# ASIP Design of a Reconfigurable Channel Estimator for OFDM Systems

Medhat Hamdy
Center for Wireless Studies
Cairo University, Egypt
Email: mhamdy_1987@yahoo.com

Omar A. Nasr
Center for Wireless Studies
Cairo University, Egypt
Email: omaranasr@ieee.org

Ahmed F. Shalash
Center for Wireless Studies
Cairo University, Egypt
Email: shalash@ieee.org

*Abstract*— **Progress in wireless communications has led to an increasing number of standards such as WIMAX, 3GPP-LTE, DVB-T, DVB-H and more. This work presents a reconfigurable channel estimation engine for OFDM systems. The engine can support multiple channel estimation algorithms for many wireless standards. It can also support both 1D and 2D channel estimation algorithms. This is important because the large performance gain of 2D over 1D channel estimation algorithms, which can be up to 5dBs. Moreover, 1D algorithms are less complex than 2D algorithms and they are the best choice when there is a need to reduce the receiver processing power. The engine does not use any dividers for least square pilots estimation, and hence, the least square pilot estimator engine is less complex than other engines. Although the engine was designed and optimized for channel estimation algorithms, it can also be configured as an FFT engine, FIR engine and many more. The engine was synthesized on Altera Startix III EP3SC150 FPGA and performs the estimation process in $94$ $\mu$second for the worst case parameters in all supported standards.**

## I. Introduction

The diversity of QoS requirements in different applications has lead to the development of a large number of wireless standards. Today, WIMAX, LTE and Wi-Fi are samples of the existing wireless standards in the market. Although different and diverse committees lead the efforts in the development process of these standards, they have all selected Orthogonal Frequency Division Multiplexing (OFDM) as their modulation technology. Unlike single carrier systems, OFDM does not need complex time domain equalization and the required FFT/IFFT operations for OFDM can be done easily using today's signal processing technology. To be able to work in different environments, some standards like WIMAX and LTE supports different FFT sizes. For example, 802.16m can support 512, 1024 and 2048 FFT sizes.

On the implementation side, ASIP is an Application Specific Instruction set Processor. ASIP is mainly designed to maintain the flexibility in a certain application domain without sacrificing the other factors like power, area, and performance. Nowadays, it is common that mobile phones support more than one communication technology at the same time. In this paper, wireless communication is our application domain. We took the channel estimation as our case study.

Several efforts had been made in the field of channel estimation engines. In [1], the authors presented two ASIC receivers to implement wireless transmitter and receiver required for a high-speed wireless OFDM systems. Paper [2] presents an Application Specific Instruction-Set Processor (ASIP) capable of supporting a set of customized hardware modules suited for wireless processing. On the other hand, paper [3] presents a dedicated hardware to perform channel estimation task on many wireless standards using specific channel estimation algorithm. In [4], an FFT processor is extended to support all data processing operations required by an OFDM channel estimation algorithm.

Although the literature is rich with different implementations of wireless communications engines, there are only few efforts that investigate the reconfigurability dimension in the implementation of wireless receivers. in [5], the author presents an energy efficient reconfigurable baseband processor that supports all the operations required by wireless transceivers. In [6], the authors presented a configurable processor to support channel estimation and other functionalities in OFDM system.

Unlike previous efforts in this field, our processor can support 2-D channel estimation algorithms and does not have any dividers. The authors in [6] introduced a reconfigurable channel estimation engine that supports only 1D channel estimation algorithms. 2-D channel estimation algorithms outperform 1D algorithms by more than 5Bs in high SNR. Moreover, we use a division free hardware engine that is based on our previous work in [3]. Thus, the least square estimation part in our engine is less complex than previous reconfigurable engines.

