Modular Design and Implementation of FPGA-based Tap-selective Maximum-likelihood Channel Estimator

JENG-KUANG HWANG *, YUAN-PING LI
* Department of Communication Engineering
Department of Electrical Engineering
Yuan-Ze University
135, Yuan-Tung Rd., Chungli City 32026
TAIWAN
eejhwang@saturn.yzu.edu.tw; s909107@mail.yzu.edu.tw

Abstract: - The modular design of the optimal tap-selective maximum-likelihood (TSML) channel estimator based on field-programmable gate array (FPGA) technology is studied. A novel range reduction algorithm is included in the natural logarithmic function (NLF) emulator based on the coordinate rotation digital computer (CORDIC) methodology and is integrated into the TSML channel estimator system. The low-complexity TSML algorithm, which is employed for sparse multipath channel estimation, is proposed for long-range broadband block transmission systems. Furthermore, the proposed range reduction algorithm aims to solve the limited interval problem in the CORDIC algorithm base on Xilinx's SG platforms. The modular approach facilitates the reuse of modules.

Key-Words: - Coordinate rotation digital computer (CORDIC), FPGA design, Maximum-likelihood channel estimation, Range reduction, Logarithm function, Parallel sorting

1 Introduction

Recently, researches on cyclic-prefix (CP) assisted block transmission systems, particularly orthogonal frequency-division multiplexing (OFDM) and single-carrier with frequency-domain equalization (SC-FDE), have attracted considerable attention. Both these systems are targeted for broadband applications and have been adopted as IEEE 802.16d PHY standards for long-range fixed wireless transmission through multipath fading channels [1]. However, in order to entirely achieve these performance benefits, accurate channel estimation is crucial.

The optimal tap-selective maximum-likelihood (TSML) channel estimator [2], which is based on the maximum-likelihood (ML) and minimum description length (MDL) criteria [4], is proposed for long-range broadband block transmission systems over sparse multipath channels. The TSML estimator can reduce the noise effect and improve the estimation performance, and furthermore dynamically adapt to the instantaneous channel sparsity and asymptotically achieve the optimal performance. For the purpose, the channel estimator can be used in many high-speed, long-range outdoor wireless transmission applications such as WiMax [1], DVB-T [5], and HDTV [6] systems.

Hence, for the above-mentioned applications, a hardware implementation is required. An FPGA-based prototyping approach for the implementation of the channel estimation scheme is presented in [3]. Recent developments in field-programmable gate array (FPGA) technology have changed the conventional methods of hardware implementation. FPGAs have become an alternative solution for the realization of digital systems. They provide a good combination of high-speed implementation features with the flexibility of a digital platform. A considerable amount of research has been conducted on the implementation of reconfigurable algorithms based on FPGAs. An FPGA is a regular structure of logic cells or modules and interconnects, which are entirely under the designer's control. For designing digital signal processing (DSP) systems and digital communication applications, a large number of choices are available to us for implementing our solutions. The choice of using FPGA technology to implement DSP systems as a digital communication solution is approved because of the higher degree of concurrency offered by the gate arrays to a DSP designer. For instance, the Xilinx Virtex-4 XC4VSX35 FPGA [7], which is used to obtain our results, possesses 15,360 slices (34,560 logic cells).

In this paper, a case study is presented in which a modular FPGA-based design approach is applied to design a TSML channel estimator. However, two different design methods are employed: (i) Matlab software for modeling the system and (ii) hardware description language (HDL) for performing hardware implementation on the FPGA. For the above applications, we employ a system generator (SG) [8] to model the DSP projects. There are two important reasons for selecting a system generator [9]: (i) modeling of the complete DSP system and (ii) transforming the theoretical design into a finite precision fixed-point system. This fast system prototyping tool allows designers to observe the effects of their decisions during the design stage.

Furthermore, the complete system is implemented by dividing the system functions into reconfigurable modules. By using reusable and reconfigurable modules, the designer's task in developing DSP systems can be considerably simplified by importing the design into a familiar platform. As a result, the development time required for designing an efficient algorithm is significantly reduced. In addition, the issue on natural logarithmic function (NLF) emulation is discussed. In general, the quick and accurate functioning of the computing logarithm function is a major goal in computer arithmetic and hardware design.

This paper is organized as follows: In Section 2, an overview of a TSML channel estimator algorithm is presented. In Section 3, the design of the components of the FPGA-based modules by using the SG platform is discussed. Section 4 a novel range reduction algorithm and the reconstruction of the NLF is presented. In Section 5, the result of our implementation is reported. The conclusions are presented in Section 6.

