High Performance FFT on Multicore Processors

J. Barhen[1], C. Kotas[1], T. S. Humble[1], P. Mitra[1], N. Imam[1], M.A. Buckner[2] and M. R. Moore[2]

[1] Center for Engineering Science Advanced Research
Computer Science and Mathematics Division

[2] RF and Microwave Systems Group
Measurement Science & Systems Eng. Division

Oak Ridge National Laboratory
Oak Ridge, Tennessee, United States of America
barhenj@ornl.gov

Abstract. **The importance of achieving high performance Fourier transforms for Cognitive Radio applications can not be over-emphasized. This includes signal detection in the presence of noise power uncertainty, multi-resolution spectrum sensing, minimization of subcarriers' side lobes in OFDM modulators, multi-stream processing, or spectrum loading, for example. With the emergence of advanced multicore processors, there is a remarkable opportunity to develop novel, massively parallel implementations of the FFT. This paper reviews recent advances in the area, and presents results for three classes of devices: the IBM Cell multi-SIMD processor, the Nvidia Tesla SIMT processor, and the EnLight digital optical core device.**

cognitive radio; ***FFT***; ***multicore procesors***; ***optical core processors***; ***transverse vectorization.***

I. INTRODUCTION

In recent years, there has been a tremendous growth of interest in efficient implementations of the Fast Fourier Transform (FFT) to support Cognitive Radio applications. This has been motivated primarily by the critical role FFTs play in signal detection approaches to channel sensing. With the emergence of revolutionary computing technologies, there is a strong rationale to review recent advances, in particular as related to various multicore and digital optical core implementations. This is the objective of this paper.

The physical layer of Cognitive Radio (CR) systems may involve different multi-carrier communication methods. A key common requirement is that spectral holes (unused parts of the RF spectrum) be determined dynamically. Other critical requirements include high spectral-resolution, a capability to estimate the average power in each sub-band of the spectrum, determining unknown directions of potentially interfering signals, and the ability to minimize the side-lobes of each subcarrier band to enable data transmission without interference with other parts of the frequency band.

A standard assumption is that 4G communications will be based on Orthogonal Frequency-Division Multiplexing (OFDM) [1]. Hence, related algorithms such as the FFT (and its inverse) may need to be adapted to the specific CR context. For example, it has been pointed out [2] that the requirement to nullify individual carriers to avoid potential interferences may result in highly sparse data sequences (ie, OFDM sequences with a large number of zero components) during white-space detection. Conventional FFT algorithms may be less efficient in such instances, and various schemes are being proposed [2] to overcome this challenge.

It has been widely recognized that the FFT, as part of an OFDM modulator, can also be used for channel sensing. To that effect, filter banks for multicarrier communications and methods for spectral analysis in a CR setting have been discussed [3]. Since many signals of interest exhibit cyclostationarity, the multitaper method (MTM) for nonparametric spectral estimation has been used for rapid spectral analysis. MTM can also perform space-time processing and time-frequency analysis [4].

Spectrum loading is a dynamic spectrum control technique for broadband CR systems. It involves the mapping of discrete spectrum components of a framed symbol sequence onto available spot-wise spectra [5]. Computational schemes based on FFTs have been designed that effectively exploit the spot-wise spectra distribution.

Physical layer spectrum sensing techniques based on FFT have been shown to enable detection of signals on a TV channel at levels as low as -116 dBm. [6]. To overcome the considerable performance degradation of conventional energy detection schemes in the presence of noise power

Digital Object Identifier: 10.4108/ICST.CROWNCOM2010.9283
http://dx.doi.org/10.4108/ICST.CROWNCOM2010.9283

uncertainty, an approach that combines the FFT and Max-to-Mean Ratio (MMR) has been shown as highly effective (in terms of noise power independence and lower sample size requirement) for weak narrowband signals [7].

It is often of interest to estimate the Signal-to-Noise power Ratio (SNR) from observed samples that are taken asynchronously. Iterative methods for computing the Maximum Likelihood estimate for this problem are typically based on the Karhunen-Loeve Transform (KLT) of the data vector. An interesting recent development shows that the computationally intensive KLT can be superseded by the more efficient FFT, asymptotically achieving the same performance [8].