This paper is organized as follows: Section II presents the basic operations used in the channel estimation algorithms. Section III discusses the reconfigurability dimensions in the proposed processor. Section IV presents the processor architecture. Section V presents the results and performance evaluation of the processor. In section VI, an implementation scenario is discussed using one of the channel estimation algorithms. Finally, the paper is concluded in VII

## II. Basic Operations In Channel Estimation

Channel estimation algorithms are classified as decision-directed and pilot-aided channel estimation. Pilot-aided estimation provides better performance than the decision-directed, especially for systems with large Doppler frequencies [7]. Our processor is concerned only with pilot-aided channel estimation algorithms as many standards use pilots to facilitate the channel estimation and synchronization tasks at the receiver.

Least Squares (LS) process, interpolation and filtering are common operations that will be used in all channel estimation algorithms under consideration.

### A. LS process

The LS process is divided into two tasks: LS separation and estimation. The separation process separates the pilots from the data subcarriers in an OFDM symbol and stores them in separate RAMs.

This work was supported by a grant from STDF, Egypt.

The separation can be performed using a reconfigurable shift register. By circularly shifting the register, we get serial output that enables the data or pilots RAM. The LS Separator is designed to be fully configurable to work for all the FFT sizes and standards.

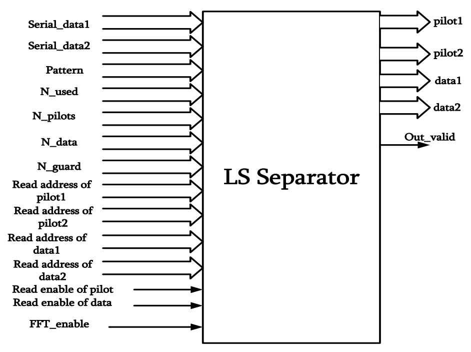

Fig. 1. LS Estimator block diagram

Fig.1 shows a block diagram for the LS separator. The different inputs are used to define the mode of operation of the LS separator. For instance, the pattern input is used to define the sequence of repetition of the pilots in the OFDM symbol. To illustrate this, consider a cluster of 802.16m standard. It has 14 subcarriers with 2 pilots in them. Thus, the pattern for the even symbol would be '00001000100000'. The '1' represents a pilot and the '0' represents a data subcarrier.

The second task is the LS estimation. The LS estimation estimates the channel response at pilot positions by dividing the received pilots by their actual transmitted values as shown in Eq.(1). In [3], the author avoided the division by multiplying the received pilots by the conjugate of their actual transmitted values and the division by $|X_p|^2$ is embedded in the filter coefficients. The problem of this solution is the complexity of implementing the algorithms that don't have filtering process. Thus, we chose to multiply directly the received pilots by the inverse of their actual transmitted values. This removes the division process and makes the engine more configurable. The channel response at pilot positions can be estimated as:

$$\breve{H}_p = Y_p/X_p = Y_p * (1/X_p) \qquad (1)$$

Where:
$\breve{H}_p$ is the channel estimate at the pilot position
$X_p$ is the value of the transmitted pilot
$Y_p$ is the received pilot

The reason for separating the two tasks is that both the channel estimation and synchronization processors need a separation of the pilots from the data. Thus, the LS separator is provided as a separate engine.

### B. Interpolation

Linear interpolation is used in our engine because it performs better than all other interpolation methods in terms of performance [8]. The function of the interpolation process is to interpolate the channel estimates of the pilots in order to get a channel estimate in between, either in time or in frequency direction. Two pilots will be multiplied by constants and products are added. The middle symbols are interpolated using Eq.(2). For the first and last symbols, an extrapolation operation is used as in (3).