Notation: We use a bold uppercase (lowercase) font to denote matrices (column vectors). **F** denotes the $N \times N$ DFT matrix whose (m,n)th element is $[\mathbf{F}]_{mn} \equiv e^{-j2\pi mn/N}$; $\mathbf{I}_N$, the $N \times N$ identity matrix; $(\cdot)^*$, $(\cdot)^{-1}$, $(\cdot)^T$, $(\cdot)^H$, $|\cdot|$, $\|\cdot\|$, $\log(\cdot)$, and $\odot$, the complex conjugate, inverse, transpose, conjugate transpose, absolute value, norm of vector, natural logarithm, and vector pair-wise multiplication operations, respectively; $E\{\cdot\}$, the expectation operator; and $\operatorname{diag}\{\mathbf{a}\}$, a diagonal matrix with the elements of **a** on the diagonal.

2 Overview of the TSML channel estimator algorithm

For fixed wireless applications [1], the composite baseband channel can be modeled as a linear time-invariant system within a small segment of time and is characterized by its impulse response [10]

$$h(\tau) = \sum_{i=0}^{L_p-1} \alpha_i \exp(j\theta_i)\, p(\tau - \tau_i), \qquad (1)$$

where $p(\tau)$ denotes the composite transmit/receive pulse shaping, and L_p denotes the number of propagation paths within the parameters $\{\alpha_i, \theta_i, \tau_i\}$, which denote the attenuation, phase shift, and delay of the ith path, respectively. For a discrete-time equivalent channel model, sampling $h(\tau)$ at symbol rate $1/T_s$ provides the channel response vector $\mathbf{h}_L = [h(0), \cdots, h(L-1)]^T$, where $L = \lceil \tau_{\max}/T_s \rceil$ is the channel length for a maximum delay spread $\tau_{\max}$. Furthermore, since we concentrate on broadband and high-speed applications, $\mathbf{h}_L$ is assumed to be a sparse channel. This means that only some channel taps in $\mathbf{h}_L$ are significant, while the other channel taps have nominal values of zero.

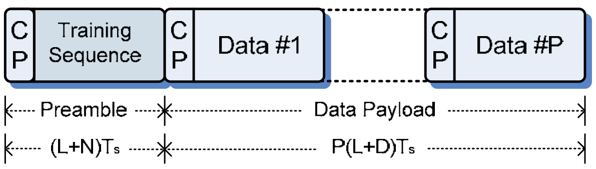

Fig. 1. Frame structure of the CP-based single-carrier block transmission.

We consider the channel estimation problem with regard to the SC-FDE block transmission system. As shown in Fig. 1, the transmission frame comprises one preamble block and P data payload blocks, where L, N, and D denote the lengths in symbols for the CP, training sequence, and a single data payload block, respectively.

We begin by formulating a channel estimation problem for the CP-based single-carrier system. Let the time-domain N-point training sequence $\mathbf{t} = [t(0), t(1), \cdots, t(N-1)]^T$ be appended by L-point CP; this results in a total preamble duration of $(L+N)T_s$. Assume that the signal is passed through an unknown discrete-time channel $\mathbf{h}_L$, where the maximum channel length is less than the CP length L. At the receiver, after removing the CP and computing the N-point FFT, the received frequency-domain signal block $\mathbf{r} = [r(0), r(1), \cdots, r(N-1)]^T$ can be expressed as

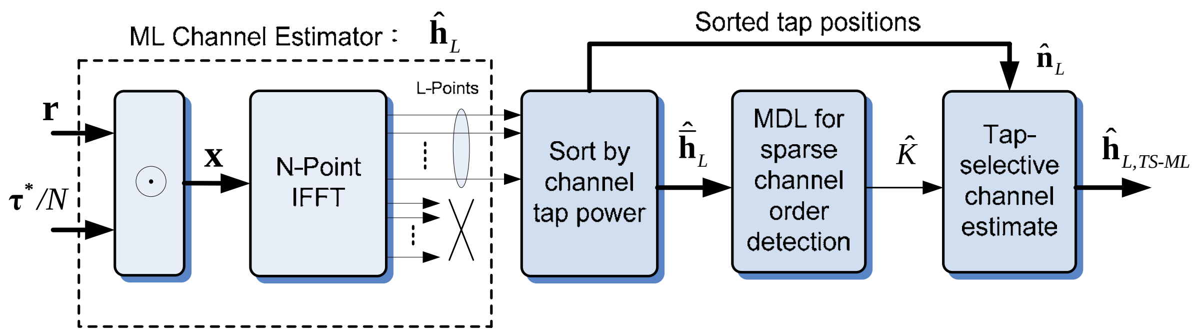
ML Channel Estimator : $\hat{\mathbf{h}}_L$
Sorted tap positions
$\mathbf{r}$
$\boldsymbol{\tau}^*/N$
$\odot$
$\mathbf{x}$
N-Point IFFT
L-Points
Sort by channel tap power
$\hat{\mathbf{h}}_L$
MDL for sparse channel order detection
$\hat{K}$
$\hat{\mathbf{n}}_L$
Tap-selective channel estimate
$\hat{\mathbf{h}}_{L,TS\text{-}ML}$

