DSP Implementation of a Bit Loading Algorithm for Adaptive Wireless Multicarrier Transceivers

Martin Cudnoch, Alexander M. Wyglinski, and Fabrice Labeau

Abstract

In this paper, we present a proof-of-concept, fixed-point, DSP hardware implementation of an adaptive bit loading algorithm that is designed for wireless multicarrier transceivers. Adaptive bit loading is used to enhance the performance of multicarrier transceivers by tailoring the subcarrier signal constellations to the channel conditions, which can vary across the subcarriers. Since most bit loading algorithms possess a high computational cost and are unable to cope with rapid variations of wireless channels, they are seldom used in present wireless standards. To prove that adaptive bit loading is feasible for wireless transceivers, our work focuses on the implementation of a known bit loading algorithm that can quickly search for the final bit allocation in an iterative manner. The goal of this algorithm is to yield the largest-possible throughput while satisfying a mean BER constraint. The performance of the hardware implementation operating in time-varying channel conditions is studied in terms of the overall throughput. Furthermore, the robustness of the hardware implementation is evaluated, relative to sudden changes in the channel that interrupts the run of the algorithm. Real-time operations and fixed-point representation issues are included in the discussion. Additionally, we propose a modified algorithm implementation that is more robust to channel variations.

This work was presented in parts at the *2006 IEEE Wireless Communications and Networking Conference* (WCNC), Las Vegas, Nevada, USA.

This work was generously supported by the Natural Science and Engineering Research Council of Canada (NSERC) and Le Fonds “Nature et Technologies” du Quebec (NATEQ).

Martin Cudnoch is with SR Telecom Inc., Montreal, Quebec, Canada.

Alexander M. Wyglinski is with the Information and Telecommunication Technology Center, The University of Kansas, Lawrence, Kansas, USA.

Fabrice Labeau is with the Department of Electrical and Computer Engineering, McGill University, 3480 University Street, Room ENGMC-754, Montreal, Quebec, Canada, H3A 2A7, Tel.: +1 514 398-7140, Fax: +1 514 398-4470, E-mail: fabrice.labeau@mcgill.ca.

Index Terms

Adaptive wireless transceivers, multicarrier modulation, bit loading, DSP implementation.

I. INTRODUCTION

Deployed in a wide range of high data rate transmission applications, e.g., digital subscriber line (DSL) modems [1] and wireless local area network (WLAN) systems [2, 3], multicarrier modulation (MCM) is the method of choice for the next generations of wireless standards [4, 5]. The primary advantage of MCM over other data transmission schemes is its ability to transform a frequency-selective fading channel into a collection of approximately-flat subchannels, which yields simpler equalizer designs at the receiver. Another advantage of MCM is its ability to *tailor* the subcarrier operating parameters to the communications channel, thereby enhancing the overall performance of the system.

One candidate technology that can facilitate the tailoring of MCM operating parameters is *cognitive radio* [6, 7], which is a class of software-defined radio (SDR) that can rapidly reconfigure and adapt its baseband radio functions to changing environmental conditions, as well as both user and network requirements. Since the baseband radio operations are implemented in software, such as modulation and coding, cognitive radios can employ and alter these functions quickly and with ease. Thus, cognitive radios can simultaneously adjust one or more MCM parameters in real-time using information acquired during operation.

Although several subcarrier parameters are available for tailoring, one of the most popular choices is subcarrier signal constellation size. The process of tailoring this parameter, called *adaptive bit loading* [8–16], involves an algorithm that adjusts the number of bits per subcarrier per symbol epoch according to the channel conditions. Moreover, the algorithm does not constrain the sizes to be equal across all subcarriers. Even though there exists several MCM implementations [2, 3] that also vary the subcarrier signal constellation size, these implementations do constrain the sizes to be identical despite the possibility that the channel distortion is non-uniform across the subcarriers. As a result, adaptive bit loading has the potential to achieve data transmissions that are very spectrally-efficient.

There have been several adaptive bit loading algorithms proposed in the literature designed to achieve spectrally-efficient communications [8–10, 14]. However, most of these algorithms either possess a very high computational complexity or yield a final solution that is far from the optimal allocation. Nevertheless, there exist several algorithms in the literature designed to quickly converge to a solution that is near-optimal [12, 17]. In particular, Wyglinski, Labeau, and Kabal proposed an algorithm that iteratively searches for a final bit allocation that is close to the optimal solution [12]. To highlight the advantages of their algorithm, they compared the performance of their proposed algorithm with several other algorithms found in the literature using computer simulations. Although the proposed algorithm performed relatively well, it is still necessary to determine its feasibility in a modern communications system, such as software-defined radios [6, 18] or cognitive radios [7]. Furthermore, the ability of the proposed algorithm to adapt to time-varying channel conditions, and its impact on system performance and resources, were never assessed.

In this paper, we present a real-time DSP hardware implementation of the adaptive bit loading algorithm proposed in [12]. Furthermore, we evaluate the performance of the bit loading hardware implementation when operating in a time-varying, frequency-selective fading channel. Finally, we propose a new initialization process for the implementation that allows the algorithm to converge faster to the final solution. The rest of this paper is organized as follows: Section II describes the framework of an adaptive multicarrier transceiver and the general concepts of adaptive bit loading. The details of the bit loading algorithm implemented in this work are presented in Section III. The implementation of the bit loading algorithm in DSP hardware is discussed in Section IV. Several experimental results of the bit loading algorithm obtained from the DSP hardware implementation are presented in Section V, followed by some concluding remarks in Section VI.

II. Adaptive Multicarrier Transceiver

A. *Multicarrier Framework*

The general setup for an MCM transceiver with adaptive bit loading is used [19]. At the transmitter, a high-rate input stream $x(n)$ is demultiplexed into N separate streams with stream i having b_i bits per symbol epoch. The value of b_i is determined by the adaptive bit loading algorithm, which uses the subcarrier SNR values γ_i, $i = 0, \ldots, N-1$, to compute the subcarrier BER [11, 12] or the channel capacity [9]. The subcarrier SNR values are computed from the channel state information (CSI) provided by the data-aided channel estimator at the receiver. In this paper, we consider the downlink of the general MCM transceiver, with adaptive bit loading performed solely at the transmitter.

