A HIGH SPEED WIRELESS LAN RADIO RECEIVER IN SOFTWARE

Mohamed Ismail, Toshiba Research Europe Ltd., Bristol, UK,
mohamed.ismail@toshiba-trel.com

ABSTRACT

As wireless LAN speeds increase on the back of ever more sophisticated processing algorithms the commensurate need for greater processing power becomes a challenge. The mass-market computing world has recently seen a move away from ever increasing clock speeds to multi-core platforms. Thus, with potentially redundant computing power the question of applying it to realise the core functions of a high-speed wireless LAN was investigated. Three functions central to any future high-speed wireless LAN demanding significant computational resources were identified, Fast Fourier Transform (FFT), matrix decomposition and channel decoding (error correction decoding). These functions were implemented on a standard server platform with an Intel Xeon™ processor with execution times benchmarked for different solutions.

1. INTRODUCTION

The new generation of wireless standards are necessarily more complex than those before consequently development time and cost increase with each new standard. Software development tools and methods are mature and well understood by most engineers but more importantly well established mechanisms exist for code reuse and reconfiguration. Add to this the evolutionary nature of standards, e.g. GSM, GPRS, EDGE, then a need for a versatile platform becomes clear; versatility that is difficult to realise in traditional hardware based solutions but readily available in software solutions.

Commercialisation of software radio solutions has been pioneered by Vanu Inc. particularly in the area of cellular technology. In [1] a description of Vanu's software radio technology highlighting benefits of software radio over software defined radio is given.

Desktop and mobile general purpose computing platforms are managing to cram in ever more processing power and storage capacity with the latest offerings being multi-core, multi-gigabyte, wireless LAN devices. It is this coming together of significant computing power and emerging high-speed (e.g. IEEE 802.11n) wireless technology on a single platform which has motivated the investigation into the possibility of using general purpose hardware and software for realising software radio architectures.

The proposed IEEE 802.11n high throughput standard points the way for algorithms necessary for high-speed wireless communication. Highlighted below are three areas believed to be key in achieving robust high throughput and being computationally demanding.

Modern wireless standards are reliant on Multiple-Input-Multiple-Output (MIMO) radio technology for reliability, in the form of spatial diversity, and throughput, in the form of spatial multiplexing. A common requirement in MIMO enabled receivers is the need to perform complex matrix inversion or decomposition. The number of transmit and receive antenna determining the size of the matrix. Such a computation would be required for every transmission channel employed by the system.

Orthogonal Frequency Division Multiplexing (OFDM) is the basis of many high-speed wireless standards whereby the available spectrum is represented as a bank of overlapping (though not interfering) frequencies. A key operation used in translating data from the time domain to the frequency domain is the Fast Fourier Transform (FFT). An attractive hardware solution for realising FFTs is by way of the butterfly operation [2] since it can be done in parallel. However, for a single processor such a solution may not be optimum.

Forward Error Correction (FEC) is deployed in wireless communication systems to decrease the probability of receiving an incorrect transmission. The most widespread FEC scheme is convolutional coding due to its simplicity in encoding and availability of optimal decoding solutions. The Viterbi algorithm gives the Maximum-Likelihood (ML) symbol sequence transmitted based on the received data and the trellis of permissible state transitions as defined by the convolutional code. A principal operation in the Viterbi algorithm is an Add-Compare-Select (ACS) operation for each output symbol with the number of comparisons being 2^K where K is the constraint length of the convolutional code. One method of improving throughput for a Viterbi decoder would be to introduce parallelism in carrying out ACS operations. With a general purpose processor, lacking a high degree of parallelism, an alternative decoding algorithm is necessary such as sequential decoding [5].

The aforementioned three functions were chosen for implementation on a Commercial-Off-The-Shelf (COTS) PC using readily available software development tools. Execution time was measured for various implementations with a view to benchmarking likely performance.