Fig. 2. Block diagram of the TSML channel estimator with the MDL criterion for sparse channel order detection.

$$\mathbf{r} = \mathbf{T}\mathbf{F}\mathbf{h}_N + \mathbf{F}\mathbf{v} = \mathbf{T}\mathbf{F}_L\mathbf{h}_L + \mathbf{F}\mathbf{v}, \qquad (2)$$

where $\mathbf{F} = [\mathbf{f}_0\ \mathbf{f}_1\ \cdots\ \mathbf{f}_{N-1}]$ is the $N \times N$ Fourier matrix for representing the N-point FFT; $\mathbf{h}_L = [h(0),\cdots,h(L-1),0,\cdots,0]^T$, the channel vector of length N with $(N–L)$ appended zeros; $\mathbf{F}_L = [\mathbf{f}_0\ \mathbf{f}_1\ \cdots\ \mathbf{f}_{L-1}]$, the $N \times L$ submatrix of $\mathbf{F}$; $\mathbf{T} = \operatorname{diag}\{\boldsymbol{\tau}\} = \operatorname{diag}\{[\tau(0),\ \cdots,\ \tau(N\text{-}1)]^T\}$, a diagonal matrix formed by $\boldsymbol{\tau} = \mathbf{F}\mathbf{t}$, i.e., the FFT of $\mathbf{t}$; and $\mathbf{v}$, an $N \times 1$ complex white Gaussian noise vector with covariance $\sigma^2\mathbf{I}_N$.

In order to facilitate the estimation of channel response $\mathbf{h}_L$, we consider the Chu-sequence [11] as the training sequence to satisfy the constant modulus property in both the time and frequency domains, i.e., $|t(n)| = 1$ and $|\tau(k)| = \sqrt{N}$ for $n,k \in \{0,1,\cdots,N-1\}$. By performing a pairwise multiplication of (2) with $\boldsymbol{\tau}^*/N$, we obtain an equivalent received data vector for the channel estimation:

$$\mathbf{x} = \mathbf{r} \odot \left(\boldsymbol{\tau}^*/N\right) = \frac{1}{N}\mathbf{T}^H\mathbf{r} = \mathbf{F}_L\mathbf{h}_L + \mathbf{w}, \qquad (3)$$

where the new Gaussian noise vector $\mathbf{w} = \mathbf{T}^H\mathbf{F}\mathbf{v}/N$ has the same covariance matrix, $\mathbf{C}_w = \sigma^2\mathbf{I}_N$, as $\mathbf{v}$. Thus, the transformed data vector $\mathbf{x}$ is itself a raw estimate of the N-point channel frequency response vector $\mathbf{F}_L\mathbf{h}_L$. Since (3) is a linear model, the ML channel estimate $\hat{\mathbf{h}}_L = [\hat{h}(0)\ \hat{h}(1)\ \cdots\ \hat{h}(L-1)]^T$ of the time-domain response vector $\mathbf{h}_L$ coincides with the least squares (LS) solution:

$$\hat{\mathbf{h}}_L = \left(\mathbf{F}_L^H\mathbf{F}_L\right)^{-1}\mathbf{F}_L^H\mathbf{x} = \frac{1}{N}\mathbf{F}_L^H\mathbf{x} = \mathbf{h}_L + \frac{\mathbf{F}_L^H\mathbf{w}}{N}, \qquad (4)$$

which is simply obtained from the first L points of the N-point inverse fast Fourier transform (IFFT) of the data vector $\mathbf{x}$.

However, for a sparse channel, the original estimation problem should be formulated as a combined detection-estimation problem [12,13]. Hence, the so-called TSML channel estimator is derived in order to solve the above combined detection-estimation problem. The derivation is based on the separation of parameters, and an efficient TSML algorithm is also obtained. Under a sparse channel situation, we show that the proposed estimator can be employed with significant improvement in the estimation accuracy. The block diagram of the overall TSML channel estimator is shown in Fig. 2, and the proposed algorithm is summarized in Table 1.