B. Bit Loading

The bit loading algorithm uses the channel estimates to determine an appropriate bit allocation across all the subcarriers. As a result, each subcarrier is allocated a different number of bits per symbol epoch. To evaluate the quality of an allocation, the algorithm employs an objective function that provides a quantitative metric in the search for an appropriate allocation. Furthermore, constraints can be applied to the algorithm to ensure that the allocation stays within a desired range of values. In this work, the objective function is the overall system throughput, while the constraint is a limit on the mean bit error rate (BER). Thus the goal of the algorithm is to maximize this objective function while obeying this constraint. Mathematically, this can be summarized by the following equation:

$$\text{maximize} \;\; R = b_{TOT} \;\; \text{bits/symbol} \quad \text{s.t.} \quad \bar{P} = \frac{\displaystyle\sum_{i=1}^{N} b_i P_i(b_i, \gamma_i)}{\displaystyle\sum_{i=1}^{N} P_i(b_i, \gamma_i)} \leq P_{TH} \tag{1}$$

where b_{TOT} is the total bit allocation obtained by summing up the throughput b_i across the N subcarriers such that $b_{TOT} = \sum_{i=1}^{N} b_i$. The maximization is subject to the constraint that the mean probability of bit error, $\bar{P}$, is below the limit P_{TH}. Note that the bandwidth of each subcarrier is assumed to be narrow enough that it can be considered as spectrally flat and, therefore, dependent on the signal-to-noise ratio (SNR) of the subcarrier. This assumption will be held for the remainder of this work.

In a wireless environment, bit allocation algorithms have the additional requirement of adapting to a time-varying channel. Therefore, an adaptive bit loading algorithm must converge to a solution within a short period of time. This amount of time will depend on the type of channel that the system is operating in. For instance, an outdoor wireless channel for a mobile system will have tighter constraints than an indoor wireless channel. In the next section, the adaptive bit loading algorithm is presented. Furthermore, several initialization schemes designed to help the algorithm start its search for the best-possible allocation, are also described.

III. An Efficient Bit Loading Algorithm

A. Algorithm Definition

Since the bit loading algorithm has to be implemented as a real-time module, a low computational complexity is of paramount importance. Thus, the optimal bit loading algorithms found in the literature [14], [20], while offering the best-possible throughput, are too slow to be used. Thus, a sub-optimal solution must be considered, where the algorithm converges to a near-optimal throughput reasonably fast.

The bit loading algorithm chosen for the DSP hardware implementation was first presented by Wyglinski, Labeau and Kabal in [12]. When comparing the proposed algorithm to two algorithms found in the literature [9, 20][1], the computation time of the proposed algorithm is less than both cases, converging to a solution in 6.6% and 54.4% less time respectively. Moreover, when comparing the proposed algorithm to the optimal solution, the difference in throughput is negligible (less than 1% under the conditions described in [12]). As such, it may be considered to be near-optimal.

The objective of the proposed algorithm is to maximize the throughput of the system while keeping the average bit error rate $\bar{P}$ below a threshold P_{TH}, as described in (1). This is done by introducing the concept of a peak BER limit, $\widehat{P}$. Each subcarrier i, where $i \in \mathcal{C}$[2], will use the largest allowed signal constellation $b_i \in \mathcal{B}$ for which the BER, $P_i(b_i, \gamma_i)$ is below $\widehat{P}$.

In the first step, the value $P_i(b, \gamma_i)$ is obtained for each available constellation , by measuring the SNR of each subcarrier, γ_i. The bandwidth of each subcarrier is narrow enough that it can be approximated by closed form expressions derived for the AWGN case [21]. After setting $\widehat{P}$ to an initial value, $\bar{P}$ is calculated. If $\bar{P}$ is below P_{TH}, $\widehat{P}$ is increased proportionally to a step δ. It is decreased proportionally to δ if $\bar{P}$ is above P_{TH}. Once $\bar{P}$ straddles P_{TH}, the quantity δ is decreased. The process repeats until the allocation below the threshold and the allocation above it differ by one subcarrier's constellation level. This indicates that it is this increase in the constellation size that is responsible for pushing the average BER over the threshold. The algorithm is then terminated.

The algorithm also provides a fast exit option. If either the smallest, non-zero, throughput,

[1]Since the implemented algorithm does not allocate power, [9] was modified to solely perform bit allocation.

[2]Note that $\mathcal{C}$ is the set of all data-bearing subcarriers.

with the smallest probability of error, is above P_{TH}, or the largest possible throughput is below P_{TH}, the algorithm terminates immediately. The complete algorithm can be summarized by the following steps:

1) *Given γ_i, calculate $P_i(b, \gamma_i)$ $\forall i \in \mathcal{C}$, $b \in \mathcal{B}$.*
2) *Compute $\bar{P}$, defined in Eq.(1) with $b_i = \max\limits_{b \in \mathcal{B}} b$, $\forall i \in \mathcal{C}$.*
3) *If $\bar{P} \le P_{TH}$, choose b_i, $\forall i \in \mathcal{C}$, and end algorithm.*
4) *If $\min\limits_{i,b} P_i(b, \gamma_i) > P_{TH}$, then $b_i = 0$, $\forall i \in \mathcal{C}$ and end algorithm.*
5) *Find b_i s.t. $b_i = \max\limits_{\substack{b \in \mathcal{B}: \\ P_i(b,\gamma_i) \le \widehat{P}}} b$, $\forall i \in \mathcal{C}$.*
6) *Compute $\bar{P}$ defined in Eq.(1).*
7) *If first iteration of algorithm, reduce $\widehat{P}$ by δ and go to Step 5.*
8) *If both current and previous $\bar{P}$ values straddle P_{TH}, go to Step 10.*
9) *If both current and previous $\bar{P}$ values are above P_{TH}, reduce $\widehat{P}$ by δ, as per Eq. (2), and go to Step 5, else increase $\widehat{P}$ by δ, as per Eq. (3), and go to Step 5.*
10) *If previous and current allocations differ by one signal constellation level, make the allocation with $\bar{P} \le P_{TH}$ the final allocation and end the algorithm.*
11) *Reduce δ.*
12) *If the current allocation gives a $\bar{P} \ge P_{TH}$, reduce $\widehat{P}$ by δ and go to Step 5, else increase $\widehat{P}$ by δ and go to Step 5.*

Note that, unlike most of the existing bit allocation algorithms [8–10, 22], this algorithm does not aim to keep a constant probability of error across all subcarriers. Since the bits allocated are integers, it is unlikely to find the same probability of error across each subcarrier. Therefore, this scheme allows for the presence of outliers, as long as the $\bar{P}$ remains below P_{TH}.

An additional advantage is that this scheme does not require a constant granularity. This, again, corresponds to standards such as IEEE 802.11a that have a granularity of one bit for smaller constellations and of two bits for larger constellations.

The choice of the initial step δ as well as the choice of the initial $\widehat{P}$ will have a large influence on the convergence speed of the algorithm. It is therefore crucial to choose them properly.