$$Y(x) = Y_{k-1} + (x - x_{k-1}) * (Y_k - Y_{k-1})/(x_k - x_{k-1}) \qquad (2)$$

Where:
$x$ is the subcarrier index between $x_{k-1}$ and $x_k$
$Y$ is the channel estimate calculated by interpolation

$$Y_1 = 3/2 * Y_2 - 1/2 * Y_4 \qquad (3)$$

### C. Filtering

A method of estimation is Minimum Mean Square Error (MMSE) robust estimation, which is the optimum algorithm for minimizing the mean square error of the estimation [9], [10]. The MMSE estimation is based on Wiener Filtering approach, where channel subcarriers are estimated through a filtering process between coefficients and the nearest $N_{tap}$ pilots to that subcarrier. Equation (4) explains the filtering process, where $\breve{H}$ is the channel estimate matrix, C is the coefficient matrix and H is the channel after the LS process. The coefficients are calculated by knowing the channel correlation matrix. Equation (5) explains how we calculate the filter coefficients, where $R_{\breve{H}}$ is the correlation matrix between different LS pilot estimates and $r_H$ is the correlation vector of the channel frequency response between data location and different pilot locations.

$$\breve{H} = C * H \qquad (4)$$

$$C = {R_{\breve{H}}}^{-1} * r_H \qquad (5)$$

## III. Reconfigurability

The reconfigurability in the proposed processor spans three dimensions: OFDM standards, FFT sizes and channel estimation algorithms. This section describes the reconfigurability dimensions and the different operations in the system.

### A. FFT sizes and standards

To meet the reconfigurability target, the system supports all the modulation schemes (up to 256 QAM) and up to 32k FFT points. Table I shows the different FFT sizes for the supported OFDM standards.

TABLE I
FFT sizes for some OFDM standards

| Standard | FFT sizes |
|---|---|
| WIMAX 802.16m | 512, 1024, 2048 |
| 3GPP-LTE | 128, 256, 512, 1024, 2048 |
| DVB-T | 2048, 8192 |
| DVB-H | 2048, 4096, 8192 |
| WIFI | 64 |

### B. Channel estimation algorithms

The channel estimation algorithms that can be supported are:

*1) 1-D frequency filtering:* this algorithm performs the channel estimation on each individual OFDM symbol. After the LS estimation, a filtering operation is done on all subcarriers in the frequency direction to estimate the channel in the received symbol [11].

*2) 1-D cascaded interpolation:* this algorithm uses the interpolation operation to find the channel in the time direction, then in the frequency direction. For instance, Fig.2 shows the pilot pattern for 802.16m, the interpolation is done first in time domain between channel estimates of pilots (shown in dark circles) to get the channel in the remaining subcarriers (shown in white circles). And then, another interpolation is done in the frequency direction to get the remaining channel estimates [8].

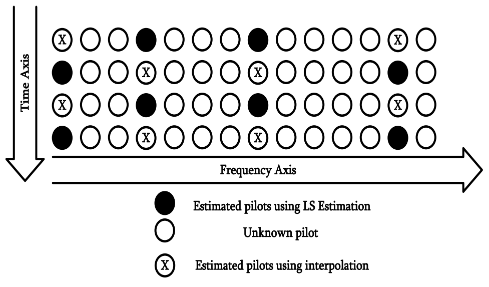
Time Axis
Frequency Axis
Estimated pilots using LS Estimation
Unknown pilot
Estimated pilots using interpolation

Fig. 2. Pilot pattern in WIMAX 802.16m

*3) 2-D filtering on two consecutive OFDM symbols:* this algorithm performs the filtering operation on two consecutive symbols by $N_{tap}$ pilots from the current and next OFDM symbols [11]. In our simulations we used 16 tap filter, 8 taps for the current symbol and 8 taps for the next symbol. This number of taps can be changed in both simulation and the ASIP by changing the program used to configure the engine.

*4) 2-D filtering on three consecutive OFDM symbols:* this algorithm performs the filtering on three consecutive symbols by $N_{tap}$ LS pilots from the current, previous and next symbols [11]. In our simulations we used 15 tap filter, 6 for the previous symbol, 5 for both the current next symbols.

*5) Time interpolation frequency filtering:* this algorithm is an enhancement on the cascaded algorithm. Interpolation is performed in time domain first and then a filtering is performed in frequency domain. The main advantage of this method is that it has very little latency compared to the cascaded filtering operation and much less complexity than the two dimensional filtering method [3].

