PAPER Special Section on Technologies for Effective Utilization of Spectrum White Space

Parallel Cyclostationarity-Exploiting Algorithm for Energy-Efficient Spectrum Sensing

Arthur D.D. LIMA[†], Carlos A. BARROS[††], *Nonmembers*, Luiz Felipe Q. SILVEIRA[†], *Member*, Samuel XAVIER-DE-SOUZA[†a)], *and* Carlos A. VALDERRAMA[†††], *Nonmembers*

SUMMARY The evolution of wireless communication systems leads to Dynamic Spectrum Allocation for Cognitive Radio, which requires reliable spectrum sensing techniques. Among the spectrum sensing methods proposed in the literature, those that exploit cyclostationary characteristics of radio signals are particularly suitable for communication environments with low signal-to-noise ratios, or with non-stationary noise. However, such methods have high computational complexity that directly raises the power consumption of devices which often have very stringent low-power requirements. We propose a strategy for cyclostationary spectrum sensing with reduced energy consumption. This strategy is based on the principle that p processors working at slower frequencies consume less power than a single processor for the same execution time. We devise a strict relation between the energy savings and common parallel system metrics. The results of simulations show that our strategy promises very significant savings in actual devices.

key words: *cognitive radio, cyclostationarity, low power, parallel scalability, spectrum sensing*

1. Introduction

In recent years, Cognitive Radio (CR) has established itself as reality in the evolution of wireless communications systems [1], [2], serving as an alternative to tackle the shortage of spectrum [3], [4].

Cognitive Radio is a radio that can change its parameters based on the interaction with the environment in which it operates. Also, Cognitive Radios can learn from their own actions [5], [6]. Its key feature corresponds to the ability to identify opportunities for transmission on frequencies that are not being used at a given time by constantly sensing the range of interest in the spectrum.

There are several techniques that can be used by CR for spectrum sensing [7]–[9], among which are: a) detection by signal energy; b) sensing by waveform; c) detection by radio identification; d) matched filters; and e) cyclostationarity.

The cyclostationarity technique [10], [11] can detect the presence of a primary signal with high accuracy. However this accuracy can yield a high computational cost if proper attention is not given to the algorithm. This cost reflects directly in the energy consumption of devices that perform this type of sensing.

In this paper we propose a strategy to execute a cyclostationary sensing algorithm with reduced energy consumption. The proposed method is based on the parallelization of the Cyclic Periodogram Detection (CPD) algorithm that estimates the second order cyclostationary function [12]. Our strategy is founded in the principle that it is possible to reduce power consumption associated with the execution of a given task by using a multi-core processor with a lower frequency in place of a single-core processor. The same principle has been used for image processing, as shown in [13].

We investigate the relation between the energy savings and the scalability metrics of the implemented parallel system. The simulation results were promising, pointing towards significant power savings in actual devices.

The rest of the paper is organized as follows. The next section presents the main topics related to sensing by cyclostationarity, as well as the characterization of CPD algorithm. Section 3 covers the details of the algorithm parallelization and the conceptual bases used to its parallel implementation. In Sect. 4 we present various issues related to reducing the energy consumption of microprocessed and battery operated devices. The simulation experiments and the results are discussed in Sect. 5, and finally, we present our conclusions about the main contributions of this paper.

2. Spectrum Sensing

The most widely used techniques in spectrum sensing and identification of opportunities for transmission in Cognitive Radio (CR) are [7]: detection of signal energy [14], matched filters [15] and detection of cyclostationary characteristics [11]. Assuming a scenario in which radio equipment operates without prior knowledge of the transmitted signals, which in principle is desirable for cognitive radio systems [16], then sensing by matched filters becomes unviable [7]. In turn, sensing by detection of signal energy proves to be inefficient in scenarios with low signal-to-noise ratios (SNR) or with non-stationary noise [17]. In this sense, the detection of cyclostationary characteristics proves to be advantageous for sensing in CR.

2.1 Sensing by Cyclostationarity

The sensing by cyclostationarity [18], [19] exploits the peri-

Manuscript received June 6, 2013.
Manuscript revised September 20, 2013.

[†] The authors are with the Departamento de Engenharia de Computação e Automação, Universidade Federal do Rio Grande do Norte, Natal, RN, 59078-900, Brazil.

[††] The author is with Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte, Natal, Brazil.

[†††] The author is with Université de Mons, Mons, Belgium.

a) E-mail: samuel@dca.ufm.br

DOI: 10.1587/transcom.E97.B.326

odicity of statistical moments of the signal, such as mean and autocorrelation. The goal is to differentiate a communication signal from Additive White Gaussian Noise (AWGN). This allows the detection of a primary user but also enables the identification of signal characteristics [20], [21], such as the type of modulation used. Detecting cyclostationary characteristics proves to be quite robust even in the detection of signals with low SNR, but it bares a high computational cost [22], [23]. Many algorithms for cyclic spectral analysis proposed in the literature (i.e. FAM, SSCA and CPD algorithms) were derived from the methods originally introduced by Gardner [23]–[25].

In this paper we chose to work with the CPD algorithm [12] whose steps implement the Spectral Correlation Density (SCD). The CPD algorithm directly implements the expressions described in the theory of cyclostationarity, what is helpful to its parallelization. Furthermore, in [26], a comparative analysis among the algorithms FFT Accumulation Method (FAM) [23], Strip Spectral Correlation Analyzer (SSCA) [23] and CPD shows the equivalence of those techniques.