Finally, interesting studies have been published in the area of energy based multi-resolution spectrum sensing techniques. Here, the emphasis has been on the design of adaptive FFT algorithms that can focus on a small part of the interested bands with finer resolutions and at lower computational costs [9]. Implementation on reconfigurable platforms has also been examined. In another hardware perspective, architectures for pipelined FFTs with multi-stream OFDM processing have been proposed [10].

The above references emphasize the central role of the FFT algorithm in CR. While the ultimate goal is clearly the design of adaptive FFT schemes, a first and essential step is to understand the considerable gain in computational throughput that can be achieved from emerging processor technology.

Motivated by these considerations, we report on recent advances in developing efficient FFT algorithms for implementation on emerging multicore processors. Section II focuses on the *EnLight* digital optical core processor. The novel FFT transverse vectorization scheme, as related to the IBM Cell multi-SIMD processor, is discussed in Section III. In Section IV we present the initial results of our SIMT (Single Instruction Multiple Thread) implementation of the FFT on the NVIDIA Tesla. Conclusions are summarized in Section V.

II. DIGITAL OPTICAL CORE PROCESSOR

Revolutionary computing technologies are defined in terms of technological breakthroughs, both at the device and algorithmic levels, which leapfrog over near–term projected advances in conventional hardware and software to (potentially) result in paradigm shifts in computational science. One of the most promising advances in this area builds upon the emergence of digital optical devices.

Optical processing is inherently capable of high-parallelism that can be translated to very high performance computing. In recent years, the Center for Engineering Science Advanced Research (CESAR) at the Oak Ridge National Laboratory (ORNL) has been collaborating with Lenslet Laboratories (Israel), to explore the feasibility of demanding signal processing computations using the novel, Lenslet–developed, *EnLight*™ platform [11].

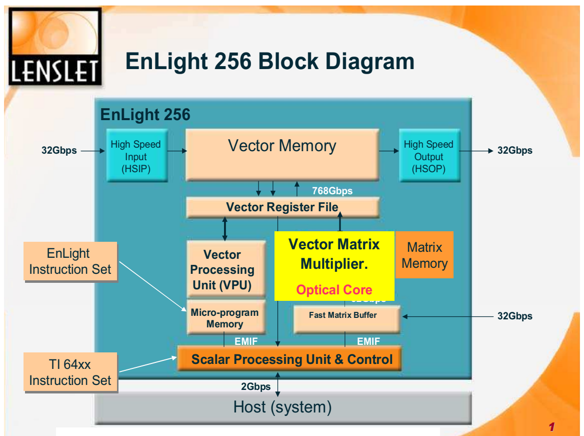

Fig. 1 *EnLight*^{TM} processor architecture

The *EnLight*™256 is a small factor signal processing chip (5×5 cm^2) with a *digital optical core*. The processor is optimized for array operations, which it performs in fixed point arithmetic. The native precision is 8–bit per clock cycle. Higher precision can readily be obtained by using more than one clock cycle per MVM [12].

The architecture of a computational node is shown in Fig 1. The optical core performs matrix-vector multiplications (MVM), where the nominal matrix size is 256×256. The system clock is 125MHz. At each clock cycle, 128K multiply-and-add operations are carried out, which yields a peak performance of 16 Trillion Operations per Second (*Tera*OPS). From a computational complexity perspective, the paradigm-shifting nature of this processor stems from the fact that a complete MVM can be achieved in a single clock cycle. Moreover, the power dissipation per board is approximately 40 Watt. Due to the inherent parallelism of the architecture, the computational speed increases with the scale of the problem. The scaling penalty of the optical chip is relatively small compared to standard DSP electronics.

At ORNL, we had access to the *EnLight*™ *Alpha* hardware, shown in Fig 2. This is a demonstrator board for the optical processing technology, with a reduced size 64×64 optical core. The *EnLight*™ *Alpha* clock operates at 60 MHz. The optical core has 64 input channels (256 vertical cavity surface emitting lasers, configured in groups of 4 per channel, hence the 8 bit native precision). The size of the active matrix is 64×64; it is embedded in a larger multiple quantum well (MQW) spatial light modulator of size 264×288. There are 64 output channels (64 light detectors integrated with an array of analog-to-digital converters). The

Digital Object Identifier: 10.4108/ICST.CROWNCOM2010.9283
http://dx.doi.org/10.4108/ICST.CROWNCOM2010.9283

optical core performs the MVM function at the rate of $60\ 10^6 \times 64^2 \times 2$ = 492 Giga operations per second. The 64 input and 64 output channels produce data streams of $60\ 10^6 \times 64 \times 8$ bits/s = 30.7 Giga-bits per second, each way.