The TSML channel estimator algorithm can reduce the estimation error in MSE by an improvement factor of L/K. If some taps have small values, the tap-selective process will automatically reduce the channel order such that the noise effect is alleviated. Hence, we also regard the TSML estimator as an adaptive and robust channel estimation method.

3 Design of FPGA-based modules for the TSML channel estimator

With regard to the previously mentioned advantages of the proposed estimator, a study of the hardware implementations for high-speed, long-range outdoor wireless transmission applications such as, WiMAX, DVB-T, and HDTV systems is significant. In this paper, the modular design of the optimal TSML

Table 1. TSML channel estimator algorithm.

---

1 From the received training block $\mathbf{r}$, compute the pairwise multiplication of $\mathbf{r}$ and $\boldsymbol{\tau}^*/N$: $\mathbf{x} = \mathbf{r} \odot (\boldsymbol{\tau}^*/N)$, where $\boldsymbol{\tau} = \mathbf{F}\mathbf{t}$.

2 Determine the IFFT of $\mathbf{x}$, and take the first L points of the result as $\hat{\mathbf{h}}_L = [\hat{h}(0)\ \hat{h}(1)\ \cdots\ \hat{h}(L-1)]^T$.

3 Sort every element of the ML estimate $\hat{\mathbf{h}}_L$ in the descending order of power, i.e., $\hat{\bar{\mathbf{h}}}_L = \left[\hat{h}(\hat{n}_1)\ \hat{h}(\hat{n}_2)\ \cdots\ \hat{h}(\hat{n}_L)\right]^T$, where $\left|\hat{h}(\hat{n}_1)\right|^2 \geq \left|\hat{h}(\hat{n}_2)\right|^2 \geq \cdots \geq \left|\hat{h}(\hat{n}_L)\right|^2$, and $\hat{\mathbf{n}}_L = [\hat{n}_1, \hat{n}_2, \cdots, \hat{n}_L]^T$ denotes the sorted position vector.

4 For the k^{th} sparse channel model, we have $MDL(k) = N \log\left(\|\mathbf{x}\|^2 - N\left\|\hat{\bar{\mathbf{h}}}_k\right\|^2\right) + \frac{3k}{2}\log N$, for $k = 1, 2, \cdots, L$. The calculation of $MDL(k)$ can be performed recursively by $\left\|\hat{\bar{\mathbf{h}}}_k\right\|^2 = \left\|\hat{\bar{\mathbf{h}}}_{k-1}\right\|^2 + \left|\hat{h}(\hat{n}_k)\right|^2$ with the initial condition $\left\|\hat{\bar{\mathbf{h}}}_0\right\|^2 = 0$, where $\hat{\bar{\mathbf{h}}}_k = \left[\hat{h}(\hat{n}_1)\ \ \hat{h}(\hat{n}_2)\ \ \cdots\ \ \hat{h}(\hat{n}_k)\right]^T$, $\hat{\mathbf{n}}_k = [\hat{n}_1, \hat{n}_2, \cdots, \hat{n}_k]^T$, and k denote the values of the first k significant elements of the non-sparse ML estimate $\hat{\mathbf{h}}_L$.

5 The number of significant channel taps is determined as $\hat{K} = \arg \min_{k \in (1,2,\cdots,L)} MDL(k)$.

6 We can directly construct the TS-ML estimate $\hat{h}_{L,TS\text{-}ML}$ by using $\hat{h}_{L,TS\text{-}ML}(n) = \begin{cases} \hat{h}(n), & n \in \hat{\mathbf{n}}_{\hat{K}} \\ 0, & n \notin \hat{\mathbf{n}}_{\hat{K}} \end{cases}$, for $n = 0, 1, \cdots, L-1$. If all the identified channel order and tap positions are correct, the MSE becomes $MSE_{TS-ML} = E\left\{\left\|\hat{\mathbf{h}}_{L,TS-ML} - \mathbf{h}_L\right\|^2\right\} = \frac{K\sigma^2}{N}$