2.2 Formal Definitions for CAF and SCD

Cyclostationary signals present second-order statistical moments that vary periodically with time[†]. The CAF is a time-domain function that allows to identify if a signal exhibits second-order cyclostationarity [11]. A stochastic process $X(t)$ is defined as cyclostationary if its expected value, $\langle x(t)\rangle$, and its autocorrelation function, $R_x(t,\tau)$, defined as

$$R_x(t,\tau) \equiv \langle x(t)x^*(t+\tau)\rangle\,, \tag{1}$$

are periodic with a certain period T_0, that is

$$\langle x(t+T_0)\rangle = \langle x(t)\rangle\,, \tag{2}$$

and

$$R_x(t+T_0,\tau) = R_x(t,\tau) \tag{3}$$

for all t and τ. Since the autocorrelation function, in Eq. (1), of a cyclostationary process is periodic, it can be represented by a Fourier series expansion as follows

$$R_x(t,\tau) = \sum_{\alpha} R_x^{\alpha}(\tau)\mathrm{e}^{\mathrm{j}2\pi\alpha t}, \tag{4}$$

where α is called the cyclic frequency and denotes all multiples of the fundamental frequency, i.e. $\alpha = n/T_0$, with $n \in \mathbb{Z}$. Formally, the CAF is defined as the coefficients of the Fourier series, $R_x^{\alpha}(\tau)$, given by

$$R_x^{\alpha}(\tau) \equiv \frac{1}{T_0}\int_{-T_0/2}^{T_0/2} R_x(t,\tau)\mathrm{e}^{-\mathrm{j}2\pi\alpha t}dt. \tag{5}$$

Generally, stationary processes represent a particular case of cyclostationary processes for $\alpha = 0$. This way, the CAF of stationary processes is zero for $\alpha \neq 0$ while the CAF of cyclostationary processes have non-zero values for some value of $\alpha \neq 0$.

The SCD function can be used as a signature feature of the signal. Starting from the cyclic relation of Wiener [26], the SCD function is defined as a Fourier transform of the CAF expressed by Eq. (5), as follows

$$S_x^{\alpha}(f) = \int_{-\infty}^{\infty} R_x^{\alpha}(\tau)\mathrm{e}^{-\mathrm{j}2\pi f\tau}d\tau. \tag{6}$$

It is also called cyclic spectrum in the cyclic frequency α and represents the statistical correlation between the spectral components $f+\alpha/2$ and $f-\alpha/2$. The SCD function can also be calculated by means of a time-varying cyclic periodogram as follows

$$S_x^{\alpha}(f) = \left\langle X\left(t, f+\frac{\alpha}{2}\right)X^*\left(t, f-\frac{\alpha}{2}\right)\right\rangle, \tag{7}$$

where

$$X(t,v) \equiv \int_{t-T/2}^{t+T/2} x(u)\mathrm{e}^{-\mathrm{j}2\pi vu}du \tag{8}$$

is a sliding Short-Time Fourier Transform (STFT) of size T.

The following subsection presents an algorithm that allows the computation of an estimate of the SCD function for a set of samples of a signal.

2.3 CPD Algorithm

The Cyclic Periodogram Detection (CPD) algorithm was developed by Zhang and Xu [12] based on Gardner's theory [24], [25]. Its input is the sampled signal and has the following original steps:

1. The input signal is divided in L blocks of N samples;
2. Given the signal $x_l[n]$, for $l = 0, 1, \cdots, L-1$ identifying the blocks and for $n = 0, 1\cdots, N-1$ identifying the samples in the blocks, compute the Discrete Fourier Transform for each of the L blocks, i.e.
$$X_l[k] = \sum_{n=0}^{N-1} x_l[n]\mathrm{e}^{-\mathrm{j}\frac{2\pi}{N}kn}, k = 0, 1, \cdots, N-1; \tag{9}$$
3. Calculate the following expression for $k = 0, 1, \cdots, N-1$ and $l = 0, 1, \cdots, L-1$
$$T_l^{\alpha}[k] = \frac{1}{N}X_l\left[k+\frac{\alpha}{2}\right]X_l^*\left[k-\frac{\alpha}{2}\right]; \tag{10}$$
4. Calculate the mean value of $T_l^{\alpha}[k]$ among all L blocks
$$T^{\alpha}[k] = \frac{1}{L}\sum_{l=0}^{L-1} T_l^{\alpha}[k], k = 0, 1, \cdots, N-1; \tag{11}$$

[†] Cyclostationarity can be generalized to higher orders [27], but it is rarely used beyond the second order due to the substantial increment in computational complexity.

5. The resulting T^α is then smoothed in the frequency domain in order to give the cyclic periodogram

$$S^\alpha[k] = \frac{1}{M}\sum_{m=0}^{M-1} T^\alpha[kM+m]. \qquad (12)$$

Although the CPD algorithm is effective in cyclostationarity based spectrum sensing, it may become more attractive if it uses less energy to extract the cyclostationary features.

In general, the feasible cyclostationarity strategies for microprocessed and battery operated devices that have been proposed so far reduce the energy consumption by reducing the area in the plane (f, α) where the functions for extraction of cyclostationary features are calculated [23], [28].

