*

The fast Burg's method, which reduces the computational capacity required for the conventional Burg's method, has been proposed in [4]. In this method, without explicitly calculating the forward prediction error energy f_i and the backward prediction error energy b_i , the reflection coefficient k_i is derived using equation (9).

$$k_i = -\frac{[a_i^T \ 0] J g_i}{[a_i^T \ 0] g_i} \tag{9}$$

where

$$g_i = \begin{bmatrix} g_i + k_{i-1} J g_{i-1} + \mathit{\Delta} R_{i+1} a_i \\ r_{i+1}^T a_i \end{bmatrix} \tag{10}$$

$$\begin{aligned} \mathit{\Delta} R_{i+1} = & -\begin{bmatrix} x_i \\ \vdots \\ x_0 \end{bmatrix} [x_i \ \dots \ x_0] \\ & - \begin{bmatrix} x_{N-i-1} \\ \vdots \\ x_{N-1} \end{bmatrix} [x_{N-i-1} \ \dots \ x_{N-1}] \end{aligned} \tag{11}$$

$$r_{i+1} = \begin{bmatrix} 2c_{i+1} \\ r_i - \begin{bmatrix} x_0 \\ \vdots \\ x_{i-1} \end{bmatrix} x_i - \begin{bmatrix} x_{N-1} \\ \vdots \\ x_{N-i} \end{bmatrix} x_{N-i-1} \end{bmatrix} \tag{12}$$

$$c_i = [x_0 \ \dots \ x_{N-i-1}] \begin{bmatrix} x_i \\ \vdots \\ x_{N-1} \end{bmatrix} \tag{13}$$

and c_i represents autocorrelation.

When the number of input data length is N and the AR order is m, the number of calculations of Burg's method is $3Nm - m^2$, and the number of calculations of the fast Burg's method is $Nm + 5m^2$. Therefore, when $m < N/3$, the computational capacity required for the fast Burg's method is smaller than that for Burg's method.

III. Performance Evaluation of Spectral Analysis Compared with Fourier Transform

A. Data Length and Accuracy

First, the spectral analysis results of FFT and Burg's method with short input data length are evaluated. As the spectral analysis accuracy of Burg's method and the fast Burg's method are the same, only Burg's method is evaluated in this section. The AR order was set to eight. A composite of sine waves of 5 Hz and 30 Hz was used as the input data. Two input data lengths of 1.0 s and 0.25 s were tested for each method. The sampling frequency of the input data was set to 256 Hz.

Fig. 1 shows the PSD estimation result using FFT and Burg's method. Although the FFT with 1-s data length accurately estimates the PSD, the peaks of the PSD estimated by FFT with 0.25-s data length are unclear. Meanwhile, all peaks can be identified regardless of the data length using Burg's method.

The FFT peak is ambiguous because the frequency resolution is determined by the relational expression between the sampling frequency shown below and the sampling time of the input data.

$$\Delta f = \frac{1}{T} = \frac{f_s}{N} \tag{14}$$

where $\Delta f, \mathrm{T}, \mathrm{f_s}, N$ denote the frequency resolution, input data time, the sampling frequency, and the number of input data length, respectively. When the input data time is set to 1.0 s and 0.25 s, the frequency resolution respectively becomes 1 Hz and 4 Hz. Therefore, it is impossible to detect the peak of an appropriate frequency. However, when Burg's method is used, the frequency resolution can be arbitrarily determined by the user. In this experiment, the frequency resolution with Burg's method is set to 1 Hz for any input data length. This result shows that even if the input is short-term data, Burg's method can maintain a high-frequency resolution compared to FFT. This method is useful as a spectral analysis method for instantaneous input data.