B. Initial Step Size

The variable δ is a function of the average SNR of the system, $\bar{\gamma}$. It has been empirically determined that for low $\bar{\gamma}$, a large step size, δ, makes the algorithm converge to a solution faster. Coincidentally, for high $\bar{\gamma}$, a smaller δ requires fewer iterations from the algorithm. Thus, the reciprocal of a sigmoid function, was chosen to define the relation between δ and $\bar{\gamma}$, as it fulfills both constraints with a smooth transition for mid-range $\bar{\gamma}$ values.

The chosen method to increment and decrement $\widehat{P}$, as stated in Step 9 of the peak initialization algorithm [12], corresponds to the two following equations:

$$\widehat{P}^{+} = \widehat{P} - \widehat{P} \times \delta \quad \text{if } \bar{P} < P_{TH} \tag{2}$$

$$\widehat{P}^{+} = \widehat{P} + \frac{\widehat{P}}{\delta} \quad \text{if } \bar{P} > P_{TH} \tag{3}$$

where $\widehat{P}^{+}$ is the peak probability of error for the next iteration of the algorithm. Since $\delta < 1$, $\widehat{P}$ is incremented by a value larger than $\widehat{P}$ and decremented by a value that is smaller than $\widehat{P}$, which means the algorithm will converge to $\widehat{P}$ from above.

C. Default Peak Initialization

When choosing an initial value of $\widehat{P}$, the objective is to select an estimate that will be as close to the final $\widehat{P}$ as possible and, therefore, converge in a minimum number of iterations. The estimate is obtained by finding for each subcarrier $i \in \mathcal{C}$, the largest signal constellations $P_{\beta,i}$ for which the corresponding BER does not exceed the constraint P_{TH}, as well as the smallest signal constellation $P_{\alpha,i}$ for which the corresponding BER exceeds P_{TH}. Starting with the smallest $P_{\alpha,i}$ and while the mean BER $\bar{P}$ is under the threshold, the values of $P_{\alpha,i}$ are used to progressively replace the corresponding $P_{\beta,i}$. $\widehat{P}$ is then chosen to be the last $P_{\alpha,i}$ added that does not violate the constraint P_{TH}. The algorithm is described as:

1) *Given* γ_i, *calculate* $P_i(b, \gamma_i)$, $\forall i \in \mathcal{C}$, $b \in \mathcal{B}$.
2) *Find* $P_{\beta,i} = \max\limits_{\substack{b \in \mathcal{B} \\ P_i(b,\gamma_i) < P_{TH}}} P_i(b, \gamma_i)$ *and* $\beta_i = \max\limits_{\substack{b \in \mathcal{B} \\ P_i(b,\gamma_i) < P_{TH}}} b$, $\forall i \in \mathcal{C}$.
3) *Find* $P_{\alpha,i} = \min\limits_{\substack{b \in \mathcal{B} \\ P_i(b,\gamma_i) > P_{TH}}} P_i(b, \gamma_i)$ *and* $\alpha_i = \min\limits_{\substack{b \in \mathcal{B} \\ P_i(b,\gamma_i) > P_{TH}}} b$, $\forall i \in \mathcal{C}$.

4) *Define set $\mathcal{D}$ s.t. $\mathcal{D} = \left\{ i : P_{\beta,i} \geq \frac{\max_{j\in\mathcal{C}} P_{\beta,j}}{10} \right\}$, ($P_{\beta,i}$ not within an order of magnitude can be neglected).*
5) *Given $P_{\beta,i}$, $i \in \mathcal{D}$, find ΔP s.t.:*

$$\Delta P = \sum_{i\in\mathcal{D}} \beta_i (P_{TH} - P_{\beta,i}) \tag{4}$$

6) *Sort $P_{\alpha,i}$ in an increasing order. Set k in $P_{\alpha,k}$ as the index of the sorted elements.*
7) *Find the largest value N_α, for which the following equation is true:*

$$\Delta P \geq \sum_{k=1}^{N_\alpha} \alpha_k (P_{\alpha,k} - P_{TH}) \tag{5}$$

The value P_{α,N_α} is the initial $\widehat{P}$ for the algorithm defined in Section III-A.

Using the algorithm for calculating the initial $\widehat{P}$ provides good results, reducing the number of iterations substantially [23]. However, the faster convergence comes at the price of an increased computational complexity. More specifically, the computation of the initial $\widehat{P}$ requires several loops of N iterations as well as at least one sort for $P_{\alpha,i}$, (O($N \log N$)). Effectively, computation of the initial $\widehat{P}$ can have a computational complexity comparable to that of several iterations of the main loop of the algorithm. This issue is addressed by an alternative peak BER initialization routine, presented in the next section.

D. Correlation-Based Peak Initialization

An alternative solution is to exploit the correlation in time between the successive variations of the channel to come up with an initial approximation of $\widehat{P}$. As the channel varies with time, the algorithm needs to recalculate the throughput of the transmitter. We can reasonably assume that for an indoor wireless local area network the channel parameters will only vary slowly with time[3]. Consequently, there will be a degree of correlation between the SNR values of corresponding subcarriers taken at consecutive time intervals. Thus, the final peak BERs resulting from successive runs of the algorithm will, therefore, be correlated. As a result, instead of recalculating the initial $\widehat{P}$ at each run, it is possible to simply use the final $\widehat{P}$ of the previous

[3]In this work, we assume the target application of the implementation is an indoor WLAN. Thus several operating parameters are borrowed from the IEEE 802.11a standard [2].

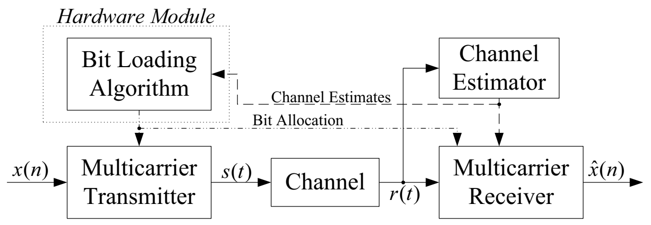

Fig. 1. Bit Loading Module, Part of an MCM hybrid system.

run as an initial estimate. Note that during the initial run of the algorithm, the peak initialization of Section III-C still needs to be used.

In the remainder of this work, the algorithm using the peak initialization described in Section III-C will be referred to as the *default algorithm*, whereas the variation proposed in this section will be referred to as the *correlation-based algorithm*.

The exact final solution, as well as the number of iterations needed to converge by the two versions of the algorithm, depend on the choice of the initial peak $\widehat{P}$. In the case of the default algorithm, the algorithm will always converge to the same solution for a given channel, since the initial peak is recalculated for each channel. However, since the correlation-based algorithm uses the final peak from the previous algorithm, the exact final solution will change depending on the solution found for the preceding channels.