Section 2 describes the platform and software environment used to implement and develop the algorithms. Section 3 describes implementation of the FFT algorithm and the results obtained. Section 4 outlines the MIMO decoder problem deriving a direct-form solution and compares the performance against standard matrix decomposition algorithms. Section 5 details the channel decoding problem comparing the performance of a sequential decoding solution against a Viterbi solution.

2. PLATFORM CONFIGURATION

Testing was conducted using an NEC 120Re-1 server running Mandriva™ Linux 2006 (kernel 2.6.12-12) configured with one single core 3.2 GHz Intel Xeon™ processor (with an 800 MHz front side bus) and 1 GB of RAM. All code was written using C/C++ and compiled using the Intel C compiler (9.1) with full optimization (-O3 -xN) linking to the Intel Maths Kernel Library (8.1) where required.

3. FAST FOURIER TRANSFORM

Traditionally, designers aim to minimise the number of operations required to perform an FFT operation optimised for a particular FFT size and precision. However, with current (and future) microprocessors this criterion is far less important due to their excellent, in terms of throughput, floating point capability and numerical precision. A more significant factor affecting performance is the interaction between main memory and processor. Addressing this bottleneck [3] describes an FFT solution optimised to run on a modern microprocessor architecture showing how by careful selection of parameters significant improvements in throughput can be obtained. For this study the Intel Maths Kernel Library (MKL™) was chosen to provide FFT routines optimised for the Intel family of processors.

Four possible combinations of precision and type were evaluated:

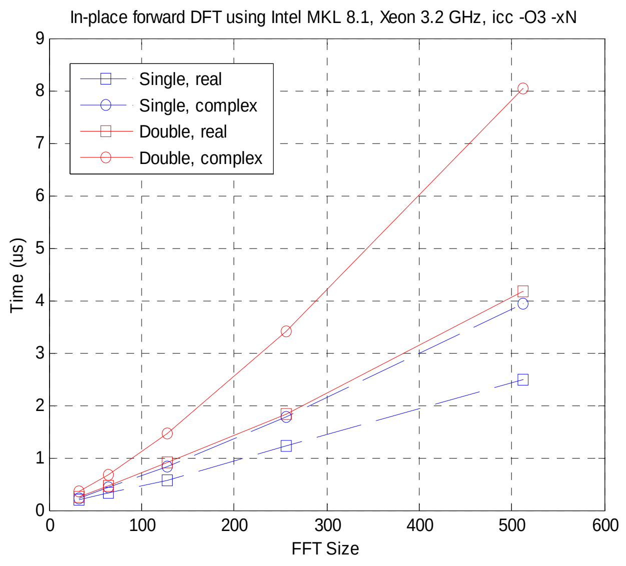

Figure 1 - In-place forward DFT

- single, real
- single, complex
- double, real
- double, complex

Where single precision was represented by 32 bit floating point values and double precision was represented by 64 bit floating point values.

An in-place FFT was performed for FFT sizes ranging from 32 point up to 512 point in steps of powers of two. The computation time was averaged over one million iterations with the results shown in Fig. 1.

A 128 point single precision complex FFT took approximately 0.9 µs to calculate which translates to over 142 MegaSamples/s and is well within the 4 µs time frame permitted for processing a received packet in the IEEE 802.11n proposed standard.

4. COMPLEX MATRIX DECOMPOSITION

The received signal as seen by a multi-carrier MIMO receiver maybe described by the following equation:

$$y = Hx + n \qquad (1)$$

where, $y \in \mathbb{C}^{N_R \times 1}$ is the vector of received samples, $H \in \mathbb{C}^{N_R \times N_T}$ is the channel matrix, $x \in \mathbb{S}^{N_T \times 1}$ is the vector of transmit symbols and $n \in \mathbb{C}^{N_R \times 1}$ is a vector of AWGN samples with variance σ^2, with N_T being the number of transmit streams and N_R the number of received streams.