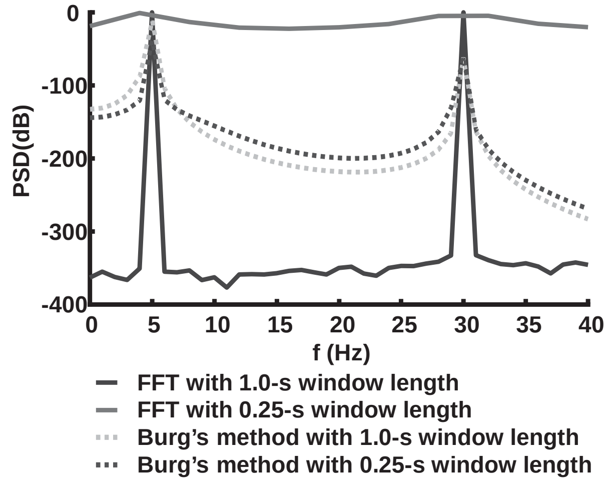


Fig. 1. Comparison of estimated PSD with 5-Hz and 30-Hz sin wave input.

B. Computational Amount

The computational amounts of Burg's method and the fast Burg's method depend on the AR order. A larger AR order increases the possibility of detecting more complicated frequency peaks but increases the computational amount. Further, if too large an AR order is used, the AR model accurately reproduces the noise, such that only the frequency peak cannot be detected clearly. Therefore, frequent cases exist where the ideal AR order is small. In this section, the difference in the computational amount between FFT and the AR method at a low AR order is compared.

Fig. 2 shows how the number of multiplications in the three methods: FFT, Burg's method, and the fast Burg's method changes with the AR order as a variable. The AR order is changed between 2 and 10. The number of input data length is set to 256 points or 1024 points. When Burg's method is used, the number of calculations becomes larger than that of FFT regardless of the AR order. However, when the fast Burg's method is used, for an input data length is 256 points, the number of calculations becomes smaller than that of FFT when the AR order is set to seven or less. For 1024 points, when the AR order is set to nine or less, the number of calculations becomes smaller than that of FFT. From this experimental result, if the spectral analysis of the input data can be executed with a low AR order, the fast Burg's method can reduce the number of calculations from the FFT. Moreover, the longer the input data length, the

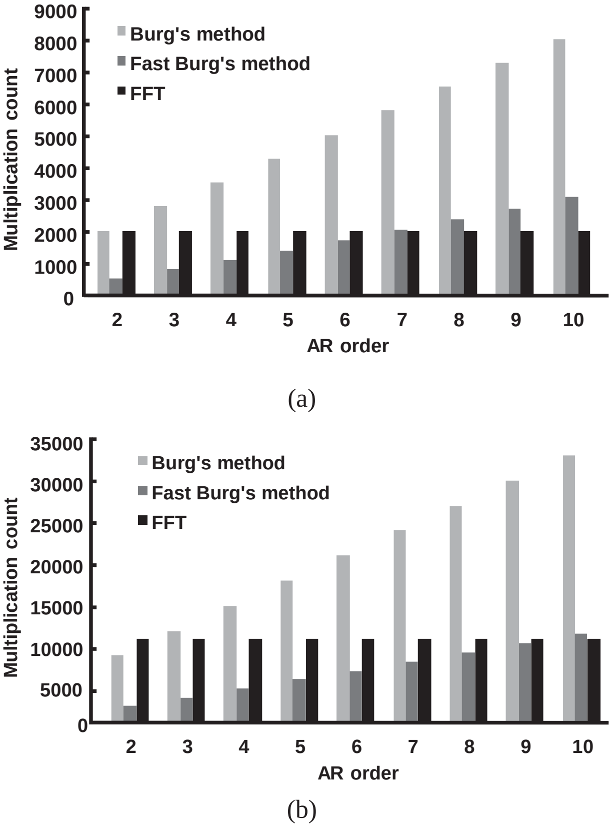


Fig. 2. Relationship between computational amount and AR order (a) with 256 points (b) with 1024 points.

closer the number of calculations of the fast Burg's method is to FFT.

IV. Signal Processing Flow and Hardware Architecture of The Fast Burg's Method

In this section, we describe the implementation method of the fast Burg's method including the AR model estimation.

Fig. 3 shows the signal processing flow of the hardware-implemented fast Burg's method. The calculation process is primarily divided into two stages. The first involves calculating the autocorrelations (AC stage). The calculation in this stage increases in proportion to the input data length. The second involves estimating the AR model (AR stage). The number of calculations in this stage increases in proportion to the AR order.

