Real-Time Spatio-Temporal Twice Whitening for MIMO Energy Detectors

T. S. Humble, P. Mitra, and J. Barhen
Computer Science and Mathematics Division
Oak Ridge National Laboratory
Oak Ridge, Tennessee, United States of America
humblets@ornl.gov

B. Schleck
Coherent Logix, Inc.
Austin, Texas, United States of America

***Abstract*—While many techniques exist for local spectrum sensing of a primary user, each represents a computationally demanding task to secondary user receivers. In software-defined radio, computational complexity lengthens the time for a cognitive radio to recognize changes in the transmission environment. This complexity is even more significant for spatially multiplexed receivers, e.g., in SIMO and MIMO, where the spatio-temporal data sets grow in size with the number of antennae. Limits on power and space for the processor hardware further constrain SDR performance. In this report, we discuss improvements in spatio-temporal twice whitening (STTW) for real-time local spectrum sensing by demonstrating a form of STTW well suited for MIMO environments. We implement STTW on the Coherent Logix hx3100 processor, a multicore processor intended for low-power, high-throughput software-defined signal processing. These results demonstrate how coupling the novel capabilities of emerging multicore processors with algorithmic advances can enable real-time, software-defined processing of large spatio-temporal data sets.**

cognitive radio; spectrum sensing; dynamic spectrum access; MIMO; data whitening; energy detector; multicore processor;

I. INTRODUCTION

Spectrum sensing is an important component in the design and development of cognitive radio (CR) programs in which the state of the transmission environment of the primary users and, more explicitly, the frequencies available for broadcast, must be identified by the secondary users. However, spectrum sensing often amounts to near-real-time monitoring of wide frequency bands, a task that is very computationally demanding. In addition, as noted by Cabric et al., any secondary user CR must outperform the receivers of a primary user by a wide margin due to consideration of the hidden node problem [1]. This requirement further increases the computational challenge placed on the software and hardware underlying any CR program. To mitigate these computational challenges requires the use of algorithms that demonstrate less computational complexity and/or take advantage of computationally more powerful hardware.

Motivated by these considerations, we report our effort to decrease the computational complexity of one local spectrum sensing strategy for the case a secondary user's receiver is composed of multiple antennas, i.e., SIMO and MIMO transmission configurations. In Sec. II, we review a local spectrum sensing strategy based on detection of a primary user's transmitted signal, i.e., energy detection. We clarify the role of spatio-temporal twice whitening (STTW) in designing the optimal energy detector for the MIMO signal, and we detail the computational complexity associated with those calculations. We provide the main results of this report in Sec. III, where we establish a reduced computational complexity of STTW by accounting for properties of the spatially diffuse noise at the receivers. We demonstrate feasibility by implementing real-time STTW on the ultra-low-power Coherent Logix hx3100 processor. As described in Sec. IV, the hx3100 processor has a peak performance of 25 GFLOPS and a peak power efficiency of 16 GFLOPS/W. The results of this implementation, presented in Sec. V, support the prospects for STTW to be used in future CR programs. These conclusions are provided in Sec. VI.

II. LOCAL SPECTRUM SENSING

Common approaches to local spectrum sensing include cooperative sensing, interference detection, and transmitter detection. The latter detection techniques include:

- matched-filter detection;
- energy detection;
- cyclostationary feature detection.

We limit subsequent discussion to energy detection, for which the spectrum of the transmission is analyzed with respect to the presence or absence of amplitude. Generically, this may be implemented by Fourier transforming the received signal and detecting the presence of amplitude with a resolution set by the number of processed samples. While this technique is incapable of discriminating interference or more subtle transmission behaviors, such as frequency-selective fading or spread spectrum techniques, energy detection provides a relatively simple and robust method for identifying transmitter presence.