In [6], all estimation algorithms were one dimensional estimation, where the cubic spline interpolation algorithm provides the best performance. Fig.3 shows a simulation comparison between different channel estimation algorithms on 802.16m standard. The cascaded interpolation algorithm performs better performance in low SNRs. The better performance of the two dimensional algorithms will come on the price of more computations as shown in Table II.

The third and fourth algorithms in Table II have the same complexity as they perform the filtering process on the same number of pilots, thus having the same number of real multiplications and additions. The presented processor can support all these channel estimation algorithms.

Fixed point simulations of 1024-point FFT in 802.16m system and time interpolation frequency filtering algorithm revealed that a 28 bit complex word length is sufficient to keep difference of less than 0.1dB at $10^{-4}$ Bit Error Rate (BER) between the fixed and floating point curves.

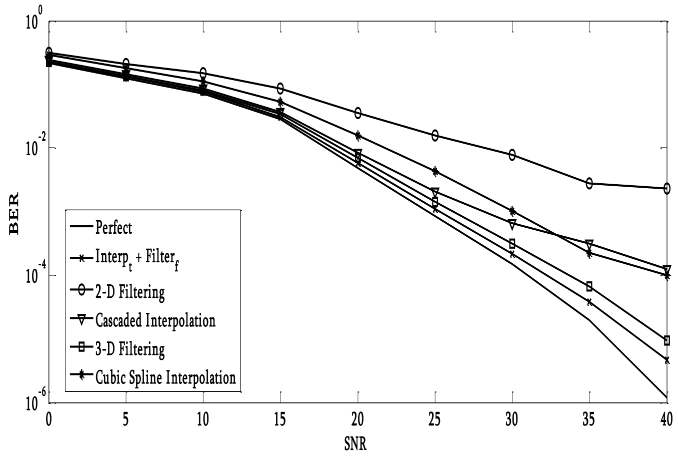

Fig. 3. Performance comparison between the different channel estimation algorithms on 802.16m standard

TABLE II
NUMBER OF PERFORMED OPERATIONS ON 1024-FFT POINTS PER OFDM SYMBOL FOR DIFFERENT CHANNEL ESTIMATION ALGORITHMS

| **Algorithm** | **Real Mul.** | **Real Add** |
|---|---|---|
| 1-D Filtering | 43200 | 43200 |
| 1-D Cascaded interpolation | 2880 | 1440 |
| 2-D Filtering on 2 consecutive OFDM Symbols | 46080 | 46080 |
| 2-D Filtering on 3 consecutive OFDM Symbols | 46080 | 46080 |
| Time Interpolation Frequency Filtering | 36480 | 36240 |
| Cubic Spline Interpolation | 7200 | 4320 |

## IV. PROCESSOR ARCHITECTURE

The processor presented in this section is responsible for performing the different arithmetic operations of the channel estimation algorithms. It can also support other communication tasks like FFT and FIR. The architecture is designed to support the reconfigurability dimensions. The choice of specific algorithm is performed according to the channel parameters. The sizes of the different elements in the system are calculated for 2k-FFT points. A generalization for N-FFT points is also considered.

Fig.4 shows the processor architecture. The processor includes one computational core, one control unit and a memory unit of five banks for storing the data and the intermediate results.