The calculation result of the AC stage depends only on the input data. Therefore, the waiting time for the AR stage processing can be eliminated. It is effective to implement the fast Burg's method using a pipeline structure.

As an architecture acceleration method, a parallel execution architecture of the AC stage is introduced. In the i-th update, the calculation of the AC stage requires $(N - i)$ cycles. Meanwhile, the AR stage requires only approximately $5i$ cycles. Therefore, in the case of N > i / 6, the AC stage can be completed in a shorter cycle than the AR stage. When the AR order is sufficiently smaller than the input data length, waiting occurs in the AR stage. By implementing the AC stage in a parallel architecture, the number of cycles required for the entire flow can be reduced. Therefore, in the case of $N > i/6$, the AC stage ends in a shorter number of cycles than the AR stage. If the AR order is sufficiently smaller than the input data length, a wait state occurs in the AR stage. By implementing the AC stage in a parallel architecture, this wait state can be eliminated and the number of cycles required for the entire flow can be reduced.

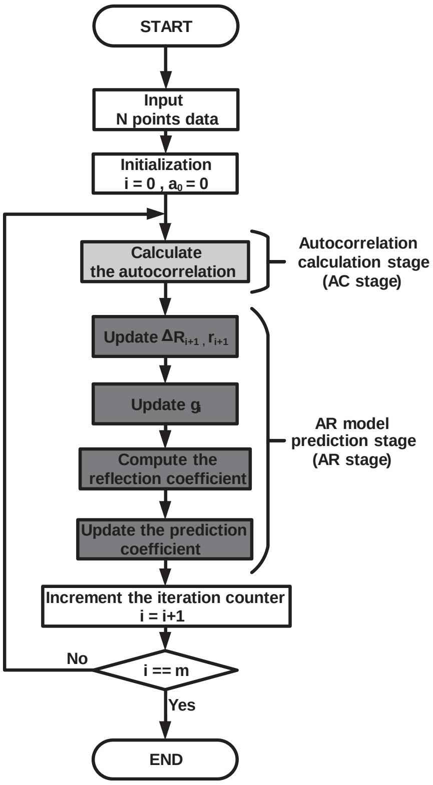


Fig. 3. Signal processing flow of the fast Burg's method.

Each time the AC stage updates, this hardware requests approximately 2N reads to the input data RAM. Therefore, the energy consumption of the RAM is very large. However, this problem can be greatly alleviated by the AC stage parallelization method. Fig. 4 shows a sample architecture incorporating two-parallel AC stage. By sharing the input of AC stage 1 with AC

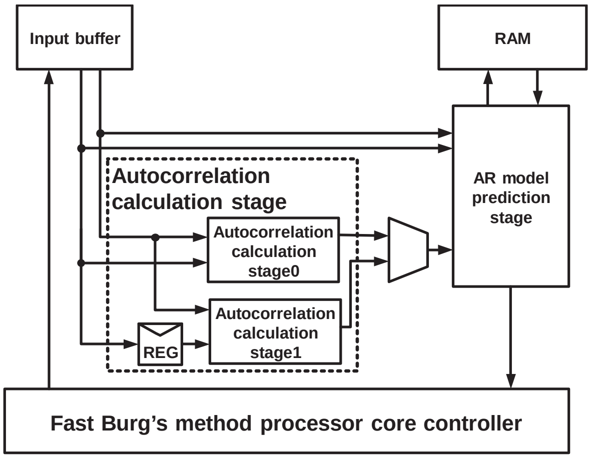


Fig. 4. Block diagram of the fast Burg's method hardware.

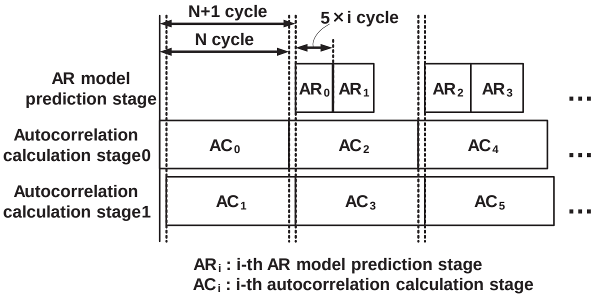


Fig. 5. Timing chart of parallel execution.