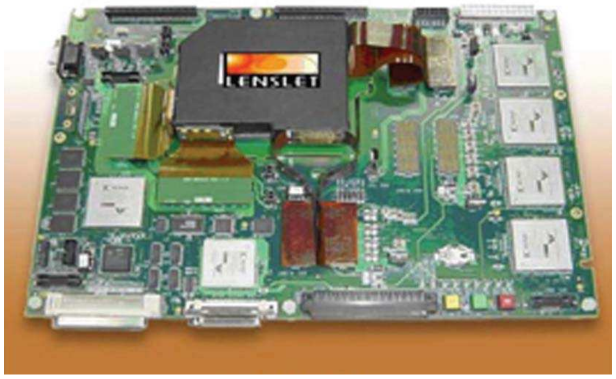

Fig. 2 The *EnLight Alpha*

The FFT results reported hereafter were obtained using both an implementation on the actual *EnLight*™ *Alpha* hardware, and a (bit-exact, cycle-exact) emulator that is included in the *EnLight*™256 software development platform. Note that the software interface between a workstation and the *EnLight Alpha* processing board has a hierarchical structure. Specifically, higher level programming languages such as FORTRAN, C, or MATLAB generate Hardware Description Language (HDL) files and bit-streams via the use of Xilinx Sysgen blocks of the MATLAB/Simulink module. These, in turn are used to program the FPGAs that control the optical core.

Our FFT tests were carried out in the framework of matched filter computations that occur in anti-submarine warfare tracking applications. For illustrative purposes, we consider a set of 33 filter banks (32 Doppler cells and plus one target echo), with complex data sequences of length 80 K samples [13]. To enable implementation on the *EnLight Alpha*, a well known factorization [14] of the (zero-padded) data vectors into multidimensional arrays was applied until a segment size fitting the core was obtained. That segment was then transformed using a DFT (since the actual matrix vector multiplication is of complexity one on the optical processor). The FFT was expanded backward via twiddle factor implementation on the FPGAs. The following results were achieved: on a dual Intel Xeon running at 2 GHz, we required 9,626 ms. This time was reduced to only 1.41 ms on the *Alpha*, which corresponds to a speed-up factor of over 13,000! The estimated performance on the *EnLight*256 was 0.17 ms. In terms of accuracy, while substantial quantization noise was observed for low-magnitude spectral values (we did not trade precision for clock cycles), all peaks of relevance to the detection and tracking process matched accurately those produced by floating point FORTRAN or MATLAB computations on the workstation, as shown in Fig. 3.

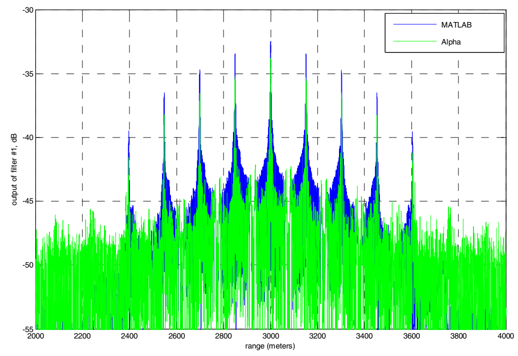

Fig. 3 Comparison of accuracy: matched filter calculation on the EnLight Alpha hardware. Filter output (green for Alpha, blue for MATLAB) is plotted versus target range (in m).

III. IBM CELL MULTI-SIMD PROCESSOR

In 2000, IBM, Sony, and Toshiba formed an alliance to develop a revolutionary, new computational platform for game and video applications and for numerically intensive tasks in science and engineering. The Cell Broadband Engine™ architecture [15] is the extraordinary resulting product of five years of a sustained, intensive R&D effort. In its latest (third generation) release, the PXC 8i includes one multithreaded 64-bit PowerPC processor element (PPE) with two levels of globally coherent cache, and eight synergistic processor elements (SPE). Each SPE consists of a processor (SPU) designed for streaming workloads, local memory, and a globally coherent DMA engine. Emphasis is on SIMD processing. An integrated high-bandwidth element interconnect bus (EIB) connects the nine processors and their ports to external memory and to system I/O.