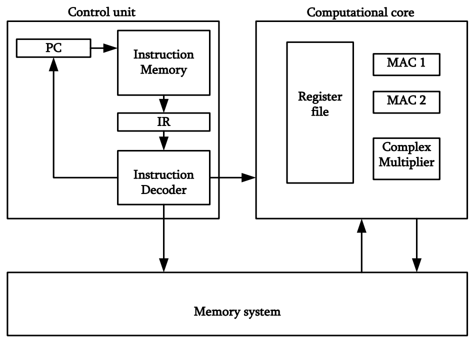

Fig. 4. Processor Architecture: Computational core, Control unit and Memory system

## A. Computational core

The computational core contains two Reconfigurable Multiply Accumulate units (R-MAC), one register file of 16 registers, 16 bits each and one complex multiplier. Fig.5 shows the R-MAC unit. The R-MAC has four real multipliers, two adders and two adder/subtractors and is configured by the control unit through 9 bits control word. It can perform interpolation, filtering, complex multiplication, real multiplication, complex addition and real addition on 14 bits data. The complex multiplier is designed using three real multipliers and performs the complex multiplication on 14 bits data.

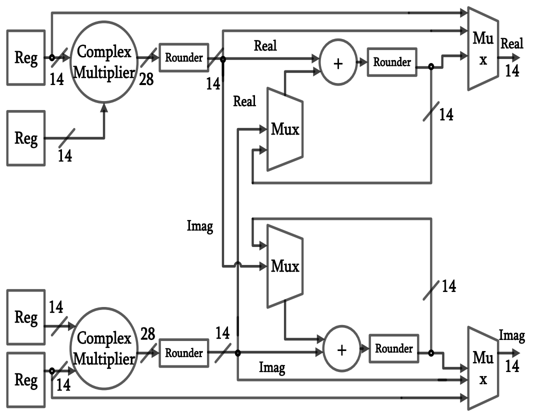

Fig. 5. Reconfigurable Multiply Accumulate Unit (R-MAC)

The use of two R-MAC units is to satisfy the timing requirements of the different standards for each of the algorithms. Fig.6 shows the symbol time in OFDM standards. To define the required frequency of operation and hence the number of R-MAC units, we need to satisfy the timing requirements for the most complex algorithm. The 2-D filtering on two consecutive symbols takes the longest time. The algorithm begins after having two symbols stored and must finish within the symbol time. For 802.16m standards, we find that the required frequency is 240 MHz. Thus, by using two MAC units, the required frequency can be 125 MHz.

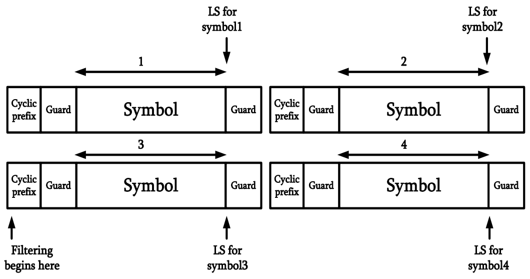

Fig. 6. Timing requirements for 2-D filtering algorithm

## B. Control unit

The control unit includes an instruction memory, program counter (PC), instruction register (IR) and controller. The program counter size is 9 bits. Thus, the instruction memory size will be 512 word, where the word is 77 bits. The controller performs the fetch, decode, execute and write back stages.

## C. Data memory unit

The memory unit includes five memory banks for storing the data and the intermediate results. The banks are dual read/write ports (i.e. The two ports may be used as read or write). The total memory size is 27kword, where the word size is 26 bits with the lowest 13 bits as the real and the highest 13 bits as the imaginary. For N-FFT points, the memory size would be 14N words.

## D. Instruction set

There are three types of instructions: data transfer, data manipulation and control instructions. Table III shows a subset of the instructions used in the processor. The data transfer instructions are used to transfer the different data between memory banks and register file. For instance, the load instruction is used to load a value from the memory into a register in the register file. The control instructions are used to control the flow of the processor. For instance, the WAIT instruction is used to wait until a valid signal comes from the LS separator. Finally, the data manipulation instructions are used to perform the different computations in the system.