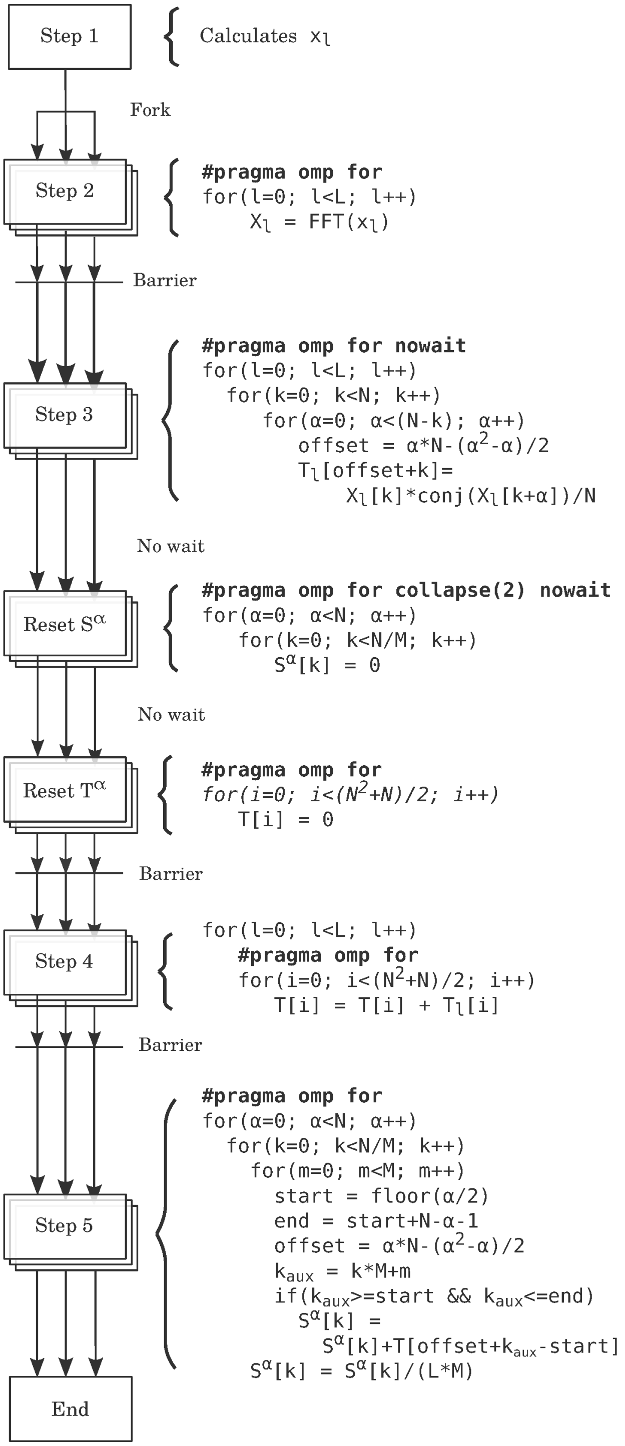
Although we would consider comparing our approach with the techniques described in [23], there is a large difference between the employed architectures, which would need porting to the multi-core architecture used here. On the other hand, the work described in [28] strongly relies on *a priori* knowledge of the signal characteristics such as modulation, roll-off factor and symbol rate, while our approach does not rely on such information. For these reasons, a direct comparison to such approaches is left to future work.

In this paper, we deal with an approach that uses less energy to extract the cyclostationary features by parallelizing the computations.

3. The Proposed Parallel CPD Algorithm

Parallel computing was revealed as a powerful resource to allow the development of solutions in different areas of knowledge that require high performance, especially in massively parallel computational contexts. In this work, we use parallelism to allow the reduction of the energy consumed to estimate the cyclostationary features in the CPD algorithm. Because it is not always straightforward to avoid excessive computation overhead caused by load imbalance and synchronization, we explain in this section our parallelization strategy for the CPD algorithm.

Figure 1 presents a block diagram of the parallelization strategy. The steps mentioned in the diagram corresponds to the steps of the algorithm described in Sect. 2.3. The actual parallelization occurs after step 1 and it is depicted by overlapping blocks, representing the computational load, and by arrows, representing executing threads, whose number can be larger than the depicted three. The necessary synchronization is carried out by the use of barriers. Between any two barriers, there are no data dependencies among different threads. The load balancing is done by scheduling N_{load} computation blocks among N_{threads} threads, with $N_{\mathrm{load}} \geq N_{\mathrm{threads}}$.



```
#pragma omp for
for(l=0; l<L; l++)
    X_l = FFT(x_l)
```

```
#pragma omp for nowait
for(l=0; l<L; l++)
  for(k=0; k<N; k++)
     for(α=0; α<(N-k); α++)
        offset = α*N-(α^2-α)/2
        T_l[offset+k]=
            X_l[k]*conj(X_l[k+α])/N
```

```
#pragma omp for collapse(2) nowait
for(α=0; α<N; α++)
   for(k=0; k<N/M; k++)
      S^α[k] = 0
```

```
#pragma omp for
for(i=0; i<(N^2+N)/2; i++)
   T[i] = 0
```

```
for(l=0; l<L; l++)
   #pragma omp for
   for(i=0; i<(N^2+N)/2; i++)
      T[i] = T[i] + T_l[i]
```

```
#pragma omp for
for(α=0; α<N; α++)
  for(k=0; k<N/M; k++)
    for(m=0; m<M; m++)
      start = floor(α/2)
      end = start+N-α-1
      offset = α*N-(α^2-α)/2
      k_aux = k*M+m
      if(k_aux>=start && k_aux<=end)
        S^α[k] =
          S^α[k]+T[offset+k_aux-start]
    S^α[k] = S^α[k]/(L*M)
```

Fig. 1 Block diagram with pseudo code of the parallel CPD algorithm.