The key design parameters of the PXC 8i are as follows. Each SPE comprises a 28.75 GFLOP (single precision) synergistic processing unit (SPU), a 256 KB local store memory, a 2×25.6 GB/s memory flow controller (MFC) with a non-blocking DMA engine, and 128 registers, each 128-bit wide to enable SIMD-type exploitation of data parallelism. It is designed to dissipate 4W at a 4 GHz clock rate. We, however, run the SPEs at 3.2 GHz. The EIB provides 2 × 25.6 GB/s memory bandwidth to each SPE local store, and allows external communication (I/O) up to 35 GB/s (out), 25 GB/s (in). The PPE has a 64-bit RISC PowerPC architecture, a 32KB L1 cache, a 512 KB L2 cache, and can operate at clock rates in excess of 3 GHz. It includes a vector multimedia extension (VMX) unit. The total power dissipation is estimated nominally around 125W per node (not including system memory and NIC). The total peak throughput exceeds 230 GFLOPS per Cell processor, for a total communication bandwidth above 204.8 GB/s. The Cell architecture is illustrated in Figure 4.

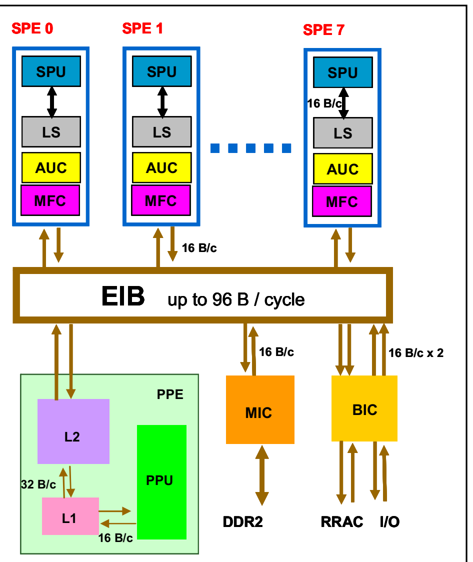

Fig. 4 IBM CELL processor architecture

Exploiting the capabilities of single-instruction-multiple-data stream (SIMD) processors to improve the performance of the FFT has long been of interest to the applied mathematics, signal processing, and computer science communities [16, 14]. Most Cell implementation efforts reported so far have focused on parallelizing a *single* FFT across all synergistic cores (SPEs). We define such a paradigm as *inline vectorization*. It is this approach that has produced the fastest execution time published to date [17]. In contradistinction, we are interested in the case where M 1D data arrays, each of length N, would be Fourier-transformed concurrently by a single SPE core. This would result in $8M$ arrays that could be handled simultaneously by the Cell processor, with each core exploiting its own SIMD capability. We have defined such a paradigm as *transverse vectorization* [18, 19].

Our method, which generalizes a decimation in frequency Cooley-Tukey radix 2 scheme, was implemented in a mixed language framework (FORTRAN 95/2003 and C) using the IBM XLF and XLC compilers for multicore acceleration under Linux. The C components made use of the IBM Cell SDK 3.1, specifically the SPE management library *libspe2* for spawning tasks on the SPU, for managing DMA transfers, and for passing local store (LS) address information to the FORTRAN subroutines. The FORTRAN code was used for the numerically intensive computations. Subroutines were written and compiled using IBM XLF 11.1. Intrinsic SIMD functions including *spu_add* , *spu_sub* , *spu_splats* , *spu_mul* , and specialized data structures for SIMD operations were exploited. These functions benefit from the large number of vector registers available within each core. The FORTRAN code was compiled at the O5 optimization level.

For test purposes we initially used up to 128 data vectors per core, each vector containing 1024 complex samples. The following scenario was adopted. The explicitly vectorized version of our algorithm was implemented using the SIMD data structure VECTOR(REAL(4)) and the SIMD intrinsic functions / instructions available on the SPU. Multiple input vectors of 1024 complex data samples were processed in each SPE core. These complex vectors were de-interleaved, and their real and imaginary components were stored separately. This step was considered necessary since some of the SIMD intrinsics do not yet support complex data types. The data was processed 4 vectors at a time, with a triple-buffering scheme used to hide the DMA latencies.