IV. Adaptive Bit Loading Implementation in DSP

The bit loading algorithm is implemented onto a fixed-point DSP. It is designed as a distinct module that can be fitted onto the transmitter of an MCM system, as shown in Fig. 1. The module receives channel parameters from a channel estimator placed at the receiver. It then outputs the corresponding bit allocation that the transmitter uses to send data. The same bit allocation is used at the receiver to properly decode the received symbols. Note that in the case of a duplex communication, the module is to be placed on only one of the transceivers e.g. the base station, as the same allocation can be re-used by the second transceiver, e.g. subscriber station. Due to its modular design, the bit loading algorithm is transferable to reconfigurable radio platforms such as cognitive radios [7] and software-defined radios [6, 18].

TABLE I
HARDWARE DESCRIPTION

Parameter	Value
CPU	C64x
Clock Rate	600 MHz
RAM	8 MB
Instruction/Cycle	8×32-bits
I/O Rate (RTDX)	8-12 kB/s
Buffer Size (Input)	2 kB
Buffer Size (Output)	576 B

A. Hardware Description

The adaptive bit loading algorithm is implemented on a Texas Instruments TMS320C6416 fixed-point DSP [24]. The choice of a fixed-point DSP system is motivated by its ease of implementation and performance, i.e. superior speed and lower power consumption. The TMS320C6416 is a high-performance 32-bit processor with a 600MHz clock frequency, making it ideal for communication applications such as the one discussed in this work. The details of the hardware used are summarized in Table I.

In the case of this implementation, C is used as the language of choice for most of the algorithm. Furthermore, built-in DSP-BIOS functions [25] are used to control the module's runtime parameters, i.e., clock speed, stack size, etc., used during the algorithm's run.

B. Algorithm Modifications

In the hardware implementation, several aspects have to be re-worked and additional routines have to be added. First, modifications aimed at increasing the module's computational efficiency and robustness to errors are implemented. Second, the module is connected to a *host computer*, used to simulate external events and data exchanges. Therefore, input-output (I/O) routines are added to make those communications possible. Data is sent from the host to an asynchronous ping-pong buffer. As one half of the buffer is receiving data in the background, the data contained by the other half is processed by the algorithm. The data transfers and the algorithm's operations are controlled by an *algorithm manager*.

While converting the algorithm to fixed-point arithmetic (using Q-notation) makes it faster, it also has the disadvantage of increasing the computational errors due to finite precision. Thus, different strategies were developed to mitigate the precision loss. For instance, unlike for the mean BER, $\bar{P}$, which is always recalculated from the initial BER values, $\widehat{P}$ is calculated iteratively. Therefore, the error due to finite precision will propagate and increase at each iteration. The more iterations the algorithm takes to converge, the larger the finite precision error on $\widehat{P}$ will be, increasing the probability of the algorithm converging to a different solution. This is particularly true for small values of δ, i.e., at high $\bar{\gamma}$. The product in (2) will be small and therefore have a large precision error. This will dramatically increase the number of iterations required for convergence. A guide rail for setting an upper limit on the processing time of a single run has been added, limiting the number of iterations to $\kappa_{MAX} = 30$. The upper limit, κ_{MAX}, is approximately six times larger than the average number of iterations that is required for convergence. One additional modification is made to Step 6 of the peak allocation algorithm. Prior to the sorting, an initial pass discards all the values of $P_{\alpha,i}$ such as $P_{\alpha,i} > \Delta P$, as those values cannot satisfy (5). This modification reduces the number of values to be sorted by, on average, 50%.

The execution of the bit loading algorithm is handled by an *algorithm manager*, shown in Fig. 2, which is also responsible for I/O transfers between the algorithm and the host.

Two control signals can call the algorithm manager. First, as the channel changes, the channel estimator tells the module to calculate a new bit allocation with the new channel parameters. Second, the MCM system sends in a request for the most up-to-date bit allocation to be sent back to the transmitter. In the case of this implementation, those two signals are grouped together, i.e., as soon as new channel parameters are available, the most recent bit allocation is automatically sent back to the host. The “new channel/allocation request” signal can be periodic or aperiodic, representing either the estimator continuously sampling the channel, or the estimator sending channel parameters to the module only once they change significantly.

In the case of this implementation, the control signal is generated on the DSP itself. This choice is justified by the better clock resolution of the C64x DSP when compared to the clock resolution of the host computer. The control signal can be chosen to be generated periodically, with a period T_{BA}, or at random times. In the latter case, the period at which a control signal is generated can be considered as a random variable T_{BA}.

The algorithm manager will interrupt the algorithm if the latter is already running. Once called, and if there is no previous instance of the algorithm running, the manager sends back the final

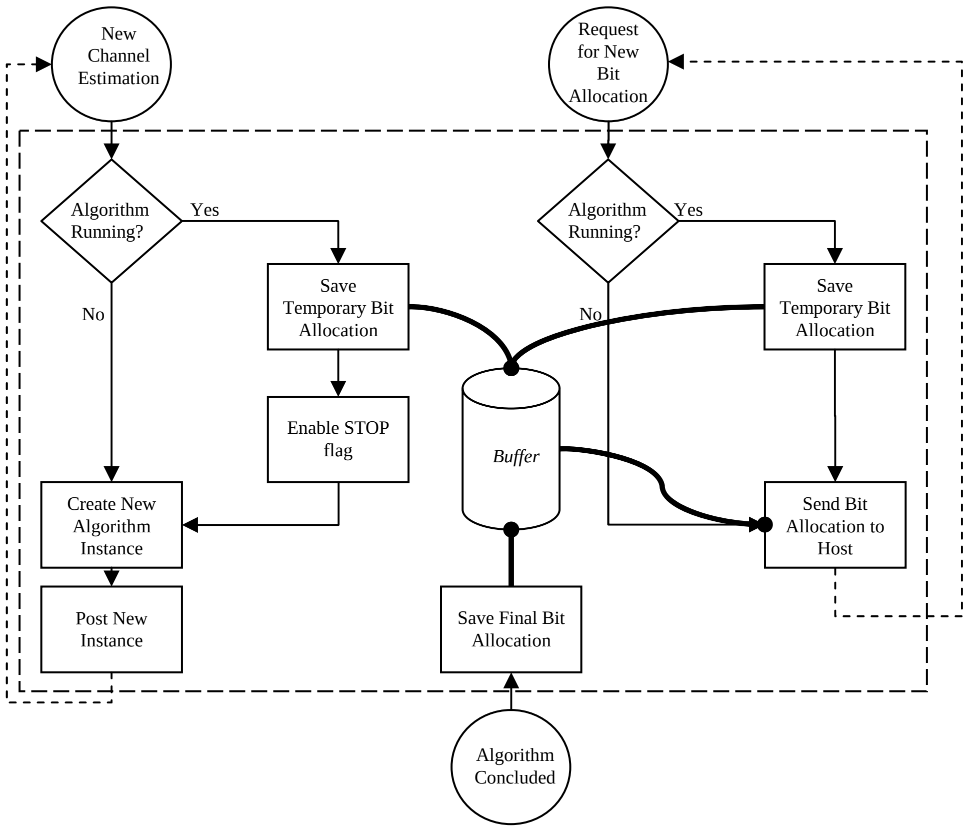
New Channel Estimation
Request for New Bit Allocation
Algorithm Running?
Yes
No
Save Temporary Bit Allocation
Algorithm Running?
Yes
No
Save Temporary Bit Allocation
Enable STOP flag
Buffer
Create New Algorithm Instance
Send Bit Allocation to Host
Post New Instance
Save Final Bit Allocation
Algorithm Concluded