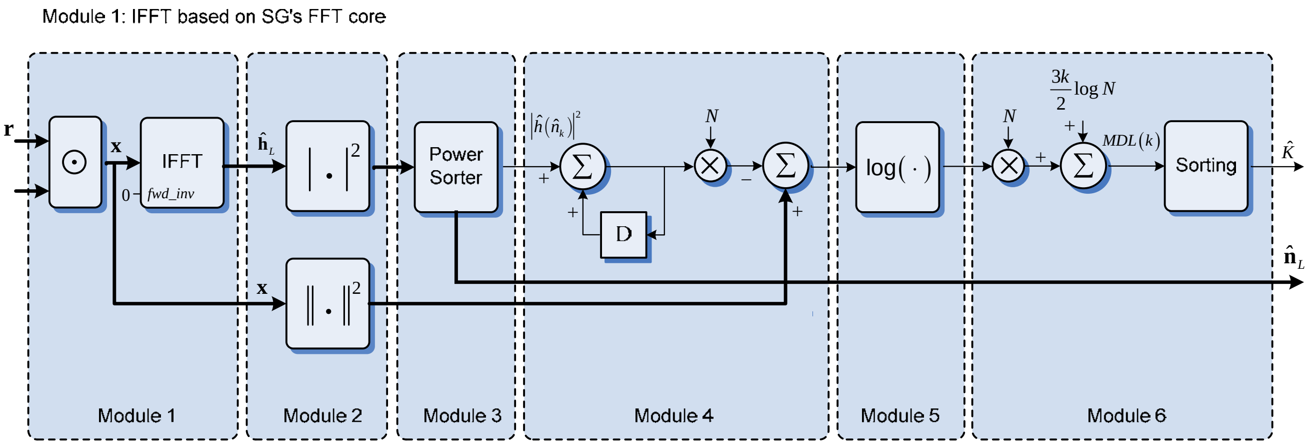

Fig. 3. Architecture of the modules for the TSML channel estimator.

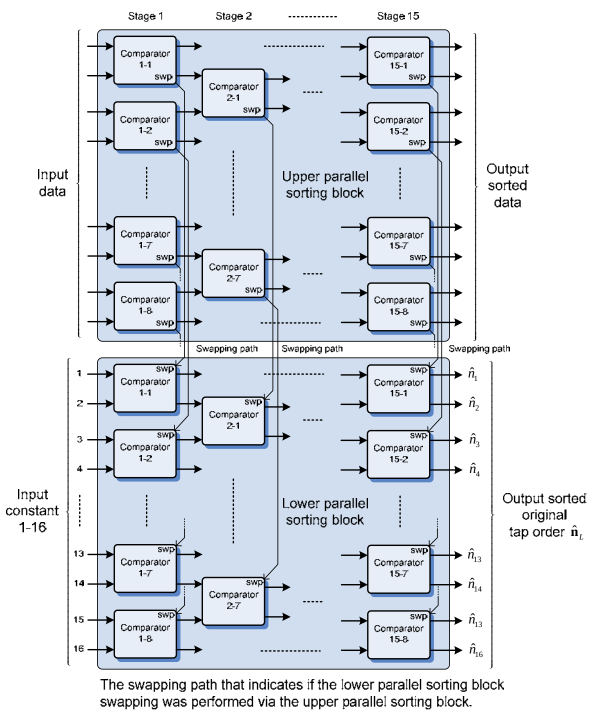
Stage 1
Stage 2
Stage 15
Input data
Upper parallel sorting block
Output sorted data
Swapping path
Input constant 1-16
Lower parallel sorting block
Output sorted original tap order $\hat{\mathbf{n}}_L$
The swapping path that indicates if the lower parallel sorting block swapping was performed via the upper parallel sorting block.

Fig. 4. Parallel sorting architecture.

TSML algorithm, which determines the organization of the course contents in the six modules. Each block corresponds to a module, as shown in Fig. 3. In the following implementation, we set the CP length as $L = 16$ and use the shortest preamble with $N = 32$. The sparse channel order is set as $K = 3$.

The TSML channel estimator employs IFFT to obtain the time domain components of the signal; thus, the first module covers the implementation issues of the IFFT algorithm. The difference in the *fwd_inv* port of the input interface is set to zero, while the inverse transform is selected for the SG's FFT block; furthermore, the pipelined, streaming input/output implementation mode is employed for allowing continuous data processing. Finally, the Xilinx FFT Core [14] parameters are enumerated, and the use of the SG's FFT core is described. The channel estimator for this module must satisfy two essential conditions:

(i) The FFT core configuration according to the real-time TSML channel estimator's specifications.

(ii) The generation of the enable signal for the FFT block (required to adapt the processing frequency).