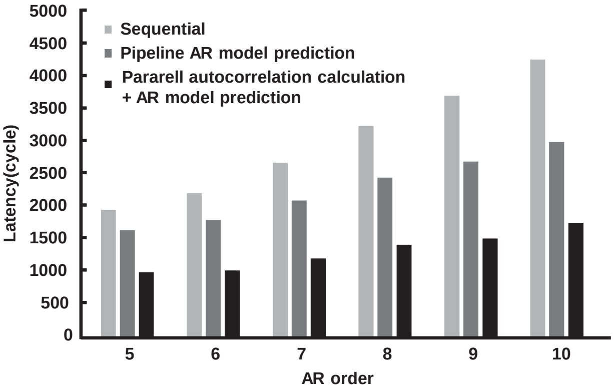


Fig. 6. Comparison of latency for AR model estimation.

stage 0, the number of reads from the input data RAM can be reduced to approximately one-half.

Fig. 5 shows the timing diagram of the architecture incorporating the two-parallel AC stage. Fig. 6 shows the effect of reducing the number of cycles by the proposed method. When the AR order is 10, the number of cycles can be reduced by 29.7% through the pipeline implementation, as compared with the case where all calculations are executed sequentially. Furthermore, by implementing the two-parallel AC stage, we achieved 58.8% reduction in the number of cycles compared with the sequential execution.

V. Implementation Result and Performance Comparison

First, we implemented the proposed architecture of the pipelined AR stage and the sequential AC stage using Verilog-HDL. For the synthesis and energy evaluation, Synopsis Design Compiler version 2017.09 was used. Fig. 7(a) shows the calculated energy consumption with 65-nm CMOS process libraries. The AR order is set to 10. The other parameters are summarized in Table I. Fig. 7(b) shows the estimated energy consumption with the two-parallel AC stage. This result is estimated from the result shown in Fig. 7(a). From these results and the RAM values, we observe that 86.3% of the energy is consumed. Moreover, we estimate that 32% reduction in energy consumption can be achieved by implementing the two-parallel AC stage.

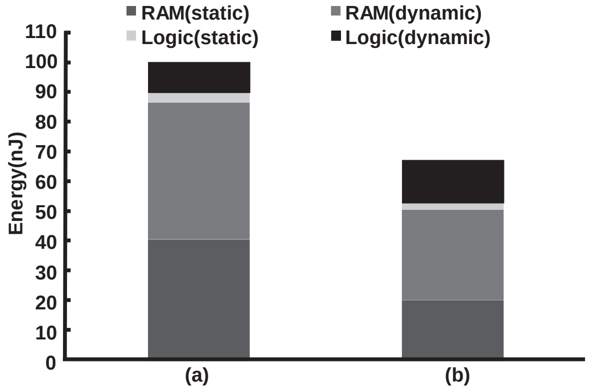


Fig. 7. Estimated energy consumption of the proposed hardware implementation. (a) with pipelined AR stage and sequential AC stage (b) with pipelined AR stage and parallel AC stage.

TABLE I. Specifications for energy estimation.

CMOS Technology	65 nm
Supply Voltage	1.2 V
Frequency	1 MHz
bit width(input/output)	12 bit / 18 bit
Input buffer size	N × 12 bit
RAM size	$\frac{\mathrm{m}}{2}(m+1) \times 18\,bit$
Data length	256
Supporting AR order	5–10

Next, the energy consumption of the proposed hardware is compared with prior works. Numerous processors for spectral analysis using FFT have been developed [6–8]. We introduce the following equation, which is defined from the literature [7] as an evaluation index.

$$FoM = Normalized\ \frac{Energy}{Execution} = \frac{Power(nW)\ \times ExecutionTime(sec)}{\left(\frac{Technology(nm)}{65nm}\right)\left(\frac{Vdd(V)}{1.2V}\right)^2} \qquad (15)$$

This formula explains the energy consumed by each PSD estimation. This formula can correct the differences in technology nodes and supply voltages. Although the proposed hardware only assumes real number spectral analysis, the comparison target [6–8] is implemented for complex FFT. The energy consumption can be reduced by up to 50% when handling real data using the complex FFT processor operation [9]. Therefore, we assumed that the energy to be compared is half of the value shown in [6–8].