Fig. 2. Algorithm Manager.

solution computed by the last instance of the algorithm, and then creates a new instance and calls it. Otherwise, if an instance of the algorithm is already running, the manager sends the host the temporary bit allocation found by the running instance, terminates it, and starts a new instance of the algorithm.

As T_{BA} decreases, or for smaller realizations of T_{BA}, it is more likely that the algorithm will not have time to finish and will be interrupted by the manager. Note that the upper constraint for T_{BA} is taken from the IEEE 802.11a standard [2]. This standard allows for a maximum block length of 4095 symbols (16ms) during which the channel is assumed to stay constant. Of course, it is preferable that the algorithm converges to a solution much faster, leaving more time for the MCM system to actually use the bit allocation for transmission.

V. EXPERIMENTAL RESULTS

A. Experimental Setup

As is the case for the IEEE 802.11a standard [2], the constellations allowed in this implementation are BPSK, 4-QAM, square 16-QAM and square 64-QAM. For especially low SNR values, each subcarrier can also be *nulled*, i.e., turned off.

The probability of error is calculated by closed form expressions derived for AWGN and specified in [21]. To minimize the computational cost, the expressions are implemented with a lookup table. In the case of this implementation, the lookup table is comprised of 2^{10} unsigned integers for each of the four constellations, i.e., 10^{15} bits per constellation. The values are logarithmically spaced. The lookup table will span probability of errors that are three orders of magnitude above the threshold P_{TH} and three orders of magnitude below P_{TH}. It is assumed that any transmission with a probability of error above the maximum value P_{MAX} of the lookup table would be unacceptably poor. Similarly, the probability of error below the minimum value of the table P_{MIN} is too small to be considered. For this implementation, $P_{TH} = 10^{-5}$ seems to be a reasonable choice for wireless data transmission without coding. Therefore, the lookup table will range from $P_{MAX} = 10^{-2}$ to $P_{MIN} = 10^{-8}$ for all four of the non-null constellations. Since there are no values larger than P_{MAX} in the lookup table, all the probabilities of error are scaled by P_{MAX} to make an efficient use of all the bits available.

The use of the lookup table gives a maximal error of 1.5% when compared to a value calculated using the closed form expressions. Also, note that the error caused by the finite precision format of the table is insignificant, i.e., at most it is less than 0.03% for the smallest values of the BER table. It is even less for larger values.

Since the target application for the bit loading algorithm is an indoor WLAN modem, the channels are generated using the statistical model for indoor environments described by Saleh and Valenzuela [26]. Although the model was developed for a 1.5GHz frequency band, it is easily expandable to 5GHz indoor environments.

The random channels generated in this experiment use the parameters displayed in Table II. The distance between the transmitter and the receiver was set to 5m, 16m and 50m, without a line of sight component. Four sets, each comprising three subsets of 10,240 channels, are generated. Each set has a different degree of correlation between the channels. In the first set,

TABLE II
GENERATED CHANNEL PARAMETERS

Parameter	Value
Number of Channels	64
Number of Guard Channels	12
Frequency Band	5 GHz
Subcarriers Power	0.83×10^{-3} W
Noise Power	-124 dB
Distance between transmitter and receiver	5 m, 16 m, 50 m
Number of clusters	5
Number of rays per cluster	100
Exponential decay of clusters	40.0×10^{-8}
Exponential decay of rays	8.0×10^{-8}

$X1$, the channels are uncorrelated. The remaining sets have a correlation coefficient of 0.910, 0.990 and 0.999. The correlated channels are obtained by interpolating uncorrelated channels in the time domain by a factor of 4, 16 and 128. They will subsequently be identified as $X4$, $X16$ and $X128$ respectively. The subsets are classified relative to the distance from the transmitter, with $\bar{\gamma}_s = 31.6$dB at 5m, $\bar{\gamma}_s = 21.5$dB at 16m and $\bar{\gamma}_s = 11.5$dB at 50m, where $\bar{\gamma}_s$ is the SNR averaged over all the subcarriers of all the channels in that subset.

B. Results

The performance of the bit loading module is tested using the four sets of random channels previously generated. The two versions of the bit loading algorithm are examined: the default version and the correlation-based version.

The consequences of using fixed point arithmetic are analyzed. Furthermore, the performance of the implementation relative to different channel variation rates is discussed. The channels are initially assumed to vary with a constant period T_{BA}. In the last part of this section, the rate at which new channels are sent to the module is varied randomly. The random variation follows a uniform distribution centered at two values T_{C1} and T_{C2}.

During the experiment, the bit loading algorithm aims to find a bit allocation that will have a probability of error below the threshold P_{TH}. However, if the bit loading algorithm is interrupted by the algorithm manager, the constraint P_{TH} may be violated. In that case, the bit allocation is deemed invalid and the throughput of that run is set to zero. To obtain statistically significant values, the data is averaged over the 10,240 channels of each subset.

1) Effect of fixed-point arithmetic: It is interesting to examine the impact the use of fixed-point arithmetic has had on the implementation's speed of convergence and on the solution the algorithm converges to. The adaptive bit loading algorithm has two principal sources of finite precision errors. First, the computation of the average BER, $\bar{P}$, defined in (1), will introduce errors that can potentially force the algorithm to make an erroneous decision in Steps 8 and 9 of the algorithm described in Section III-A. In this scenario, the algorithm would converge to a solution that is slightly above or below that of a floating-point implementation. This error, however, seldom happens since the BER values are stored in a 32-bit variable which minimizes finite precision errors. Further, the average BER is recomputed at each iteration from the lookup table which means the error does not propagate. The second source of finite precision errors is more critical. Contrary to $\bar{P}$, the error on the peak BER $\widehat{P}$ does propagate and increases at each iteration. Further, the presence of a multiplication in (2) and (3) will make the error more substantial, especially for small δ. Here, finite precision error has two consequences. First, it can cause the wrong decision during the choice of the individual subcarrier's throughput as per Step 5 of the algorithm described in Section III-A. This will lead to a different solution or, at best, to a larger convergence time. Second, because of the error on δ, $\widehat{P}$ will increase and decrease more slowly, requiring more time to converge for the algorithm.