In order to compute the magnitude of the complex signals $\hat{\mathbf{h}}_L$ and $\mathbf{x}$, the second module employs the coordinate rotation digital computer (CORDIC) algorithm [15–18], which is introduced as an iterative algorithm that requires only adder-subtractors and shifters. The functioning of this module typically consists of three steps:

(i) A CORDIC algorithm is used for rectangular to polar conversion of the transformed values. In other words, for a given complex input, the magnitude and the angle are computed. Furthermore, the magnitude scale factor is not compensated in the processor, i.e., the magnitude output should be scaled by this gain factor with a constant coefficient multiplier.

(ii) In the module, the power of the magnitudes must be generated by a multiplier. The product of the data is computed on the two connected input ports of the multiplier and the power result is obtained on its output port.

(iii) On the other hand, according to steps (i) and (ii), the power of magnitudes is generated in the same manner for data $\mathbf{x}$ before the SG's FFT block.

The third module focuses on the implementation of the sorting data for the non-sparse ML estimate $\hat{\mathbf{h}}_L$ in the descending order of power. A direct hardware implementation of quick sort-type algorithms would require sophisticated control units. In order to tackle this problem, simpler sorting algorithms are used. For the purpose, we employ a parallel sorting architecture such as [19]. The parallel architecture, which is based on comparators of dotted lines, indicates that all the comparisons and swaps are performed. Thus, in this manner, for an array of L data, the number of steps required for a sorting stage is $L-1$. In this study, we employ 16 data in order to explain the manner in which parallel sorting operates. Figure 4 shows an example of an array of 16 data and 15 stages. Two parallel sorting structures have been presented:

(i) The comparator block of the upper parallel sorting block is shown in Fig. 5a. The inputs of this block are (1) datum a; (2) datum b. The

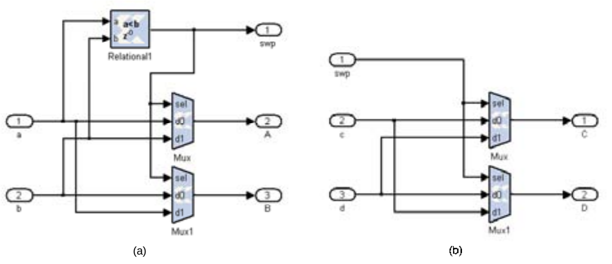

Fig. 5. (a) Comparator block of the upper parallel sorting block and (b) Comparator block of the lower parallel sorting block.

outputs of this block are (1) a *swp flag* signal that indicates if swapping was performed, (2) datum A, and (3) datum B with the corresponding value that indicates whether or not swapping occurred. In addition, the equal data will remain unchanged.

(ii) Another comparator block of the lower parallel sorting block is shown in Fig. 5b. This block contains two multiplexers. The inputs of this block are (1) a *swp flag* signal that indicates if swapping was performed, (2) datum c, and (3) datum d. The outputs of this block are (1) datum C and (2) datum D. These outputs will be swapped depending on the status of the *swp flag* signal. The datum c and datum d inputs are connected with two fixed constants in order to record the original tap orders for the TSML algorithm.

The sorting is completed when all *swp flags* are set to zero, i.e., swapping was not performed in the all the levels of the comparators.

The fourth module works with the recursive relation updated equation, which is introduced by using an adder-based accumulator architecture, as a method of generating recursion in the input path. We compute $\left\|\hat{\mathbf{h}}_k\right\|^2$ under the initial condition $\left\|\hat{\mathbf{h}}_0\right\|^2 = 0$. Furthermore, the completed $\left\|\hat{\mathbf{h}}_k\right\|^2$ must be multiplied by *N*, where *N* = 32. Multiplication by 2^5 ($32 = 2^5$) corresponds to a 5-bit left shift. For example, consider a decimal-to-binary equivalence of $3.390625_{10} = 11.011001_2$, and the result when multiplied by 32 will be $108.5_{10} = 1101100.1_2$. Finally, subtract $N \cdot \left\|\hat{\mathbf{h}}_k\right\|^2$ from $\left\|\mathbf{x}\right\|^2$ of the second module.

Because the TSML channel estimator employs the MDL criterion, two parts of the equation can be simplified; this includes the natural logarithm of the recursive relation updated equation (the fourth module) and the constant term. The main equation for MDL is illustrated in step 4 of Table 1. In the fifth module, the natural logarithm is computed, which is represented in Section 4.