The parallel implementation was made in the programming language C with the use of the Open Multi-Processing (OpenMP) Application Programming Interface (API) [29]. OpenMP is a standard maintained by the Architecture Review Board (ARB), whose characteristics for parallel programming includes efficiency, portability and friendly usage across many different platforms. The load distribution among threads and the management of private and shared memory locations are done with the use of directives to the compiler (`#pragma omp <directive> <clauses>`), which inserts code at the appropriate places to allow the parallelism. An introduction to the API can be found in [29].

We assume throughout the rest of the paper that the reader is familiar with the basic of this API.

Step 1 has low computational complexity, consisting of the assignment of local pointers, x_l, for $l = 0, 1, \ldots, L-1$, to the start of each of the L blocks, and therefore executes serially.

In step 2, the discrete Fourier transform is applied to each of the L blocks in parallel using the Fastest Fourier Transform in the West [30], resulting in L vectors, X_l, for $l = 0, 1, \cdots, L-1$.

Step 3 has the largest computational cost of the algorithm. Each thread computes its respective partial SCD, T_l^α, as in Eq. (10), where the non-null values are defined on the interval $-(N-\alpha)/2 \le k \le (N-\alpha)/2 - 1$, resembling a triangle. When implementing the parallel CPD algorithm, the SCD function was represented as a unidimensional and contiguous data structure with length $(N^2 + N)/2$, called T_l as in Fig. 1. The mapping of the original α inside this data structure was done by the variable `offset`, as in Fig. 1.

In order to avoid overhead, two configurations were important: the `nowait` clause, avoiding a implicit barrier at the end of the parallel loop; and the initialization of the vectors T^α and S^α between steps 3 and 4 in separate loops in order to take advantage of the load imbalance that may potentially occurs in step 3.

In step 4, an average is taken among the L partial SCDs. For this, the L blocks are traversed serially and parallelization occurs across blocks.

Finally, in step 5, the smoothening in frequency axis is performed by parallelizing the α domain among the threads.

4. Reducing Power Consumption with Multiple Processors

Particularly, in digital circuits constructed with CMOS technology, the average dissipated power can be expressed by [31],

$$P = P_{\text{sw}} + P_{\text{sh}} + P_{\text{leak}} + P_{\text{stat}}, \tag{13}$$

where P_{sw} is associated with the switching of the transistors representing the transition between binary values, P_{sh} represents the short-circuit component, P_{leak} is related to leakage currents and P_{stat} denotes the static current component. For the circuits of modern processors, P_{sw} dominates the other components and can be considered as a good first order estimate of P [13], [32], [33],

$$P \approx P_{\text{sw}} = kV^2F, \tag{14}$$

where k is a proportionality constant, and V and F are the voltage and the operating frequency of the considered circuit. Reducing voltage would, therefore, drastically reduce power consumption; however, the frequency F is limited by the same voltage V

$$F_{\max} \propto \frac{(V - V_{\text{th}})^h}{V}, \tag{15}$$

where V_{th} denotes the threshold voltage and h is a value commonly assumed to be 2 [13], [33], [34].

The use of a threshold voltage V_{th} as a parameter of a CMOS device is connected to the control of its leakage current [13] and to its margins of acceptable noise levels [32]. The value of V_{th} depends on the manufacturing technology of its transistors, mainly the gate length and V_{DS}. The device is supposed to operate for voltages higher than V_{th}. Consequently, assuming V_{th} to be a fraction of V, the maximum operating frequency $F_{\max}$ is proportional to the voltage V, and therefore, for the purpose of reducing the frequency to save energy, from Eq. (14) we have [35]

$$P \propto F^3. \tag{16}$$

Due to Eq. (16), the microprocessor manufacturing industry changed its course in mid 2000's to enter the era of multi-core processors [36]. Improving performance is since then mainly achieved through multiple parallel processing cores, replacing the strategy of continuously increasing the operating frequency of a single processing core.

The relationship between the power dissipation and the operating frequency expressed by Eq. (16) may have even more critical consequences when referring to devices powered by batteries, which have strong energy-availability limitations for their operation. However, for some recurrently executed tasks, such as cyclostationarity sensing, Eq. (16) suggests that we can use a multi-core processor with lower voltage and operating frequency to replace a single core processor with equivalent processing capacity for the purpose of obtaining a lower power consumption, given that sufficient performance can be delivered by the multiple cores.