Timing results for all of the routines were obtained using the SPU decrementers and time base to convert clock cycles into microseconds. The SPU clock was started *before* issuing the first DMA fetch of input data and stopped *after* confirming the final output data was written to main memory. The results were compared with results from a scalar FFT program run on the PPU core and found to be accurate to within floating-point (32-bit) precision.

We now provide a comparison to the latest and best results related to FFT implementation on the Cell that have been published to date in the open literature [17]. For complex data vectors of length 1024, Bader reports a processing time of 5.2 microseconds when using all SPEs of a Cell processor running at 3.2GHz. This outperformed the industry standard FFTW by a factor of two. To compare his processing time to ours, we observe that we need 181 microseconds to process 64 vectors on the 8 SPEs of the Cell. This results in a time of 2.82 microseconds per 1D data vector. We also timed the processing of 128 vectors per SPE (that is a total of 1024 vectors, each of length 1024 complex samples). The average (over the SPEs) time is 2709.75 microseconds, corresponding to 2.64 microseconds per FFT. This is even faster than the result shown above for 64 vectors per Cell, due to the reduced impact of pipeline start-up and shutdown.

IV. THE NVIDIA TESLA PROCESSOR

The recent emergence of the NVIDIA Tesla general purpose graphical processing unit (GP GPU), which exploits the NVIDIA CUDA architecture [20], is beginning to revolutionize the area of high-performance computing. The Tesla is built in terms of an array of 30 Streaming Multiprocessors (SMs). Each SM consists of eight scalar processor cores. This results in a total of 240 cores for the Tesla. In addition, an SM includes two special units for transcendental functions, a multithreaded instruction unit, and 16 KB of shared memory. There are 4 GB of global RAM available to the GPU, and its clock runs at 1.296 GHz. The architecture of the Tesla is shown in Fig. 5.

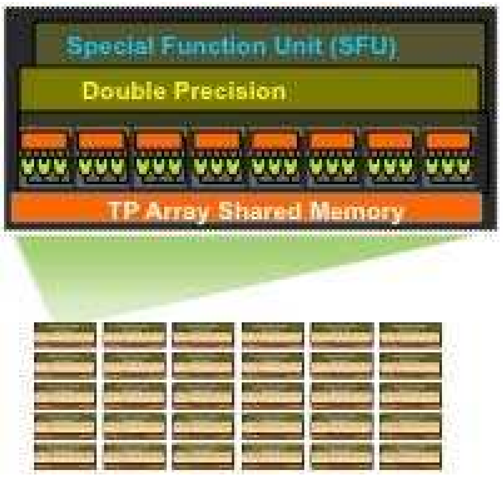

Fig. 5 Architecture of the Tesla GPU (courtesy NVIDIA).

From a purely computational perspective, the fundamental underlying paradigm is the concept of scalar thread. Threads are grouped in blocks that execute concurrently on one SM with zero overhead [20]. Massive parallelism is achieved in terms of a kernel grid comprising the thread blocks. As any thread block terminates, a new block is launched on the first available (vacated) multiprocessor.

It is important to realize that the concept of the SIMD parallelism does not apply directly to the Tesla. Rather, the CUDA architecture exploits the single-instruction-multiple-thread (SIMT) concept [20]. Each thread block executing on an SM is partitioned into groups of 32 threads (called *warps*) that are scheduled by the SIMT unit to execute concurrently. Under SIMT, each thread from a warp is assigned to one of the scalar processor cores belonging to the SM. Threads composing the warp start at the same program address, but are then nominally free to branch and execute independently. At every instruction select time, the SIMT unit selects a warp that is ready to execute and issues the next instruction to the active threads in the warp. If threads in a warp diverge via data-dependent conditional branching, the warp serially executes each branch path taken, while disabling threads that are not on that path. Finally, when all paths complete, the threads converge back to the same execution path [20]. Since a warp executes one common instruction at a time, the highest efficiency is achieved when all 32 threads in a warp agree on their execution path. This is precisely what happens under our transverse vectorization construct.

The specific computational platform considered for this study is the NVIDIA Tesla C1060 GPU operating in conjunction with an Intel Xeon E5530 2.4 GHz CPU that accesses 6 GB of ECC DDR3 1066 MHz RAM. The operating system is the 64-bit edition of Windows XP. Our algorithms are programmed in CUDA FORTRAN using a compiler provided by the Portland Group Inc [21].