Energy detection minimizes the error associated with detection of the signal at the i^{th} antennae, i.e.,

$$\mathbf{r}^{(i)} = \mathbf{s}^{(i)} + \boldsymbol{\eta}^{(i)}, \tag{1}$$

where $\mathbf{r}^{(i)}$ is the signal received, $\mathbf{s}^{(i)}$ is the transmitted signal, and $\boldsymbol{\eta}^{(i)}$ is additive noise. For example, hypothesis testing discriminates (1) from the absence of signal ($\mathbf{r}^{(i)} = \boldsymbol{\eta}^{(i)}$) by comparing the detection statistic $\mathbf{r}^{(i)\dagger}\mathbf{H}^{(i)-1}\mathbf{r}^{(i)}$ against a

Digital Object Identifier: 10.4108/ICST.CROWNCOM2010.9299
http://dx.doi.org/10.4108/ICST.CROWNCOM2010.9299

threshold [2]. This detection statistic depends explicitly on the whitened signal vector

$$\mathbf{z}^{(i)} = \mathbf{H}^{(i)-1/2}\mathbf{r}^{(i)}. \quad (2)$$

where $\mathbf{H}^{(i)}$ denotes the noise covariance matrix for signal i, cf. Sec. III. Note the inversion of $\mathbf{H}^{(i)}$ requires $O(N_t^3)$ operations, where N_t is the number of samples. Consequently, better resolved detection are computationally very costly. Moreover, for multiple antennae, the full spatio-temporal covariance matrix $\mathbf{H}$ must be employed to whiten the signal composed from all partitions corresponding to the N_A antennae. The inversion of $\mathbf{H}$ has complexity $O(N_A^3\, N_t^3)$. The subject of the present paper is to exploit features in the spatially additive noise that reduces this complexity to $N_t \times O(N_A^3)$; given that $N_A << N_t$, this reduction in the cost of spatio-temporal data whitening can be significant.

III. Spatio-Temporal Twice Whitening

Consider a collection of N_A antennae having some fixed spatial distribution. In addition, assume each collects N_t complex-valued samples with the output defined in the temporal domain. This yields a total of $N_\tau = N_A \times N_t$ samples for the antennae array that are represented collectively by a vector $\mathbf{r}$. The vector $\mathbf{r}$ is constructed from N_A ordered partitions, e.g., $\mathbf{r}^{(i)}$ represents the i^{th} channel, such that

$$\mathbf{r} = \begin{pmatrix} \mathbf{r}^{(1)} \\ \vdots \\ \mathbf{r}^{(N_e)} \end{pmatrix}. \quad (3)$$

The i^{th} partition of samples $\mathbf{r}^{(i)}$ corresponds to either the sum of a sought after signal and the associated noise, cf. Eq. (1) or to the absence of any signal and, therefore, only the background noise. A fundamental goal of detection theory is to discriminate between these two hypotheses [2].

The spatio-temporal noise covariance matrix for the array of N_A antennae is represented by

$$\mathrm{H} = \begin{pmatrix} \mathrm{H}^{(1,1)} & \cdots & \mathrm{H}^{(1,N_e)} \\ \vdots & \cdots & \vdots \\ \mathrm{H}^{(N_e,1)} & \cdots & \mathrm{H}^{(N_e,N_e)} \end{pmatrix} \quad (4)$$

where

$$\mathbf{H}^{(i,j)} = \boldsymbol{\eta}^{(i)}\, \boldsymbol{\eta}^{(j)\dagger} \quad (5)$$

represents an $N_t \times N_t$ positive-definite submatrix formed by the outer product of the noise vectors for the i^{th} and j^{th} antennae. Whitening of the signal vector $\mathbf{r}$ refers, mathematically, to the application of the square-root inverse of $\mathbf{H}$, i.e., $\mathbf{z} = \mathbf{H}^{-1/2}\mathbf{r}$. Twice-whitening, i.e., $\mathbf{x} = \mathbf{H}^{-1}\mathbf{r}$, forgoes the square root operation, and is sufficient, for example, when computing a statistical description of the signal, c.f. the case of energy detection in Sec. II.

For an estimate of the sizes attributed to these tensors, the number of antennae may be of the order $N_A \sim 10$. Assuming each sensor collects $N_t \sim 10^6$ samples (per window), the resulting $N_\tau \times N_\tau$ noise-covariance matrix $\mathbf{H}$ has on the order of 10^{14} entries. Constructing the inverse of $\mathbf{H}$ is essential in the theory of the optimum receiver [1]. The computational complexity of matrix inversion scales as $O(N_\tau^3)$; however, due to the symmetric positive semi-definiteness of $\mathbf{H}$, inversion be more efficiently obtained by the Cholesky decomposition [4]. Yet, even then, direct calculation of the twice-whitened signal vector $\mathbf{x}$ is exceedingly expensive.