Considering the same operating frequency, the total power consumed by a multi-core processor with p cores, P_{mc}, can be roughly estimated to be equal to p times the power consumed by a single-core processor, P_{sc}. In fact, in [37], the authors show that the total power dissipated by a multi-core processor is less than that, due to the energy savings that occur because of resource sharing. In other words, it would be beneficial to our approach to consider the multi-core power consumption to be less than the power consumed by p single-core processors, i.e. $P_{\text{mc}} < pP_{\text{sc}}$, but we rather keep the equality, i.e. $P_{\text{mc}} = pP_{\text{sc}}$, and consider our results as an upper bound for the power consumption of multi-core processors. Therefore, under the follow two reasonable assumptions:

1. It is possible to reduce the execution time of a given task that executes in a single processor core by running a parallel version of the task on p processing cores of a multi-core architecture; and

2. It is possible to increase the execution time of a parallel task by reducing simultaneously the operating frequency and voltage of the the processing cores in this architecture;

it is possible to derive the estimated energy scaling factor when using the proposed parallel CPD algorithm in a multi-core architecture by

$$\frac{E_{\mathrm{mc}}}{E_{\mathrm{sc}}} = \frac{P_{\mathrm{mc}}T_p(F_{\mathrm{mc}})}{P_{\mathrm{sc}}T_1(F_{\mathrm{sc}})}, \tag{17}$$

with E_{mc} and E_{sc} denoting the energy consumed by the multi-core and the single-core execution of the CPD algorithm, and $T_p(F_{\mathrm{mc}})$ and $T_s(F_{\mathrm{sc}})$ denoting the execution time of the parallel and the sequential tasks as a function of their operating frequencies.

By combining (16) with (17), we have the energy scaling factor as

$$\frac{E_{\mathrm{mc}}}{E_{\mathrm{sc}}} = \frac{pF^3_{\mathrm{mc}}T_p(F_{\mathrm{mc}})}{F^3_{\mathrm{sc}}T_1(F_{\mathrm{sc}})}, \tag{18}$$

and, finally, by calling E_r, F_r and T_r the scaling factors of the energy, the frequency, and the execution time, respectively, we have

$$E_r = pF^3_rT_r, \tag{19}$$

which gives us a direct relation between the energy scaling and the scaling of the frequency and the execution time when going from a single-core scenario to a multi-core scenario with p cores.

There is, however, an intrinsic relation between the scaling of the execution time and the scaling of the frequency. Although the scaling of T_r is inversely proportional to the scaling of the operating frequency, it is also affected by the speedup S_p, which is the time reduction factor achieved by a parallel task running in p processors. Since the speedup is given by the relation between the sequential executing time $T_1(F)$ and the parallel executing time $T_p(F)$, i.e.

$$S_p = \frac{T_s(F)}{T_p(F)}, \tag{20}$$

for any given frequency F. In order to devise the relation between F_r, T_r, and S_p, we start with the definition of T_r used in Eq. (19) and a few straightforward algebraic moves, as follows.

$$T_r = \frac{T_p(F_{\mathrm{mc}})}{T_1(F_{\mathrm{sc}})} = \frac{T_p(F_{\mathrm{mc}})}{T_1(F_{\mathrm{sc}})}\frac{T_1(F_{\mathrm{mc}})}{T_1(F_{\mathrm{mc}})} = \frac{1}{S_p}\frac{T_1(F_{\mathrm{mc}})}{T_1(F_{\mathrm{sc}})}, \tag{21}$$

and since

$$\frac{T_1(F_{\mathrm{mc}})}{T_1(F_{\mathrm{sc}})} = \frac{F_{\mathrm{sc}}}{F_{\mathrm{mc}}} = \frac{1}{F_r}, \tag{22}$$

we have

$$T_r = \frac{1}{S_pF_r}. \tag{23}$$

In order to guarantee that the parallel cyclostationarity sensing task executes with the same time as in the single-core processor, we must have no time scaling, i.e. $T_r = 1$, resulting in $F_r = 1/S_p$ according to (23), and, therefore, we can rewrite (19) for this specific case:

$$E_r = \frac{p}{S^3_p}, \tag{24}$$

which gives the estimated reduction factor for the energy consumed by the CPD algorithm running in parallel with the same execution time as the sequential version. Observe that p contributes to increase the consumption and S_p, which also depends on p, contributes to decrease the consumption. The energy savings in this model occurs when $S_p > \sqrt[3]{p}$.

In the next section we present the results we obtained for the speedup S_p of the CPD algorithm and the resulting reduction of energy consumption considering the model described by (24).

5. Experiments and Results

The serial CPD algorithm and the proposed parallel version were implemented with the GCC compiler 4.6.3, which supports the OpenMP API. The test platform consists of a computer with two multi-core AMD Opteron 6172 processors, each with 12 cores, totalizing 24 cores. Each processor has the following cache configuration: 12 × 128 KB L1 cache; 12 × 512 KB L2 cache; and L3 cache of 12,288 KB. This machine has 16 GB of DDR3 1,333 MHz memory and uses the operating system GNU/Linux Ubuntu 12.04 64-bit.

We analyzed the speedups for different combinations of N and L, parameters of the parallel CPD algorithm, with varying the number of threads. The number of threads ranged from 1 to 24, and each one was executed by a distinct processing core. The results show a reduction in execution time of up to 18 times when all the 24 processing cores were used, as it can be seen in Fig. 2. The area in the graphic above the limiting $S = \sqrt[3]{p}$ curve corresponds to the set of speedups which, according to Eq. (24), yields energy savings.