Our implementation comprises two components. A code that runs on the host CPU and a kernel that runs on the GPU and is invoked by the host. In practice we run a set of identical kernels on a one-dimensional grid partitioned into thread blocks. These blocks get assigned to different SMs of the GPU as scheduled by the hardware. Because of the latencies associated with data retrieval from the GPU's global memory, there is a strong incentive to maximally exploit all banks of shared memory, as well as the constant memory. The abstract-level outline of our approach is as follows. Let

$$y_m = \sum_{n=0}^{n=N-1} x_n W_N^{nm} \quad \text{where } \dim \mathbf{x} = \dim \mathbf{y} = N$$

denote the DFT of a vector $\mathbf{x}$. If we factor the vector length N as $N = N_1 \text{ x } N_2$, and let $n = n_1 + n_2 N_1$ and $m = m_1 N_2 + m_2$, then we obtain the following DFT factorization that exploits the periodicity of the exponential matrix $\mathbf{W}_N$

$$y_{m_2,m_1} = \sum_{n_1=0}^{n_1=N_1-1} \sum_{n_2=0}^{n_2=N_2-1} x_{n_1,n_2} W_{N_2}^{n_2 m_2} W_N^{n_1 m_2} W_{N_1}^{n_1 m_1}$$

It is easy to recognize that the above expression represents N_1 concurrent N_2 – point Fourier transforms along the rows of $\mathbf{X}$ (innermost summation), followed by multiplication by the "twiddle" factors, leading into N_2 concurrent N_1 – point Fourier transforms along the columns. This allows us, by judiciously choosing N_1 and N_2, not only to assign stride one thread indices, but also to optimally exploit the shared memory. We use a radix-8 Stockham – based FFT scheme [22], where we have implemented loop unrolling. Moreover, note that, even though complex numbers are supported both by NVIDIA's CUDA C and PGI's CUDA FORTRAN compilers, we split the real and imaginary components to achieve better alignment in memory fetches.

To benchmark our performance, we have also implemented the cuFFT library (for NVIDIA's CUDA architecture, version 2.3) on the Xeon / Tesla C1060 system. The driver program was written in C and compiled using Microsoft's Visual C++ Express Edition compiler. Preliminary FFT timing runs suggest that processing performance in excess of 190 GFLOPS can be achieved for complex data sample vectors of length 512 points, when processed in batches of 16384. For vector lengths of 4096 points, the performance of cuFFT drops to under 138 GFLOPS for batches of 2048. Our vectorized Stockham scheme exceeds 87 GFLOPS for this vector length and batch size. The fastest results previously reported [23] near 100 GFLOPS. Note that our algorithm has not yet been fully optimized (neither in terms of stride balance, nor in terms of optimal memory assignments), so that substantial room for progress exists. All previous times refer to device execution only.

V. CONCLUSIONS

It is widely recognized that the FFT algorithm plays a central role in Cognitive Radio. This paper has attempted to illustrate the considerable gain in computational throughput that can be achieved from emerging processor technology, including the IBM Cell multi-SIMD processor, the NVIDIA Tesla GPU, and the EnLight digital optical core device. In CR, the ultimate goal is clearly the design of *adaptive* FFT schemes. These can not only exploit the advances reported herein, but also incorporate unconventional architectural advances in processors such as the Coherent Logix HyperX [24]. This ultralow power processor (a few pico-Joules per floating point operation) offers real time compute and I/O capabilities, and is the subject of a companion paper [25].

Regarding GPU computing, we note that the recent release by PGI of the high-performance CUDA FORTRAN compiler for the NVIDIA Tesla opens, via mixed language (C and FORTRAN) programming an optimal framework for ultra fast future implementations. Such a framework would fully exploit the intrinsic array language, compiler optimization and numerical capabilities of FORTRAN in conjunction with the DMA and system capabilities of C.