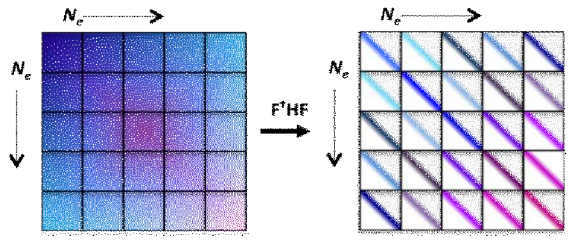

Figure 1. Block structure of the covariance matrix $\mathbf{H}$ before (left) and after (right) Fourier transformation. Symmetry arises from the condition of spatially diffuse noise sources (neglecting differences due to conjugations). The Fourier transform reveals the block-digaonal submatrix Λ.

To overcome the poor scaling associated with direct computation of $\mathbf{H}^{-1}$, we reformulate data whitening to exploit the spatio-temporal organization of array data. First, we limit consideration of the noisy background to spatially diffuse noise that is wide-sense stationary. This imposes the restriction that the submatrices of $\mathbf{H}$ be Toeplitz and that the full matrix be block Toeplitz. Consequently, due to the Hermitian form of $\mathbf{H}$ following Eq. (4), the noise covariance matrix exhibits the block-diagonal structure illustrated in Fig. 1. Note that this approximation excludes contributions from discrete interferes to sources of noise in Eq. (1).

The spectral theorem yields the eigenvalues of each $N_t \times N_t$ submatrix $\mathbf{H}^{(i,j)}$ for $i, j = 1$ to N_A, while Fourier analysis of the full covariance matrix $\mathbf{H}$ yields

$$\mathbf{H} = \mathbf{F}\, \boldsymbol{\Lambda}\, \mathbf{F}^{\dagger}, \quad (6)$$

where $\boldsymbol{\Lambda}$ is an $N_\tau \times N_\tau$ band-diagonal matrix composed of diagonal submatrices $\Lambda^{(i,j)}$ representing the eigenvalues of the corresponding noise covariance between channels i and j, and $\mathbf{F}$ symbolizes the block-diagonal form of the discrete Fourier transforms applied to each block. The resulting structure of $\boldsymbol{\Lambda}$ is illustrated in Fig. 1. Elements of $\boldsymbol{\Lambda}$ may then be permuted to form the block-diagonal matrix

$$\mathbf{K} = \boldsymbol{\Omega}\, \boldsymbol{\Lambda}\, \boldsymbol{\Omega}^{T}, \quad (7)$$

where $\boldsymbol{\Omega}$ symbolizes the orthogonal permutation matrix and the k^{th} block $\mathrm{K}^{(kk)}$ is $N_A \times N_A$ for $k = 1$ to N_t. See Fig. 2.

Inversion of $\mathbf{H}$ can now be recast as

$$\mathbf{H}^{-1} = \mathbf{F}\, \boldsymbol{\Omega}\, \mathbf{K}^{-1}\, \boldsymbol{\Omega}^{T}\, \mathbf{F}^{\dagger}. \quad (8)$$

Digital Object Identifier: 10.4108/ICST.CROWNCOM2010.9299
http://dx.doi.org/10.4108/ICST.CROWNCOM2010.9299

The cost of computing the inverse of **H** via the inverse of K then reduces from $O(N_\tau^3)$ to $O(N_t\ N_A^3)$, where inversion of the N_t submatrices of K each carry $O(N_A^3)$ cost. With respect to the resource estimates cited at the beginning of Sec. II, exploiting the spatial structure of the noise covariance matrix reduces the cost of inversion by ~6 orders of magnitude.