Speedups in the area below this curve yield energy increase. The estimated savings associated to these speedups

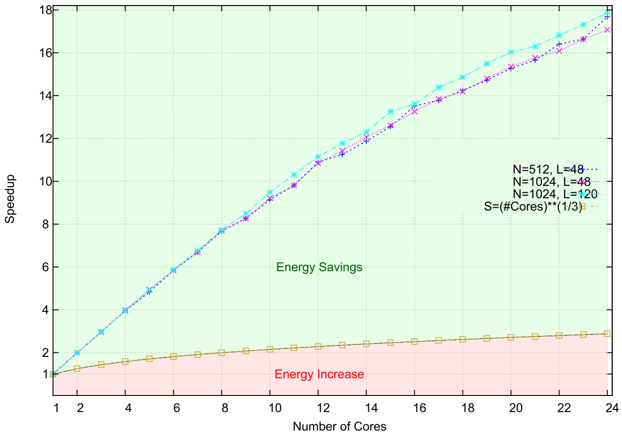


Fig. 2 Speedups for different values of L and N for number of processing cores $p = 1, 4, \cdots, 24$.

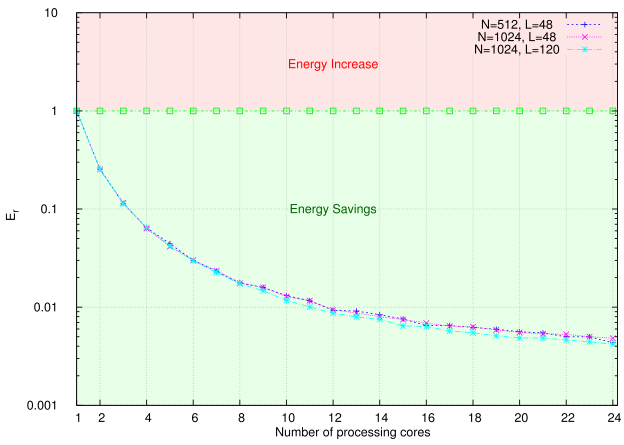


Fig. 3 Estimated energy savings according Eq. (24) for the speedups achieved by the proposed parallel approach.

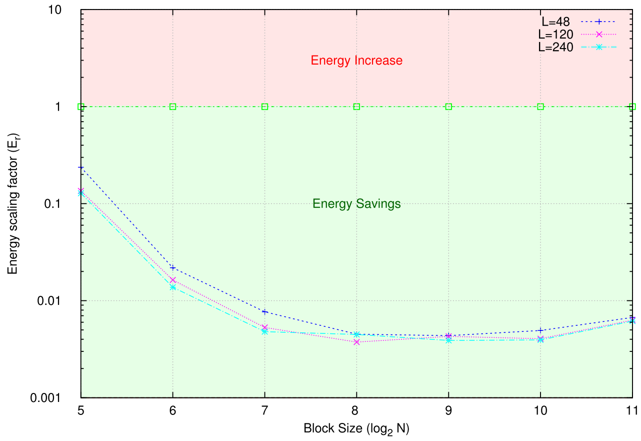


Fig. 4 Variation in energy savings when the number of blocks is kept fixed $L = 48, 120, 240$, for speedup values obtained with 24 processing cores.

can be seen in Fig. 3. Observe that, by using only 4 processing cores, it is possible to bring the energy consumption to less than 10% of the single-core scenario. Above 12 processing cores it is possible to run the proposed algorithm with less than 1% of the energy.

We have also analyzed the effects of the parameters L and N in the energy savings. For that, we measured the speedup values using 24 processing cores for different parameter combinations. Figure 4 shows the variation of energy savings when N varies and L is kept fixed. Observe that there is clear region, between $N = 2^8$ and $N = 2^{10}$, that minimizes energy consumption. A possible explanation to this effect is that there are L partial SCDs, each one with length of $(N^2 + N)/2$. Since each partial SCD is computed in a single core and exhibits quadratic growth with N, it quickly becomes too big to fit in cache memory, and then the speedup will be reduced. On the other hand, when N is fixed and L varies, there is no significant variation in energy consumption, as seen in Fig. 5.

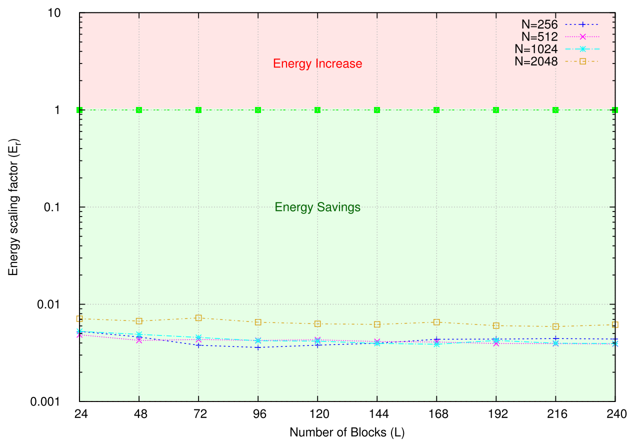


Fig. 5 Variation in energy savings for fixed block sizes $N = 256, 512, 1024, 2048$, for speedup values obtained with 24 processing cores.

Based on a simplified model that approximates the power consumption by its dominant share, the estimated energy savings that the proposed approach could achieve are very promising. If the dominance of dynamic power P_{sw} over the other terms in Eq. (13) are in the same order of magnitude of the savings estimated here, it would have a great impact on the practical implementation of cognitive radios. In the worst case, reducing the voltage and operating frequencies would make P_{sw} so low that other power components would stand out. However these components too tend to be reduced with lower voltages, even if not at the same rate. Therefore. in this case, the proposed approach would at least make P_{sw} no longer dominant, which, by itself would be a reasonable achievement.