MIMO detection is the process of forming an estimate of the transmitted symbols, $\hat{x}$, based on the received symbols and the channel matrix. One common formulation of symbol estimates is using the Minimum Mean Squared Error (MMSE) technique, given by:

$$\hat{x} = \left(H^H H + \sigma^2 I_{N_T}\right)^{-1} H^H y \qquad (2)$$

where, I_{N_T} is an $N_T x N_T$ identity matrix.

The term in brackets requiring inversion is computationally the most challenging part. For a two-transmit antenna two-receive antenna system (2x2) the term in brackets is a 2x2 matrix. Let A represent the term in brackets, then A and H^H can be written as follows:

$$A = \begin{bmatrix} a_{11} & a_{12} + jb_{12} \\ a_{12} - jb_{12} & a_{22} \end{bmatrix}, \qquad (3)$$

$$H^H = \begin{bmatrix} c_{11} + jd_{11} & c_{12} + jd_{12} \\ c_{21} + jd_{21} & c_{22} + jd_{22} \end{bmatrix}$$

The four terms of the product $A^{-1}H^H$, excluding for clarity the determinant of A, can be written as:

$$\begin{gathered} -a_{22}c_{11} + a_{12}c_{12} + b_{12}d_{12} + ja_{22}d_{11} - ja_{12}d_{11} + jb_{22}c_{12} \\ -a_{22}c_{21} + a_{12}c_{22} + b_{12}d_{22} + ja_{22}d_{21} - ja_{12}d_{22} + jb_{12}c_{22} \\ a_{21}c_{11} - b_{21}d_{11} - a_{11}c_{12} - ja_{21}d_{11} - jb_{21}c_{11} + ja_{11}d_{12} \\ a_{21}c_{21} - b_{21}d_{21} - a_{11}c_{22} - ja_{21}d_{21} - jb_{21}c_{21} + ja_{11}d_{22} \end{gathered} \qquad (4)$$

The MMSE estimate formed from (4) was coded in C++ as an explicit multiply and sum for each of the elements of the estimated vector $\hat{x}$. In comparison, three well established matrix inversion techniques [4] namely; the Givens, Fast Givens and the Modified Gram-Schmidt (MGS) method were implemented. These implementations were compared with the QR factorisation (sgeqrf) function provided within the Intel MKL

Table 1 summarises results obtained for real and complex valued single precision (32 bit), 2x2 matrix inversion/decomposition running on the test platform.

Table 1 - 2x2 Real and complex matrix inversion/decomposition times

	Real Valued				Complex Valued	
Method	Givens	Fast Givens	MGS	Intel MKL	Fast Givens	Direct
Time (µs)	0.09	0.08	0.08	6.7	0.16	0.05

Although the theoretical number of floating point operations (flops) [4] required for the real-valued Givens, Fast Givens and Modified Gram-Schmidt algorithms are 16, 10.7 and 16, respectively, execution of which on a nominal 3 GHz floating point unit would take no more than 6 ns the observed absolute values are significantly greater. This was due to main memory accesses which typically incur many clock cycles. Using the direct-form method gave the best result as it had minimum overheads incurring only the cost of floating point operations and minimal memory access. The Intel MKL solution is significantly slower than the other methods the reason, as given by Intel, was due to the MKL library being designed for large matrix sizes.

5. OPTIMAL CHANNEL CODING

Convolutional coding is the method of choice for reducing the probability of erroneous transmission of data over a noisy communications channel. Viterbi decoding has become the de-facto algorithm used in the decoding of convolutional codes in recent times. Such widespread use of the Viterbi algorithm, despite suffering from high complexity for convolutional codes with a long constraint length, has been possible due to efficient hardware solutions.

In contrast to the computational complexity of the Viterbi algorithm, sequential decoding is well known for having a computational complexity independent of code constraint length [5]. Although suboptimal in performance, sequential decoding can achieve a desired bit error probability when a sufficiently large constraint length is taken for the convolutional code. Unlike the Viterbi algorithm which locates the best codeword by testing, exhaustively, all possibilities, sequential decoding tests only a subset of all possible code words based on some metric. Commonly, the metric used is a function of the received data and thus the sequential decoder performance is a function of the noise level present in the received signal.