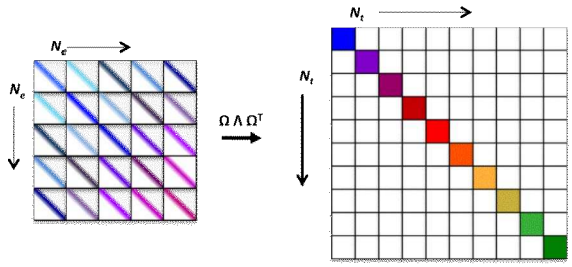

Figure 2.Transformation of the band-diagonal matrix Λ by the permutation matrix Ω yields the block-diagonal matrix K, where each diagonal block $K^{(kk)}$ is $N_A \times N_A$ in size.

Given the block-diagonal structure of K illustrated in Fig. 2, twice whitening the spatio-temporal data reduces to a system of N_t uncoupled equations, i.e., for $k = 1$ to N_t,

$$\mathbf{K}^{(kk)}\,\mathbf{y}^{(k)} = \mathbf{d}^{(k)}, \tag{9}$$

where $\mathbf{y}^{(k)}$ and $\mathbf{d}^{(k)}$ represent $N_A \times 1$ ordered partitions of the transformed vectors

$$\mathbf{y} = \mathbf{\Omega}^T \mathbf{F}^\dagger \mathbf{x} \quad \text{and} \quad \mathbf{d} = \mathbf{\Omega}^T \mathbf{F}^\dagger \mathbf{r}, \tag{10}$$

respectively. Solutions to the latter recover the twice-whitened vector **x**. Using the Cholesky decomposition

$$\mathbf{K}^{(kk)} = \mathbf{L}^{(k)}\,\mathbf{L}^{(k)\dagger}, \tag{11}$$

with $\mathbf{L}^{(k)}$ a non-singular lower triangular matrix, the twice-whitened partition $\mathbf{y}^{(k)}$ can be recovered from Eq. (9) by first solving for the intermediate result $\mathbf{b}^{(k)}$, defined by

$$\mathbf{L}^{(k)}\,\mathbf{b}^{(k)} = \mathbf{d}^{(k)}, \tag{12}$$

using forward elimination, before solving the equation

$$\mathbf{L}^{(k)\dagger}\,\mathbf{y}^{(k)} = \mathbf{b}^{(k)}. \tag{13}$$

This result can then be permutated and inverse transformed in order to obtain the twice-whitened signal vector **x**.

IV. HyperX hx3100

A. *HyperX Architectural Overview*

The hx3100 processor is the latest entry in the HyperX family of ultra low power, massively parallel processors produced by Coherent Logix, Inc [3]. The hx3100 processor is a 10-by-10 array of processing elements (PEs) embedded on an 11-by-11 array of data management and routing units (DMRs). At a system clock frequency of 500 MHz, the maximum chip-level throughput for 32-bit floating-point operations is 25 GFLOPS. Alternatively, when power consumption is prioritized, performance can be measured as 16 GFLOPS/Watt. This translates into energy consumption on the order of 10 picoJoules (pJ) per instruction that rivals the performance of dedicated ASIC designs.

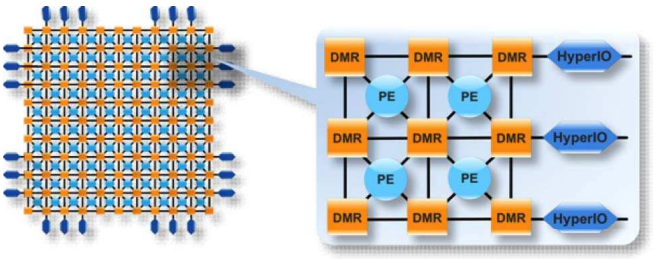

Figure 3. The Coherent Logix hx3100 processor, a 10-by-10 array of processing elements (PEs) inleaved by an 11-by-11 array of data management and routing units (DMRs). An exapnded view of four PEs surrounded by 9 DMRs demonstrates the degree of connectivity. HyperIO references the input/output ports used by the hx3100 processor to community with off-chip memory or other hx3100 processors.