Although finite precision error is, indeed, present, its effect is not significant. When compared to theoretical results, obtained with floating-point arithmetic simulations, the error for the default algorithm is, on average, 0.0036 bits per allocation. For the correlation-based version, it is, on average 1.5 bits. However, since the algorithm is not optimal, the solution obtained using fixed-point arithmetic is in no way worse than the theoretical results. It is simply a different near-optimal solution. It is important to emphasize that, in both cases, P_{TH} is respected.

2) Periodically Time-Varying Channel: T_{BA} is varied from 500μs to 60μs. Those values seem reasonable and are significantly less than 16ms. At $T_{BA} = 500\mu$s, both the correlation-based algorithm and default algorithm have enough time to finish each run. Conversely, at $T_{BA} = 60\mu$s, the two versions of the algorithm are interrupted at each run. Fig. 3 shows the averaged number of interrupts for the $X128$ channel set, at 5m and 16m for the two versions of the algorithm. Clearly, the algorithm is interrupted earlier for channels when the receiver is at 5m than when

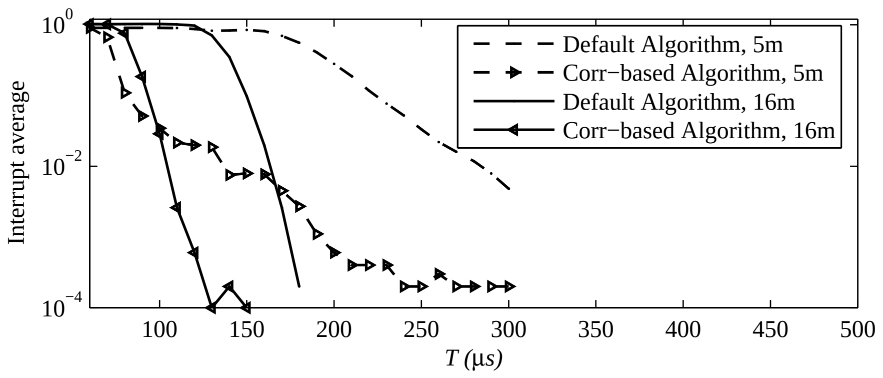

Fig. 3. Interrupt average for Default and Correlation-based algorithm, X128 channel set at 5m and 16m.

the receiver is at 16m. As the distance between the transmitter and the receiver decreases, the step size δ gets smaller and is more prone to finite precision errors. Thus, the algorithm takes more iterations to converge. In the worst cases, convergence can take up to 2ms. However, because the maximum number of iterations is limited to 30, the algorithm is never interrupted at $T_{BA} = 500\mu$s. Note that this phenomenon is only valid for very small values of δ. There isn't any noticeable difference in the number of interrupts between subsets at 16m and at 50m.

As expected, at equal T_{BA}, the average is larger for the default version because of the peak initialization routine. The difference is larger for low $\bar{\gamma}$, being approximately 130μs at 5m, whereas it is 50μs at 16m. This discrepancy is caused by a longer peak initialization at 5m.

As the number of interrupts increases, the throughput itself decreases. Fig. 4 shows the throughput at 16m for the $X128$ channel set. Here, the correlation-based algorithm offers a clear advantage when compared to the default algorithm as the throughput starts declining on average 60μs later than for the default version. This difference is even larger for channels with high SNR due to the longer initial $\widehat{P}$ calculations for the default algorithm. Interestingly, this is true not only for the correlated channel sets, but also for the uncorrelated set, $X1$. All in all, the correlation-based algorithm offers a better throughput, notwithstanding the correlation level.

Relating the throughput of the system with the average number of iterations completed by each run, sheds more light on the algorithm. As shown in Fig. 5, the number of iterations starts decreasing approximately 20μs before the throughput does. This indicates that the loss of the last few iterations does not produce any significant reduction in throughput. Since δ is larger during the first few iterations, it is then that $\widehat{P}$ changes more substantially.

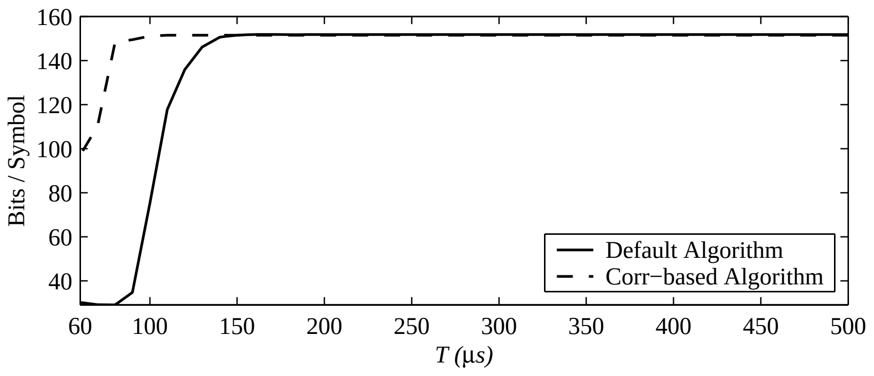

Fig. 4. Throughput for Default and Correlation-based algorithm, X128 channel set at 16m.

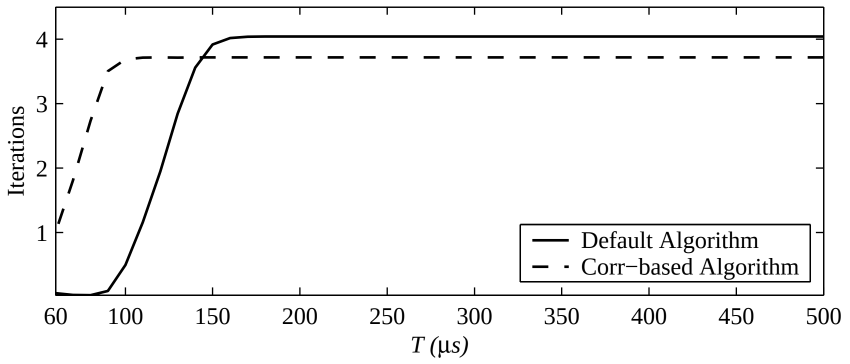

Fig. 5. Number of Iterations for Default and Correlation-based algorithm, X128 channel set at 16m.

When comparing the two versions of the algorithm for the uncorrelated channel set $X1$, it appears that the number of iterations needed for convergence is roughly the same for both versions, with the correlation-based version requiring marginally more iterations. This would tend to suggest that the precise value of the initial $\widehat{P}$ is not crucial, as it does not significantly increase the number of iterations or the final throughput. However, as shown in Fig. 5, for correlated channels, the number of iterations decreased by an average of 10%.