ACKNOWLEDGMENT

Portions of this work were 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. R. van Nee, *OFDM for Wireless Multimedia Communications*, Artech House (1999).
2. Q. Zhang, A. Kokkeler, and G. Smit, “An efficient FFT for OFDM based Cognitive Radio on a reconfigurable architecture”, *IEEE International Conference on Communications*, pp. 6522-6526 (2007).
3. B. Farhang-Boroujeny and R. Kempter, “Multicarrier communication techniques for spectrum sensing and communication in cognitive radios”, *IEEE Communications Magazine*, **46**(4), 80-85 (2008).
4. S. Haykin, D. Thomson, and J. Reed, “Spectrum sensing for Cognitive Radio”, *Proceedings of the IEEE*, **97**(5), 849-877 (2009).
5. S. Sampei, S. Miyamoto, and S. Fbi, “Spectrum loading type dynamic spectrum allocation technique for Cognitive Radio systems”, *Second IEEE International Conference on Cognitive Radio Oriented Wireless Networks and Communications*, pp. 535-539 (2007).
6. C. Cordeiro, M. Ghosh, D. Cavalcanti, and K. Challapali, “Spectrum sensing for dynamic spectrum access of TV bands”, *Second IEEE International Conference on Cognitive Radio Oriented Wireless Networks and Communications*, pp. 225-233 (2007).
7. C. Liu, Y. Zeng, and S. Attallah, “Max-to-Mean Ratio detection for Cognitive Radio”, *IEEE 67th Vehicular Technology Conference*, pp. 1959-1963 (2008).
8. R. Lopez-Valcarce and C. Mosquera, “Maximum likelihood SNR estimation for asynchronous oversampled OFDM signals”, *IEEE 9th workshop on Signal Processing Advances in Wireless Communications*, pp. 26-30 (2008).
9. Q. Zhang, A. Kokkeler, and G. Smit, “An efficient multi-resolution spectrum sensing method for Cognitive Radio”, *Third International Conference on Communications and Networking in China*, pp. 1164-1167 (2008).
10. S. Yoshizawa, K. Nishi, and Y. Miyanaga, “Reconfigurable two-dimensional pipeline FFT processor in OFDM Cognitive Radio systems”, *IEEE International Symposium on Circuits and Systems*, pp. 1248-1251 (2008).
11. J. Barhen, N. Imam, M. Vose, A. Averbuch, and M. Wardlaw, “Underwater threat source localization: processing sensor network TDOAs with a terascale optical core device”, *NATO Book Series*, **D 8**, 56-68, IOS Press (2007).
12. J. Barhen, N. Toomarian and A. Fijany, “High precision computing with charge-domain devices and a pseudo-spectral method therefor: *Part* II”, *U.S. Patent No. 5,680,515* (1997).
13. N. Imam and J. Barhen, “Acoustic source localization via time differences of arrival estimation for distributed sensor networks using terascale optical core devices”, *Journal of Sensors*, **2009**, ID187916 (2010).
14. C. Van Loan, *Computational Frameworks for the Fast Fourier Transform*, SIAM Press (1992).
15. J. A. Kahle *et al*, “Introduction to the Cell multi-processor”, *IBM Journal of Research and Development*, **49**(4-5), 589-604 (2005).
16. R. W. Hockney and C. R. Jesshope, *Parallel Computers*, Adam Hilger (1981).
17. D.A. Bader, V. Agarwal, and S. Kang, “Computing discrete transforms on the Cell Broadband Engine”, *Parallel Computing*, **35**(3), 119-137 (2009).
18. J. Barhen *et al*, “Vector-sensor array algorithms for advanced multicore processors”, *US Navy Journal of Underwater Acoustics* (in press, 2010).
19. J. Barhen, T. Humble, P. Mitra, and M. Traweek, “Multi FFT vectorization for the Cell multicore processor”, in *Frontiers in GPU, Multi and Many Core Processing,* IEEE /ACM Workshop Proceedings, IEEE (in press, 2010).
20. Anonymous, *NVIDIA CUDA Programming Guide*, NVIDIA Corporation (August 2009).
21. M. Wolfe *et al*, *CUDA FORTRAN Programming Guide and Reference*, The Portland Group, Inc (November 2009).
22. P. Swarztrauber, “FFT algorithms for vector computers”, *Parallel Computing*, **1**, 45-63 (1984).
23. N. Govindaraju *et al*, “High performance discrete Fourier transforms on graphics processors”, in *Proceedings Supercomputing 2008 Conference* (November 2008).
24. M. Stolka (Coherent Logix), personal communication; see also: www.coherentlogix.com
25. T. Humble, P. Mitra, J. Barhen and B. Schleck, “Real-time spatio-temporal twice whitening for MIMO energy detectors”, in *Proceedings CrownCom 2010* (in press, 2010).