Although sequential decoding offers a less computational and therefore potentially quicker decoding solution in the classical form it suffers from a number of technical problems:

- Not Maximum-Likelihood decoding
- Unbounded complexity (can potentially exceed Viterbi)
- Hard decision output

A number of techniques have been proposed to address one or two of these problems, the most comprehensive solution being detailed in [6]. The cited paper details a Maximum-Likelihood Soft Decision (MLSD) sequential decoding algorithm for use with binary convolutional codes which has a bounded complexity. The authors compare the performance and complexity of a Viterbi algorithm (VA) and a standard sequential decoding (SA) algorithm (stack algorithm with a Fano metric) with the MLSD algorithm. The results from [6] show the maximum number of metric computations required by the MLSD algorithm not exceeding the Viterbi algorithm and the average number of metric computations of the MLSD algorithm being 1.6% of the Viterbi algorithm at a signal-to-noise ratio of 7dB.

The MLSD algorithm was implemented in C using a balanced-tree data structure [7] to efficiently handle stack manipulation.

An Additive White Gaussian Noise (AWGN) channel operating at a Signal-to-Noise Ratio (SNR) of 20 dB was used with 1000 individual runs being averaged to derive the timing values. A ½ rate convolutional code with a constraint length of seven and generator polynomials (147,135) with a block size of 216 bits was used for all simulations. Execution time of the MLSD algorithm was compared to a C implementation of the Viterbi algorithm.

Execution time for the two algorithms, MLSD and Viterbi, was measured to be 0.2 ms and 0.6 ms respectively. Although the number of metric computations required by the MLSD algorithm is significantly lower than the Viterbi algorithm the implied gain in execution time is not observed. Once again the cost of main memory access impinges upon the execution time. Nonetheless, the MLSD algorithm gives significant improvements in execution time at good signal-to-noise ratios.

7. CONCLUSION

An analysis on the feasibility of using general purpose COTS computing architecture for the purpose of implementing functions required in a high-speed wireless LAN highlighted three areas of comparatively high computational cost. Fast Fourier Transforms, complex matrix decomposition and optimal channel coding were implemented in C/C++ on a COTS server platform compiled using standard software tools. A 128 point single precision complex FFT, as would be used in an IEEE 802.11n system, required 0.9 µs. A carefully designed implementation of a 2x2 complex matrix decomposition proved to be over two-thirds faster than using a Fast Givens matrix decomposition algorithm. A sequential decoding solution was shown to execute in one third the time of a Viterbi algorithm when decoding a convolutional code. These results demonstrate the possibility of using general purpose computing solutions that are widely available to undertake computationally intense functions necessary for high-speed wireless LANs.

8 REFERENCES

[1] J. Chapin, "Overview of Software Radio", Vanu Inc, http://www.vanu.com/resources/whitepapers/tech-overview-whitepaper-3-6-02.pdf.

[2] J. W. Cooley and J.W. Tukey, "*An Algorithm for machine computation of Fourierseries*", Math. Comput., 19, 297-301, 1965.

[3] M. Frigo, S.G. Johnson, "*FFTW: An Adaptive Software Architecture for the FFT*", ICASSP 1998, vol.3 pp. 1381-1384.

[4] G.H. Golub, C.F. Van Loan, "Matrix Computations", 3rd edition, Johns Hopkins University Press, 1996, ISBN 0-8018-5414-8.

[5] S. Lin, D.J. Jr. Costello, "Error Control Coding: Fundamentals and Applications", Prentice-Hall, 1983, ISBN 0-13-283796-X.

[6] Han Y.S, Chan P, Wu H, "A Maximum-Likelihood Soft-Decision Sequential Decoding Algorithm for Binary Convolutional Codes", IEEE Trans. Comms., vol. 50, no.2, Feb. 2002.

[7] T.H. Cormen, C.E. Leiserson and R.L. Rivest, "*Introduction to Algorithms*", Cambridge, MA:MIT Press 1991.