C. Time-Varying Channel with Random Period

In this section, T_{BA} is set to be a discrete random variable, represented by T_{BA}. It is centered around two center periods, set to be $T_{C1} = 205\mu$s and $T_{C2} = 235\mu$s. The performance of the two versions of the algorithm is examined in relation to the standard deviation of T_{BA}, σ_T. As was the case in the previous section, the correlation-based algorithm is consistently less interrupted

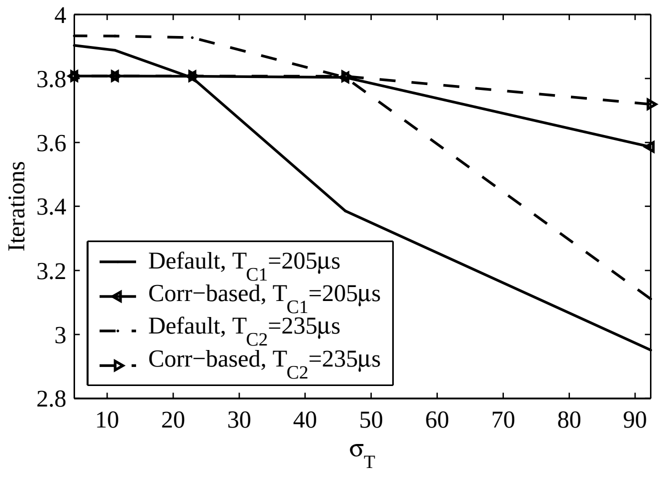

Fig. 6. Number of Iterations for Default and Correlation-based algorithm, X16 channel set at 16m.

than the default version.

Whether or not the number of interrupts increases or decreases with σ_T will depend on the values of T_{C1} and T_{C2}. If the mean of T_{BA} is large relative to the average execution time of the algorithm, the number of interrupts will increase with increasing σ_T. As σ_T increases, some runs are interrupted earlier, whereas others are given "unnecessary" processing time, thereby increasing the average number of interrupts. Conversely, if the mean of T_{BA} is small relative to the average execution time of the algorithm, for low σ_T almost every run is interrupted early in its execution. When σ_T increases, however, at least some realizations of T_{BA} will be large enough to allow the algorithm to complete, thereby reducing the average number of interrupts.

The number of iterations, shown in Fig. 6 for $X16$ channel subset, at 16m, correlates with the average number of interrupts. For runs that have a short execution time relative to the mean of T_{BA}, the number of iterations will decrease with an increasing σ_T. For runs with larger execution time with respect to the mean of T_{BA}, the number of iterations will increase with σ_T.

Fig. 7 shows the throughput of the $X16$ channel subset, at 16m. Note that the increase in the number of interrupts and the decrease in the number of iterations does not have any significant effect on the throughput of the correlation-based algorithm, as it stays approximately constant for all σ_T. This confirms that the last iterations of each run have mitigated effect on the final throughput. Conversely, the throughput decreases with reduced iterations for the default algorithm as it gets interrupted earlier in its execution.

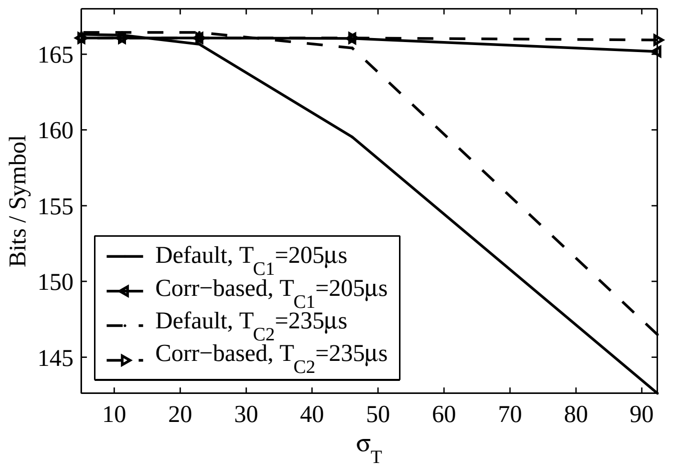

Fig. 7. Throughput for Default and Correlation-based algorithm for $X16$ channel at 16m.

VI. Conclusion

A modular DSP implementation of a robust adaptive bit loading algorithm is presented in this work. The objective of this proof-of-concept is to operate in a real-time, computationally efficient manner. The algorithm is converted to fixed-point arithmetic, and its flow is streamlined. Furthermore, a second, computationally simpler version of the default algorithm is proposed: a correlation-based version, which uses information from previous runs to minimize the number of iterations required for convergence. Due to these modifications, both versions converge to a solution within a fraction of the maximum time constraint. In the experimental results, for rapidly time-varying channels, the correlation-based version proves to be superior in every aspect considered (number of iterations, throughput, etc.), whereas for slowly varying channels, the performance of the two bit loading algorithm versions are comparable. The level of correlation between the successive variations of the channels plays a minimal role in the correlation-based version's throughput, making it the method of choice for future usage.

With decreasing variation rates of the channel, the CPU load decreases. The implementation then becomes particularly interesting for multi-user schemes, such as wireless base stations in a cellular network. In this case, several instances of the algorithm can run in parallel on a single DSP chip.

References

[1] J. A. C. Bingham, *ADSL, VDSL, and Multicarrier Modulation*. John Wiley and Sons, New York, 2000.

[2] IEEE, “Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY): High-speed physical layer in the 5 GHz band,” *IEEE Standard 802.11a*, Nov. 1999.

[3] IEEE, “Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Further Higher-Speed Physical Layer Extension in the 2.4 GHz Band,” *IEEE Standard 802.11g*, June 2003.

[4] IEEE, “Air Interface for Fixed Broadband Wireless Access Systems,” *IEEE Standard 802.16*, Apr. 2004.

[5] IEEE, “Air Interface for Fixed and Mobile Broadband Wireless Access Systems Amendment 2: Physical and Medium Access Control Layers for Combined Fixed and Mobile Operation in Licensed Bands and Corrigendum 1,” *IEEE Standard 802.16*, Feb. 2006.

[6] J. Mitola, “Software radios survey, critical evaluation and future decisions,” *IEEE Aerospace and Electronic Systems Magazine*, vol. 8, pp. 25–36, 1993.

[7] J. Mitola, “Cognitive radio for flexible mobile multimedia communications,” *IEEE International Workshop on Mobile Multimedia Communications*, pp. 3–10, 1999.