The DMR network provides Direct Memory Access (DMA) for the PEs to both on-chip and off-chip memory. Each DMR has 8 kB of SRAM and operates at a read/write cycle rate of 500 MHz, while the eight independent DMA engines within a DMR may act in parallel. A PE directly accesses data memory in four neighbouring DMRs, such that 32 kB of data is addressable by any given PE. In addition, when a subset of PEs shares direct access to the same DMR, the associated memory space may act as shared memory between PEs. This inter-connectedness provides for a mixture of shared and distributed memory hierarchies. Each DMR consists of 8 16-bit DMA engines that route memory requests from neighbouring PEs and support routing memory requests managed by other DMRs. In addition to supporting on-chip DMAs, the DMRs handle requests to off-chip memory, including the eight DDR2 DRAM ports. Moreover, the 24 IO ports surrounding (six per side) the chip can be wired to connect together multiple HyperX chips.

B. *Integrated Sofware Development Environment*

Programming the HyperX entails writing an ANSI C program, which defines the parallelism in the algorithm through the use of the industry standard Message Passing Interface (MPI) protocol. The Coherent Logix software tools automatically assign individual tasks to PEs, and create routing to support the movement of data between PEs. Tasks may be both parallelized and pipelined across multiple cores, while DMAs are controlled explicitly in software. The latter steps provide opportunities for designing the flow of program execution such that resource constraints and programming requirements are met.

C. *Power Management*

The current version of the HyperX architecture provides the capability to power down quadrants of the PE grid that are unneeded by a designed application. As a result,

Digital Object Identifier: 10.4108/ICST.CROWNCOM2010.9299
http://dx.doi.org/10.4108/ICST.CROWNCOM2010.9299

programs can be optimized with respect to energy usage (of the chip) as well as the computational speed and memory bandwidth usage. Wake-up signals can be triggered by external events, such that the processor may be shutdown during period of computational idle time.

V. TWICE-WHITENING ON HX3100

A. Program Design

Our implementation of twice whitening on the hx3100 organizes the algorithm into multiple, pipelined and parallelized programs occupying 17 of the 100 available PEs. Refer to Fig. 5 for locations of the programs described below. First, prior to any data processing, simulated input data is loaded into off-chip DRAM memory banks. Here, the data is single-precision, complex-valued signal vectors stored in sample-consecutive order in DRAM 1. At runtime, program IOPE 1 fetches one input vector from DRAM 1 and copies the data onto the chip via a series of 4 routing programs labeled RPEs. The latter manage and synchronize the input data for a complex-to-complex FFT program.

Upon completion of the FFT of one input vector, the RPEs trigger IOPE 2 to store the transformed data off-chip in DRAM 2. Storage of the intermediate results is required to free on-chip memory for processing the FFT of the next input vector. The copying process occurs directly from the DMRs used by the FFT cores and, specific to our algorithm, requires a substantial amount of processing time. This overhead results from the use of a strided-write operation in which each complex-valued sample is written to DRAM with a stride of N_A. The purpose of this operation is to perform the permutation denoted by Ω in Fig. 2, cf. Fig. 4.

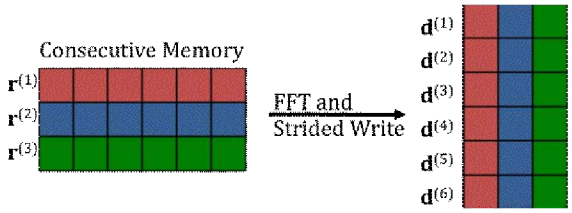

Figure 4. How input vectors are sequentially transformed and permutated using a N_A–strided write to DRAM. Consecutive memory is from left to right. As a result, the vector $\mathbf{d}^{(k)}$ read directly from DRAM 2 can be immediately used for the twice-whitening operation defined by Eq. (9).

Once all N_A input vectors have been transformed and written to DRAM 2, IOPE 2 changes state and begins to fetch the permutated, length-N_A vectors that represent the partitions $\mathbf{d}^{(k)}$ defined in Eq. (10). IOPE 3 fetches a precomputed Cholesky decomposition from DRAM 3. The latter is represented by the $N_A (N_A - 1) / 2$ elements of an upper triangular, complex-valued matrix obtained by offline decomposition of the matrix $\mathbf{K}^{(kk)}$. Future versions of this program will implement the Cholesky decomposition of each matrix alongside the transformation and permutation of the signal vectors. Both sets of data are moved to the DMRs of the whitening program (LPE). Within this program forward elimination (12) and back substitution (13) solve Eq. (9), both requiring $O(N_A^2)$ operations to complete. The output of this program is written back to DRAM 4 via IOPE 4. Again, a strided-write operation is used, with the N_A components (samples) spaced by N_t positions; once all N_t solutions are obtained, DRAM 4 stores, in consecutive order, the N_A partitions of $\mathbf{\Omega y}$ defined in Eq. (10), i.e., the FFT of the twice-whitened vector $\mathbf{x}$.