TABLE III
INSTRUCTION SET AND THEIR MEANING

| Instruction | Description |
|---|---|
| WAIT | Wait for a valid signal from the LS separator |
| INC | Increment register content |
| DEC | Decrement register content |
| LOAD | Load value into register |
| ADD.N | Adds N values in two memory banks |
| MUL.N | Multiply N values in two memory banks |
| STORE.N | Store N values from LS separator in a certain memory bank |
| COPY.N | Copy N values from memory to memory |
| INTERP.N | Interp N values in two memories and store the result in another memory |
| FILTER.t | Filter N values with filter taps t |
| MAC.N | Multiply accumulate N values |

The processor includes two types of instructions: Scalar and Vector instructions. The vector instructions are used to perform the operation on an array of data. For instance, the INTERP.N instruction is used to perform an interpolation operation on N values from two memory banks and puts the result in a memory bank.

To support a flexible programming, the addressing has two modes: auto-increment and direct addressing. The auto-increment mode is used with the vector instructions, where the specified operation is performed on an array of data that exists in memory banks and the result is put on a different memory bank. This mode was chosen for speed requirements. On the other hand, the direct addressing is used with the scalar instructions like load. Note that the vector instruction can be used as a scalar instruction through assigning a value of 1 to the N parameter in the vector instruction.

## V. PERFORMANCE EVALUATION

The whole system has been coded by Verilog HDL which is compatible with Synopsys Design Compiler. Functional verification was carried out using Altera tools and Mentor-Graphics Modelsim. The processor was implemented in a 0.18 $\mu$m CMOS process at a voltage of 1.8V. The engine occupies $1mm^2$ without memories and consumes 7.4 mWatt at 140 MHz. These results are summarized in table IV.

TABLE IV
SYNTHESIS RESULTS (WITHOUT MEMORIES)

| | |
|---|---|
| **Technology** | TSMC 180 nm CMOS technology |
| **Volt** | 1.8 V |
| **# of cells** | 2825 |
| **Area** | 1 $mm^2$ |
| **Power** | 4.8 mW at 100 MHZ |
| **F$_{max}$** | 140 MHZ |

The system also is synthesized on Altera Stratix III FPGA. The synthesis report of the system has been summarized in Table V.

TABLE V
SYNTHESIS RESULTS

| | |
|---|---|
| **Device Family** | Stratix III |
| **FPGA** | EP3SC150 FPGA |
| **Adaptive look-up tables** | 113600 ALUTs |
| **Embedded Block RAM** | 5499 Kb |
| **Logic utilization** | $<$ 1%(561 ALUTs) |
| **Memory utilization** | 9%(505,856) |
| **DSP Blocks 18 x 18** | 22 |

Power comparisons between the proposed processor and other architectures is shown in table VI. The powers of the stated engines are scaled (technology & frequency scaling) to match the proposed processor. The power is considered without the memory power except for [12].

TABLE VI
POWER COMPARISON BETWEEN THE PROCESSOR AND OTHER ARCHITECTURES

| Architecture | Type | Power |
|---|---|---|
| [5] | Total receiver | 59.5 mW |
| [6] | Reconfigurable channel estimator | 11.3 mW |
| [12] | DSP | 80.18 mW |
| Proposed processor | ASIP | 7.4 mW |

## VI. IMPLEMENTATION SCENARIO

This section discusses an implementation scenario for time interpolation frequency filtering algorithm in 802.16m standard case. The system begins by waiting four valid signals from the LS separator engine. A valid signal comes from the LS separator after it separates one OFDM symbol, removing the guard subcarriers from it. The LS separator separates the pilots from the data into two different memory banks. The processor receives one OFDM symbol after each of the four valid signals. From Fig.2, we note that each symbol needs two other symbols around it for performing the time interpolation to get the in between subcarriers, thus we need to store four symbols for the time interpolation. This would be 8 symbols in case of 3GPP-LTE or DVB systems.

On receiving the four symbols from the LS separator, an LS estimation process is performed by multiplying the pilots by the inverse of the actual transmitted pilot. Thus, storing each of the pilots and data in separate memories.