6. Conclusions

In this paper we present a parallel CPD algorithm to calculate the second order cyclostationarity function of a signal for the purpose of spectrum sensing. It was implemented in C/OpenMP in order to assess the speedups achieved versus the number of cores. We also present an analytical model of the processor circuits that associates these speedups with a reduction factor for the consumed energy when comparing the proposed multi-core scenario with the conventional single-core type of computation. We show that the savings are estimated at more than 2 orders of magnitude compared to the conventional method. Another advantage of the propose approach is that it can be combined to the other existing methodologies to reduce power consumption for cyclostationary spectrum sensing.

References

[1] A.M. Wyglinski, M. Nekovee, and Y.T. Hou, Cognitive Radio Communications and Networks — Principles and Practice, ch. 3, Academic Press, San Diego, 2010.

[2] J. Marinho and E. Monteiro, “Cognitive radio: Survey on communication protocols, spectrum decision issues, and future research directions,” Wireless Netw., pp.147–164, 2012.

[3] D.A. Roberson, C.S. Hood, J.L. LoCicero, and J.T. MacDonald, “Spectral occupancy and interference studies in support of cognitive

radio technology deployment," 1st IEEE Workshop on Networking Technologies for Software Defined Radio Networks, 2006. SDR'06, pp.26–35, 2006.

[4] Federal Communications Commission, "Notice of proposed rule making and order: Facilitating opportunities for flexible, efficient, and reliable spectrum use employing cognitive radio technologies," ET docket, no.03-108, 2005.

[5] J. Mitola, III, Cognitive Radio — An Integrated Agent Architecture for Software Defined Radio, Ph.D. Thesis, KTH Royal Institute of Technology, Sweden, May 2000.

[6] S. Haykin, "Cognitive radio: Brain-empowered wireless communications," IEEE J. Sel. Areas Commun., vol.23, no.2, pp.201–220, 2005.

[7] T. Yucek and H. Arslan, "A survey of spectrum sensing algorithms for cognitive radio applications," IEEE Communications Surveys Tutorials, vol.11, no.1, pp.116–130, 2009.

[8] D.D. Ariananda, M.K. Lakshmanan, and H. Nikookar, "A survey on spectrum sensing techniques for cognitive radio," Second International Workshop on Cognitive Radio and Advanced Spectrum Management, 2009. CogART 2009. pp.74–79, 2009.

[9] V. Stoianovici, V. Popescu, and M. Murroni, "A survey on spectrum techniques for cognitive radio," Bulletin of the Trans. Univ. of Braşov, vol.15, no.50, pp.1–6, 2008.

[10] W.A. Gardner and L.E. Franks, "Characterization of cyclostationary random signal processes," IEEE Trans. Inf. Theory, vol.21, no.1, pp.4–14, 1975.

[11] W.A. Gardner, An Introduction to Cyclostationary Signals, ch. 1, IEEE Press, New York, 1994.

[12] Z. Zhang and X. Xu, "Implementation of cyclic periodogram detection on VEE for cognitive radio," Global Mobile Congress, Mobile World Congress Shangai, pp.1–5, 2007.

[13] E.F. Weglarz, K.K. Saluja, and M.H. Lipasti, "Minimizing energy consumption for high-performance processing," Design Automation Conference, 2002. Proc. ASP-DAC 2002. 7th Asia and South Pacific and the 15th International Conference on VLSI Design. Proceedings., pp.199–204, 2002.

[14] H. Urkowitz, "Energy detection of unknown deterministic signals," Proc. IEEE, vol.55, no.4, pp.523–531, 1967.

[15] G.L. Turin, "An introduction to matched filters," IRE Trans. Information Theory, vol.6, no.3, pp.311–329, 1960.

[16] J. Mitola, III and G.Q. Maguire, Jr., "Cognitive radio: Making software radios more personal," IEEE Pers. Commun., vol.6, no.4, pp.13–18, 1999.

[17] D. Bhargavi and C.R. Murthy, "Performance comparison of energy, matched-filter and cyclostationarity-based spectrum sensing," Signal Processing Advances in Wireless Communications (SPAWC), 2010 IEEE Eleventh International Workshop on, pp.1–5, 2010.

[18] W.A. Gardner, "Signal interception: A unifying theoretical framework for feature detection," IEEE Trans. Commun., vol.36, no.8, pp.897–906, 1988.

[19] W.A. Gardner and C.M. Spooner, "Signal interception: Performance advantages of cyclic-feature detectors," IEEE Trans. Commun., vol.40, no.1, pp.149–159, 1992.

[20] A. Al-Dulaimi, N. Radhi, and H. Al-Raweshidy, "Cyclostationary detection of undefined secondary users," Third International Conference on, Next Generation Mobile Applications, Services and Technologies, 2009. NGMAST'09, pp.230–233, 2009.

[21] F. Ge and C.W. Bostian, "A parallel computing based spectrum sensing approach for signal detection under conditions of low snr and rayleigh multipath fading," 3rd IEEE Symposium on New Frontiers in Dynamic Spectrum Access Networks, 2008. DySPAN 2008. pp.1–10, 2008.

[22] K. Po, J.i. Takada, and M. Kim, "Performance evaluation of cyclic detector in cognitive radio," IEICE Technical Report (SR), vol.107, no.352, pp.49–53, 2007.

[23] R.S. Roberts, W.A. Brown, and H.H. Loomis, Jr., "Computationally efficient algorithms for cyclic spectral analysis," IEEE Signal Process. Mag., vol.8, no.2, pp.38–49, 1991.