An important programming consideration is the synchronization between individual PEs to ensure continuity and correctness of data movement. The 8 kB memory available from each DMR necessitates precise accounting for the size, location, and timing of each data segment. This inter-core communication is performed using a proprietary API written for the C language. For example, blocking variants of DMA send and receive functions provide a basic communication mechanism for data synchronization, while techniques are available for reading and writing to DRAM.

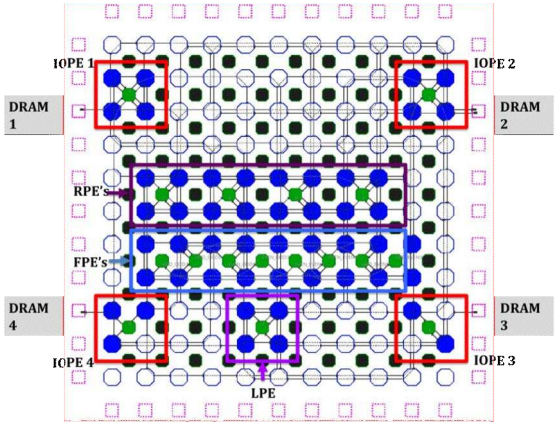

Figure 5. Layout of the STTW program on the hx3100 processor. Boxes circumscribe PEs and DMRs for each individual program, cf. Table 1.

The placement and routing of individual programs, e.g., the IO servers or the FFT cores, with respect to the layout of available cores is also an important consideration for performance. While this level of detail can be programmed explicitly, the HyperX software development environment provides a global optimizer and scheduler to perform placement and routing. For example, routing constraints ensure the IO server programs are placed adjacent to the off-chip IO ports and that on-chip DMA routes do not collide, e.g., when transferring data between PEs.

B. Program Results

As implemented, the full program processed a series of 8192-complex-point input vectors, each representing a noisy sinusoidal typically of a monotone acoustic source collected against a diffusive (thermal) background. At a sampling rate of 64KHz, this amounts to ~125 ms of data. When running on the hx3100 processor, simulated input vectors are loaded into the off-chip DDR2 memory banks. Vectors are stored in deinterleaved, bit-reversed order, with a single input vector representing 64 kB of memory. Eight DMA transfers are required to load a collection of 8 on-chip DMRs. This IO service requires 1 core for managing request to the DDR2 and, in a double buffering approach, 3 neighboring DMRs to optimize timing of the data transfer.

Digital Object Identifier: 10.4108/ICST.CROWNCOM2010.9299
http://dx.doi.org/10.4108/ICST.CROWNCOM2010.9299

An 8192-point, complex-to-complex decimation-in-time FFT is used to transform the input data. Each instance employs 8 PEs, cf. Fig. 5. The FFT algorithm requires bit-reversed-order input data and reordering of the input is done prior to loading the DRAM. The input and output servers are synchronized to avoid overwriting data stored in the FFT processing cores.

Table 1. Program resources used, including PEs, DMRs, and cycles per iteration of each program element.

Program	PEs	DMRs	Cycles/iteration
IOPE 1	1	4	37,494
RPE's	4	16	N/A
FPE's	8	18	137,741
IOPE 2 (write)	1	4	565,749
IOPE 2 (read)	1	1	136
IOPE 3	1	2	358
LPE	1	4	4,384
IOPE 4	1	3	539
TOTAL	17	52	42,014,925