The interpolation and filtering processes are performed in sequence using the R-MAC units and the results of each process are stored in a memory bank. Finally, a phase correction process is performed by multiplying the data by the conjugate of the estimated channel. This process can be performed by a complex multiplier. The sources and destinations are defined by the instructions and this process is not controlled by specific modes of operation. Thus, it allows a great flexibility in performing different tasks other than channel estimation. This algorithm performs the channel estimation in 94 microsecond at 140 MHz clock speed.

## VII. CONCLUSION

In this paper, we presented a novel ASIP architecture of a reconfigurable processor. The reconfigurability dimensions are investigated and the channel estimation algorithms are presented and compared in terms of performance and computation complexity. The comparison poses the vision of altering the channel estimation algorithm according to the channel parameters. Thus, illuminating the goal of supporting many channel estimation algorithms in our processor. Furthermore, the system is not only limited to the channel estimation operations, but also can support other communication tasks like FFT and FIR operations. The architecture was presented and the different instructions were explored. A processing element is embedded in the computational core so as to support different operations in communication systems.

## REFERENCES

[1] W. Eberle, V. Derudder, G. Vanwijnsberghe, M. Vergara, L. Deneire, L. V. der Perre, M. G. E. Engels, I. Bolsens, and H. D. Man, "An 11 mm2, 70 mw fully programmable baseband processor for mobile wimax and dvb-t/h in 0.12 um cmos," in *IEEE Journal of Solid-State Circuits*, January 2009, pp. 90 – 97.

[2] D. Lo Iacono, J. Zory, E. Messina, N. Piazzese, G. Saia, and A. Bettinelli, "Asip architecture for multi-standard wireless terminals," in *Proceedings Design, Automation and Test in Europe*, March 2006.

[3] K. ElWazeer, M. Khairy, H. A. Fahmy, and S. E. Habib, "FPGA implementation of an improved channel estimation algorithm for mobile WIMAX," in *International Conference on Microelectronics (ICM)*, Dec 2009, pp. 280 – 283.

[4] S. Haene, A. Burg, P. Luethi, N. Felber and W. Fichtner, "Fft processor for ofdm channel estimation," in *IEEE International Symposium on Circuits and Systems*, May 2007, pp. 1417 – 1420.

[5] A. S. Y.Poon, "An energy-efficient reconfigurable baseband processor for wireless communications," in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, March 2007, p. 319.

[6] C. Azar, M. Ojail, S. Chevobbe, and R. David, "CERA: a channel estimation reconfigurable architecture," in *International Conference on Telecommunications (ICT)*, April 2010, pp. 957 – 964.

[7] Y. G. Li and G. L. Stuber, *Orthogonal Frequency Division Multiplexing for Wireless Communications (Signals and Communication Technology)*. Springer; 1st Edition. edition, 2010.

[8] D. C. P. Dong Heon Lee, Suk Chan Kim and Y. il Kim, "A comparative study of channel estimation for mobile WIMAX system in high mobility," in *International Conference on Advanced Communication Technology*, April 2009, pp. 781 – 785.

[9] Jan-Jaap van de Beek, Ove Edfors, Magnus Sandell, Sarah Kate Wilson, and Per Ola Borjesson, "On channel estimation in ofdm systems," in *Proc. of the IEEE Vehicular Technology Conference, VTC 95*, July 1995, p. 815819.

[10] Ye (Geoffrey) Li, Leonard J. Cimini, and Nelson R. Sollenberger, "Robust channel estimation for OFDM systems with rapid dispersive fading channels," in *IEEE Transactions on Communications*, July 1998, pp. 902–915.

[11] L. Hanzo, T. Keller, M. Muenster, and B.-J. Choi, *OFDM and MC-CDMA for Broadband Multi-User Communications, WLANs and Broadcasting*. New York, NY, USA: John Wiley and Sons Inc., 2003.

[12] E. Tell, A. Nilsson, and D. Liu, "A programmable dsp core for baseband processing," in *The 3rd International IEEE-NEWCAS Conference*, June 2005, pp. 403–406.