[8] P. Chow, J. Cioffi, and J. Bingham, “A Practical Discrete Multitone Transceiver Loading Algorithm for Data Transmission over Spectrally Shaped Channels,” *IEEE Transactions on Communications*, vol. 43, pp. 773–775, Feb. , Mar., Apr. 1995.

[9] A. Leke and J. Cioffi, “A Maximum Rate Loading Algorithm for Discrete Multitone Modulations Systems,” *IEEE Global Telecommunications Conference*, vol. 3, pp. 1514–1518, Nov. 1997.

[10] R. Fischer and J. Huber, “A New Loading Algorithm for Discrete Multitone Transmission,” *IEEE Global Telecommunications Conference*, vol. 1, pp. 724–728, Nov. 1996.

[11] A. M. Wyglinski, F. Labeau, and P. Kabal, “An Efficient Bit Allocation Algorithm for Multicarrier Modulation,” *IEEE Wireless Communications Networking Conference*, vol. 2, pp. 1194–1199, Mar. 2004.

[12] A. M. Wyglinski, F. Labeau, and P. Kabal, “Bit Loading with BER-Constraint for Multi-Carriers Systems,” *IEEE Transaction on Wireless Communications*, June 2003. accepted.

[13] T. Keller and L. Hanzo, “Adaptive Multicarrier Modulation: A Convenient Framework for Time-Frequency Processing in Wireless Communications,” *Proceedings of the IEEE*, vol. 88, pp. 611–640, May 2000.

[14] D. Hughes-Hartog, “Ensemble Modem Structure for Imperfect Transmission Media,” *U.S. Patent 4679227*, July 1987.

[15] E. Baccarelli, A. Fasano, and M. Biagi, “Novel Efficient Bit-Loading Algorithms for Peak-Energy-Limited ADSL-type Multicarrier Systems,” *IEEE Transactions on Signal Processing*, vol. 50, pp. 1237–1247, May 2002.

[16] D. Daly, C. Henegan, and A. Fagan, “Power- and Bit-Loading Algorithms for Multitone Systems,” *International Symposium on Image and Signal Processing and Analysis*, vol. 2, pp. 639–644, 2003.

[17] B. Krongold, K. Ramchandran, and D. Jones, “Computationally Efficient Optimal Power Allocation Algorithms for Multicarrier Communication Systems,” *IEEE Transactions on Communications*, vol. 48, pp. 23–27, Jan. 2000.

[18] S. Srikanteswara, R. Palat, J. Reed, and P. Athanas, “An overview of configurable computing machines for software radio handsets,” *IEEE Communications Magazine*, vol. 41, pp. 134–141, July 2003.

[19] A. M. Wyglinski, F. Labeau, and P. Kabal, “Effects of Imperfect Subcarrier SNR Information on Adaptive Bit Loading Algorithms for Multicarrier Systems,” *Proceedings of the IEEE Global Telecommunications Conference*, vol. 6, pp. 3835–3839, Nov. 2004.

[20] B. Fox, “Discrete Optimization via Marginal Analysis,” *Management Science*, vol. 13, pp. 210–216, Nov. 1966.

[21] J. Proakis, *Digital Communications*. McGraw-Hill, New York, 3rd ed., 1995.

[22] E. Baccarelli, A. Fasano, and A. Zucchi, “On the Loading of Peak-Energy-Limited Multicarrier Systems Transmitting over

Spectrally-Shaped Crosstalk-Impaired DSLs," *IEEE International Conference on Communications*, vol. 1, pp. 310–314, June 2001.

[23] A. M. Wyglinski, *Physical Layer Loading Algorithms for Indoor Wireless Multicarrier Systems*. PhD thesis, McGill University, Nov. 2004.

[24] Texas Instrument, *TMS320C64x Technical Overview, Lit. Num. SPRU395B*, Jan. 2001.

[25] Texas Instrument, *TMS320C6000 DSP/BIOS User's Guide, Lit. Num. SPRU303B*, May 2000.

[26] A. A. M. Saleh and R. A. Valenzuela, "A Statistical Model for Indoor Multipath Propagation," *IEEE Journal on Selected Areas in Communications*, vol. 5, pp. 128–137, Feb. 1987.

Martin Cudnoch received his M.Eng. degree in Electrical Engineering from McGill University, Montreal, Canada, in 2006, specializing in telecommunications and signal processing. He received his B.Eng. (Honours) degree in Electrical Engineering in 1999, also from McGill University, specializing in telecommunications and control & automation.

Between 1999 and 2003, Martin worked as an applications engineer at Teradyne Inc. in Boston. He is presently working as a DSP engineer at SR Telecom Inc., Montreal, developing WiMAX technology. His areas of interest include adaptive OFDM transmission and algorithm optimization techniques.

Alexander M. Wyglinski is an Assistant Research Professor with the Information and Telecommunication Technology Center (ITTC) at The University of Kansas. He received his Ph.D. degree from McGill University in 2005, his M.S. degree from Queens University at Kingston in 2000, and his B.Eng. degree from McGill University in 1999, all in electrical engineering.

Professor Wyglinski serves on the editorial board for the IEEE Communications Surveys and Tutorials, as a guest editor for the IEEE Communications Magazine feature topic on Cognitive Radios for Dynamic Spectrum Access (to appear May 2007), as the "Transceiver Technologies" track co-chair for the 64th IEEE Vehicular Technology Conference (VTC), as the technical program committee co-chair for the 2nd International Conference on Cognitive Radio Oriented Wireless Networks and Communications (CrownCom 2007), and as a technical program committee member for the 2006 IEEE Global Telecommunications Conference (Globecom 2006), the 2007 IEEE International Conference on Communications (ICC 2007), the 2007 IEEE Wireless Communications and Networking Conference (WCNC 2007), and the 2007 IEEE Workshop on Cognitive Radio Networks. He also regularly conducts reviews for several IEEE publications and conferences.

Professor Wyglinski's current research interests are in the areas of wireless communications, wireless networks, cognitive radios, software-defined radios, transceiver optimization algorithms, dynamic spectrum access networks, hybrid fiber-wireless networking, and signal processing techniques for digital communications.

Fabrice Labeau received his “Ingenieur civil” degree, “Diplome d’etudes avancees en Telecommunications” and Ph.D. degree in Electrical Engineering from the Universite catholique de Louvain, Belgium, in 1995, 1996 and 2000 resp. Since 2000, he has been with the Electrical and Computer Engineering Department at McGill University, Montreal, Canada, where he is now an associate professor. His current research interests include joint source/channel coding, mutlirate systems and multimedia transmission. He was a TPC member for the IEEE Vehicular Technology Conference in the Fall of 2004, and for the EUSIPCO conference in 2004 and 2005. He was part of the organizing committee of ICASSP 2004 in Montreal, and is the technical program committee co-chair for the IEEE VTC 2006 Fall conference.