The last module comprises three components—a constant coefficient multiplier, an adder based accumulator, and a parallel sorting block. Because *N* =32, we can replace the constant coefficient multiplier with a 5-bit left shift. Moreover, $(3 \times k \times \log N)/2 = (3/2) \times \log 32 \times k$, where $k = 1, 2, \cdots, L$ and $L = 16$. Hence, the recursive relation can be realized by an adder-based accumulator. Because of the condition $\hat{K} = \arg \min_{k \in (1,2,\cdots,L)} MDL(k)$, a parallel sorting is required; for example, the upper parallel sorting block of the third module. Finally, we can directly obtain $\hat{K}$ and construct the TSML estimate $\hat{h}_{L,TS\text{-}ML}$.

4 A novel natural logarithmic function emulator implementation based on the FPGA module

The NLF is extensively employed in digital signal processing, communication systems, control systems, etc. In general, computing the function efficiently and accurately is a major target in computer arithmetic and hardware design. Software implementations are often too slow for numerically intensive or real-time applications. Hence, the hardware implementation of an efficient emulation is required.

For this purpose, a NLF evaluator typically comprises range reduction and the actual function approximation over a small interval. Range reduction [17,20,21] is crucial because function approximation is rather limited without it and numerous applications have a large dynamic range. However, sufficient attention has not been provided for the hardware implementation of function approximation with range reduction for different ranges, precisions, and approximation methods. This is the first study that deals with this important issue. We define the sign bit and the integer bits to indicate the range and the fractional bits to indicate the precision for the input range. The fixed-point bit format representation for the designs is used in this study and shown in Fig. 6.

Figure 7 shows the overall block diagram of the NLF evaluator based on the FPGA module. The design of a more suitable NLF evaluation for the TSML channel estimator includes automating the selection of range reduction and the design of the logarithm function evaluation method. When implementing hardware designs, any bit-width can be selected for the range and the precision of the fixed-point number. As a result, a NLF evaluation obtains the range and precision of the input and this information can be used to obtain a more suitable NLF evaluation unit for the TSML channel estimator. Thus, based on the input range and precision, the following three procedures are performed:

(i) Approximation method selection.
(ii) Applicability of range reduction.
(iii) Evaluation method.

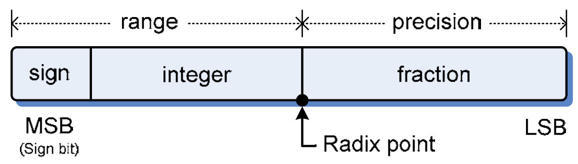

Fig. 6. Binary fixed-point representation used in this study.

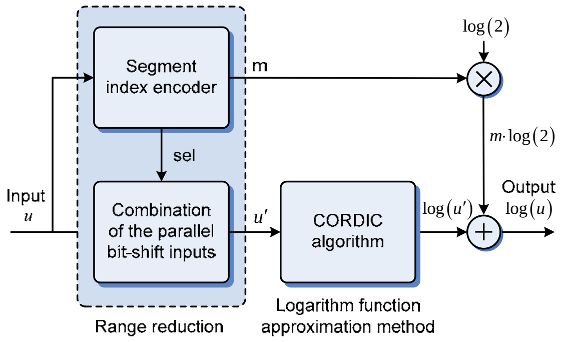

Fig. 7. Block diagram of the NLF evaluator based on the FPGA module.

4.1 Approximation method selection

There are many possible function evaluation methods. Each method has its strengths and weaknesses. However, direct table lookups are impractical for precisions higher than a few bits since table size increases exponentially with the input. We employ the CORDIC algorithm for designing the function evaluation due to its popularity in research topics and because it involves only shift-and-add operations.

The low-complexity TSML algorithm is applied in high-speed, long-range outdoor wireless transmission applications. The CORDIC algorithm is more suitable for the channel estimator to obtain the natural logarithm. Because the CORDIC algorithm is a collection of iterative shift-and-add algorithms, which provide an extremely efficient means of computing the logarithmic function. In addition, the approximation error analysis with the logarithm function based on the CORDIC algorithm is presented in [18,22].

However, for natural logarithms, the range of the valid input with a CORDIC processor is limited in a domain of convergence for a fixed-point format, for example, $[0.5,1)$ for SG [8] platforms based on Xilinx FPGAs. Hence, there has been a lack of attention on the hardware implementation of the CORDIC approximation method for a large dynamic range. Therefore, an equation can be used to assist the computation according to the fundamental property of a logarithmic function. It is known that

$$\log(u) = \log(u') + m\cdot\log(2), \tag{5}$$

where $u = u'\cdot 2^m$. By using this equation, the natural logarithm can be computed by using a CORDIC algorithm and adding an additional multiplier constant. As a result, we can design the applicability of range reduction.