Fig. 8 shows the figure of merit (FoM) in the case of 512 input data length. The AR order is set from 5 to 10. This result shows that the proposed hardware can be executed with almost the same computation energy as conventional processors when operated at a low order. When the AR order is eight, it can operate with approximately the same energy consumption as that used in the case in [8].

Fig. 9 shows the FoM when the AR order is fixed to 5. The number of input data length is set from 128 points to 2048 points. This result shows that the proposed hardware can operate with a lower energy than the conventional processor if it has a lower AR order and a longer input data point. For example, when the AR order is 5 and the number of input data length is 2048 points, it can operate with 2% less energy consumption than [6].

Finally, we compare the proposed hardware with the estimated PSD using FFT. We assume the heart rate variability analysis (HRVA) as an application herein. This application enables the monitoring of cardiac diseases and stress conditions by the spectral analysis of the heart beat interval [5]. The proposed hardware renders two advantages for this application. First, a stable high-frequency resolution can be obtained. In HRVA, it is necessary to acquire the PSD of a frequency component of 0.04 Hz to 0.4 Hz. When using the fast Burg's method, the resolution can be determined irrespective of the collected data. Therefore, a smooth frequency peak can be obtained even in the low-frequency region. Second, this method eliminates low-frequency noise. Fig. 10 (a) shows an example of the heart rate variation data. This data was obtained by converting the ECG data acquired in 128 s at a sampling rate of 1024 Hz into 128 heart rate variability data. Fig. 10 (b) shows the result of the spectral analysis of this data by the proposed hardware. The AR order was set to 10. In both methods, the results show a large peak between 0.05 Hz and 0.1 Hz. However, the PSD using FFT was significantly influenced by the DC component noise in the low-frequency region. Meanwhile, we

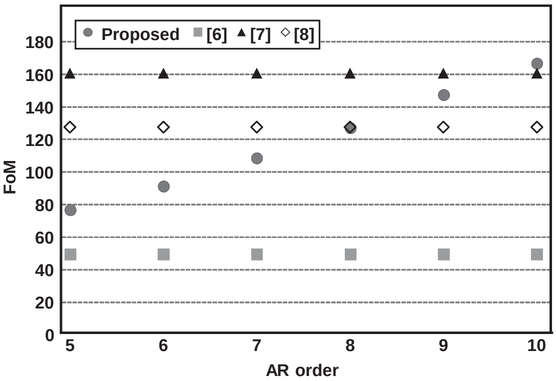


Fig. 8. FoM comparison with prior work of FFT hardware implementation. Input data length is fixed to 512 points. AR order is set from 5 to 10.

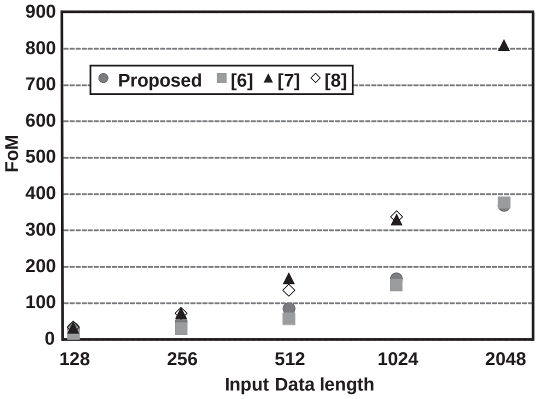
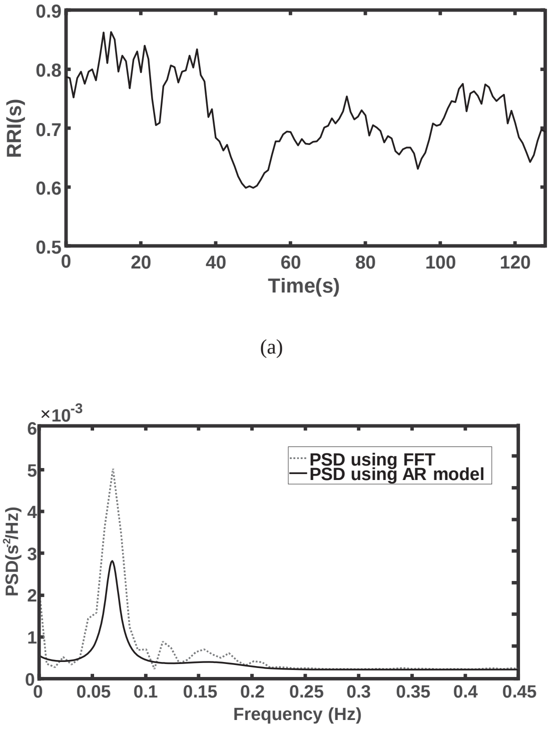