[24] W.A. Gardner, Statistical Spectral Analysis: A Nonprobabilistic Theory, Prentice-Hall, New Jersey, 1988.

[25] W.A. Gardner, "Exploitation of spectral redundancy in cyclostationary signals," IEEE Signal Process. Mag., vol.8, no.2, pp.14–36, 1991.

[26] W. Lei, X. Jing, Z. Tong, and Y. Xiao, "Digital implementation analysis of cyclostationary algorithms in cognitive radio," 2011 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC), pp.1–4, 2011.

[27] W.A. Gardner, "Spectral characterization of N-th order cyclostationarity," Fifth ASSP Workshop on Spectrum Estimation and Modeling, 1990, pp.251–255, 1990.

[28] Q. Thai, S. Reisenfeld, S. Kandeepan, and G.M. Maggio, "Energy-efficient spectrum sensing using cyclostationarity," 2011 IEEE 73rd, Vehicular Technology Conference (VTC Spring), pp.1–5, 2011.

[29] B. Chapman, G. Jost, and R. Pas, Using OpenMP: Portable Shared Memory Parallel Programming, ch. 1, The MIT Press, Cambridge, 2008.

[30] M. Frigo and S.G. Johnson, "The design and implementation of FFTW3," Proc. IEEE, vol.93, no.2, pp.216–231, 2005.

[31] K. Banerjee, M. Pedram, and A.H. Ajami, "Analysis and optimization of thermal issues in high-performance VLSI," ISPD'01, pp.230–237, 2001.

[32] A.P. Chandrakasan, S. Sheng, and R.W. Brodersen, "Low-power CMOS digital design," IEEE J. Solid-State Circuits, vol.27, no.4, pp.473–484, 1992.

[33] D.R. Suleiman, M.A. Ibrahim, and I.I. Hamarash, "Dynamic Voltage Frequency Scaling (DVFS) for microprocessors power and energy reduction," 4th International Conference on Electrical and Electronics Engineering, 2005.

[34] T. Mudge, "Power: A first-class architectural design constraint," Computer, vol.34, no.4, pp.52–58, 2001.

[35] E. Seo, J. Jeong, S. Park, and J. Lee, "Energy efficient scheduling of real-time tasks on multicore processors," IEEE Trans. Parallel Distrib. Syst., vol.19, no.11, pp.1540–1552, 2008.

[36] M.D. Hill and M.R. Marty, "Amdahl's law in the multicore era," Computer, vol.41, no.7, pp.33–38, 2008.

[37] R. Basmadjian and H. Meer, "Evaluating and modeling power consumption of multi-core processors," 2012 Third International Conference on Future Energy Systems: Where Energy, Computing and Communication Meet (e-Energy), pp.1–10, 2012.

Arthur Diego de Lira Lima received his degree in Computer Engineering from Universidade Federal do Rio Grande do Norte (UFRN) in 2009. Since 2011 he has been working toward a M.Sc. degree in the same university. His current research include cyclostationary signal processing and dynamic spectrum access.

Carlos Avelino de Barros received the B.S. degree in Electrical Engineering in 1989, and the M.Sc. degree in Computer and Information Sciences in 2001, both from Universidade Federal do Rio Grande do Norte (UFRN). He is currently pursuing the Ph.D. degree in Electrical and Computer Engineering at the same university. He works currently with power saving, parallelism and cognitive radio.

Luiz Felipe de Queiroz Silveira received his Bachelor Degree in Electrical Engineering, from Universidade Federal da Paraiba (UFPB), Brazil, 2000, his Master Degree (2002) and Ph.D. (2006) in Electrical Engineering, from Universidade Federal de Campina Grande (UFCG), Brazil. Since 2010, he is Professor at the Department of Computer Engineering, Universidade Federal do Rio Grande do Norte, Brazil. In 2006, he was also professor in Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte. He has experience in the Electric Engineering area, with emphasis in Telecommunication Systems, acting mainly in the following subjects: Channel Coding, Digital Signal Processing and Cognitive Radio.

Samuel Xavier-de-Souza received the Computer Engineering degree from Universidade Federal do Rio Grande do Norte (UFRN), Brazil, in 2000 and his Ph.D. degree in Electrical Engineering from Katholieke Universiteit Leuven (KULeuven), Belgium, in 2007. Worked with the Interuniversity MicroElectronics Center (IMEC), Belgium, as a Software Engineer and with the Flemish Supercomputer Centre (VSC), Belgium, as a Research Consultant. Since 2009, is a Professor at the Department of Computer Engineering and Automation at UFRN. His research interests are in Parallel Software Efficiency, Parallel Software Scalability, Parallel Algorithms, and Simulation of Complex Systems.

Carlos Alberto Valderrama obtained in 1998 the Ph.D. degree in Microelectronics at the INPG/TIMA lab in Grenoble, France, as member of the Brazilian government R&D program. In 1989, he graduated as electric-electronics engineer from the UNC, in Cordoba, Argentina. Since September 2004, he is leading the Electronics and Microelectronics Department of the Polytechnic Faculty of Mons FPMs, in Mons, Belgium. Between 1999 and 2004, he was leading the CoWare NV. Hardware Flow team located in Belgium. He was also invited professor in two Brazilian universities, in 2004 at the Federal University of Pernambuco UFPE and in 1998 at the Federal University of Rio Grande do Norte UFRN.