We have implemented the above on the HyperX hx3100, and we have profiled the program elements using the Integrated Software Development Environment. The results below refer to version 3.0.1 of the HyperX ISDE tools for the case of $N_A = 8$ and $N_t = 8192$ complex samples. The number of cycles, normalized to the relevant input type, are shown for each program in Table 1. For example, IOPE 1 requires 37,494 clock cycles to read one input vector and synchronize transfer of that vector to the RPEs memory space. This accounting of clock cycles highlights the processing bandwidth associated with each program; data transfer is not computationally complex, but a moderate amount of time is needed to bring the data onto the chip. Regarding IOPE 1, the reported cycles include sending data to the RPE's, while the cycles reported for the FPEs incorporates the read operation from the RPE memory space. IOPE 3 reads the transformed and permutated vectors $\mathbf{d}^{(k)}$ from DRAM 3; recall the latter are identified by the sample index k and there are a total of 8192 vectors to process. The other program elements have similar interpretations of the reported cycle counts.

IOPE 2 represents the most costly operations in this implementation. As stated in Sec. IV.A and illustrated by Fig. 4, IOPE 2 is performing a sample-by-sample strided write of the FFT output. Many samples (8192) must be processed this way and the cost associated with strided writes to DRAM is high. More important, the IOPE 2 write operation represents a significant bottleneck in this implementation as the write operation takes significantly more time than the FFT processing. However, the size of the processed data set (8 × 8192 samples = 512 KB) prohibits performing transposition within on-chip memory space. The cost of the strided-write operation is constant with respect to the number of input vectors.

Our present implementation omits concurrent calculation of the Cholesky decompositions for permutated and transformed noise covariance matrices $\mathbf{K}^{(kk)}$. Note that those calculations utilize a method *identical* to the computation of $\mathbf{y}$ (in timing and complexity), and that the decomposition of each $\mathbf{K}^{(kk)}$ could be easily accommodated within the LPE provided N_A is not too large (~35).

Table 2. Power consumption with respect to clock frequency and voltage.

	Tunable PE Clock Frequency (MHz)			
Voltage(V)	200	300	400	500
1.00 V	914.29	1162.15	1410.01	1657.89
0.95 V	865.95	1089.65	1313.35	N/A
0.90 V	820.09	1020.86	1221.64	N/A

Table 2 reports the total power consumption with respect to the PE clock frequency and operating voltage. The clock, tuned from 200 MHz to 500 MHz, determines the overall time-to-solution of the program. Based on the total cycles reported in Table 1, a range of 84.2 ms at 500 MHz to 210 ms at 200 MHz is observed. Lowering the voltage, from 1.0 V to 0.95 V to 0.9 V, provides a corresponding decrease in power. A two-fold increase in power occurs between the case of 0.90 V at 200 MHz and the 1.0 V at 500 MHz. Depending on the performance requirements, STTW on the hx3100 satisfies a range of power and timing constraints.

VI. CONCLUSIONS

We have presented an implementation of STTW on the hx3100 processor. Our implementation exploits spatially diffuse noise to reduce the computational complexity by a factor of N_t^2, amounting to several orders of magnitude. We demonstrated real-time STTW on the hx3100 processor. The hx3100 processor provides a platform capable of handling multiple instruction, multiple data (MIMD) processing, and the ability for block-stream processing of MIMO data. The high throughput, low-power features of our realization suggest opportunities to employ real-time STTW signal processing techniques for MIMO energy detectors.

ACKNOWLEDGMENT

This work was supported by the United States Office of Naval Research. Oak Ridge National Laboratory is managed for the US Department of Energy by UT-Battelle, LLC under contract DE-AC05-00OR22725.

REFERENCES

[1] D. Cabric, S. M. Mishra, and R. W. Brodersen, "Implementation issues in spectrum sensing for cognitive radios," Signals, Systems and Computers, 2004. Conference Record of the Thirty-Eighth Asilomar Conference on, Vol. 1 (2004), pp. 772-776 Vol.1.

[2] H. L. van Trees, Detection, Estimation, and Modulation Theory, Vol. 1, New York: John Wiley & Sons, Inc., 2001.

[3] www.coherentlogix.com

[4] G. H. Golub and C. F. Van Loan, Matrix Computations, Baltimore, ND: John Hopkins University Press, 1996.

Digital Object Identifier: 10.4108/ICST.CROWNCOM2010.9299
http://dx.doi.org/10.4108/ICST.CROWNCOM2010.9299