Fig. 9. FoM comparison with prior work of FFT hardware implementation. AR order is set to 5. Input data length is set from 128 points to 2048 points.

found that a clear frequency peak can be obtained even in the low-frequency region using the proposed hardware.

VI. Conclusion

We implemented the fast Burg's method for an AR model based on spectral analysis. The proposed hardware is implemented using Verilog-HDL and its energy consumption is estimated using the 65-nm CMOS process. Compared with FFT, the fast Burg's method can analyze with high-frequency resolutions even when using shorter input data length. When the AR order is set to 10 and the number of input data length is set to 256 points, the implemented hardware can operate with 99 nJ Furthermore, using the two-parallel autocorrelation stage, the energy consumption can be reduced to 66 nJ, which is nearly the same energy consumption as that of the FFT processor. Therefore, the proposed processor is capable of performing spectral analysis with a higher resolution than the conventional FFT processor with the same energy consumption.



(a)

(b)

Fig. 10. Example of heart rate variability analysis using FFT and proposed technique (a)input : heart rate (b) output : PSD of 0 Hz to 0.45 Hz

References

[1] H. Akaike, "Power spectrum estimation through autoregression model fitting," Ann. Inst. Statist. Math., vol. 21, pp. 407–419, 1969.

[2] T. Y. Kim, Y. H. Noh, D. U. Jeong, "On the use of the Akaike Information Criterion in AR spectral analysis of cardiovascular variability signals: a case report study," Proc. of Computers in Cardiology, pp. 471–474, Sep. 1993.

[3] D. Chakraborty, S.K. Sanyal, "Performance Analysis of Different Autoregressive Methods for Spectrum Estimation along with their Real Time Implementations," ICRCICN2016 IEEE International Conference, September, pp. 141–146, 2016.

[4] K. Vos, "A Fast Implementation of Burg's Method,", www.opuscodec.org/docs/vos_fastburg.pdf, August 2013.

[5] M. V. Kamath, M. A. Watanabe, A.R.M. Upton, "Heart Rate Variability (HRV) Signal Analysis" CLINICAL APPLICATIONS, CRC Press, 2012.

[6] C.-H. Yang, T.-H. Yu, D. Markovic, "Power and area minimization of reconfigurable FFT processors: A 3GPP-LTE example", IEEE J. Solid-State Circuits, vol. 47, no. 3, pp. 757–768, Mar. 2012.

[7] G. Zhong, F. Xu, A. N. Willson, "A power-scalable reconfigurable FFT/IFFT IC based on multi-processor ring," IEEE J. Solid-State Circuits, vol. 41, no. 2, pp. 483–495, Feb. 2006.

[8] A. Wang, A. Chandrakasan, "A 180-mV subthreshold FFT processor using a minimum energy design methodology", IEEE J. Solid-State Circuits, vol. 40, no. 1, pp. 310–319, Jan. 2005.

[9] M. Garrido, K. K. Parhi, J. Grajal, "A pipelined FFT architecture for real-valued signals", IEEE Trans. Circuits Syst. I Reg. Papers, vol. 56, no. 12, pp. 2634–2643, Dec. 2009.