4.2 Range reduction

In order to design the automation of the selection of the range reduction for a large dynamic input range, consider the function $\log(u)$ from (5), where u has a given range $[s,e)$. The function approximation processor has a valid input range $[a,b)$. The NLF estimator typically comprises three steps [21]:

(i) Range reduction: reducing input u over the interval $[a,b)$ to a more convenient output u' in an interval $[a',b')$ smaller than $[a,b)$.
(ii) Function approximation emulator on the reduced interval based on the CORDIC method.
(iii) Range reconstruction: expansion of the result back to the original result range.

The multiplicative range reduction for the natural logarithm occurs mainly as follows: u is equal to $u'\cdot 2^m$, where integer m is defined by the m-bit left shift or right shift. The range reduction follows the concept presented in [20]. For a power of two, multiplication by 2^m corresponds to an m-bit left shift, and multiplication by 2^{-m} corresponds to an m-bit right shift. The input and output transformations can exploit the bit-shift for simple multiplication. The range reconstruction follows the concept presented in [17].

As an example to illustrate our approach, consider a given range $u=[s,e)$ and 12 inputs assigned to a 1-output multiplexer. Table 2 shows the range reduction algorithm for the domain. The inputs for this algorithm are a numerical function $\log(u)$ and a domain $[s,e)$ for u. The given domain $[s,e)$ is partitioned into twelve segments $[s_0,e_0)$, $[s_1,e_1)$,..., $[s_{11},e_{11})$. The interval of every segment is defined in Table 3 based on the valid input range $[0.5,1)$ for $[a,b)$ of the CORDIC algorithm base on Xilinx's SG platforms; the

Table 2. Range reduction algorithm for the domain.

Input: Numerical function $\log(u)$. Domain $[s, e)$ for u.

Output: A more convenient u' in an interval $[a', b')$ smaller than $[a, b)$, where $[a, b)$ is a small, natural evaluation interval based on CORDIC.

Process: This is range reduction procedure.

1. Let u be an input in parallel.
2. Partition the given domain $[s, e)$ into twelve segments $[s_0, e_0)$, $[s_1, e_1)$, ..., $[s_{11}, e_{11})$.
3. Let us input the combination of twelve bit-shift segments into a 12-input to 1-output multiplexer in parallel.
4. The multiplexer will select a correct interval for the 12 inputs in parallel via a *sel* index of segment index encoder output.
5. The output of the multiplexer will be modified to a more convenient u' over a smaller interval.

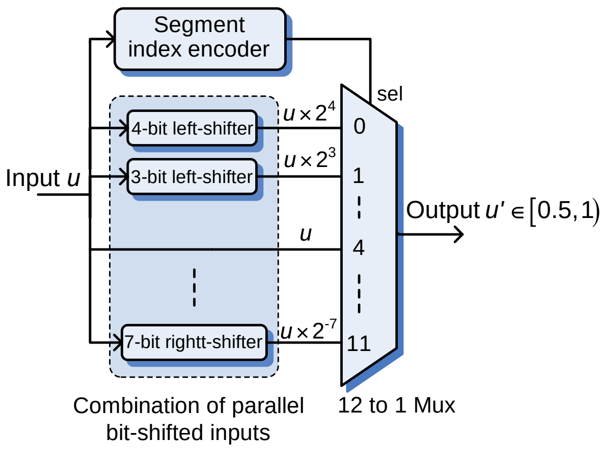

Fig. 8. Architecture of the combination of the bit-shift segments in parallel.

combination of the bit-shift segments in parallel is shown in Fig. 8.

Table 3 shows the segment index encoder. A corresponding input value u is converted into a multiplexer select index (*sel*) and a bit-shift index for the interval of the corresponding segment, and

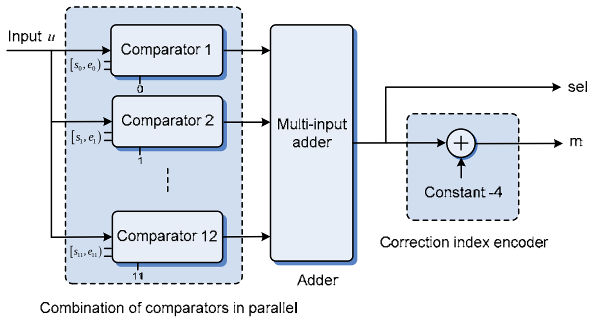

Fig. 9. Architecture of the segment index encoder.

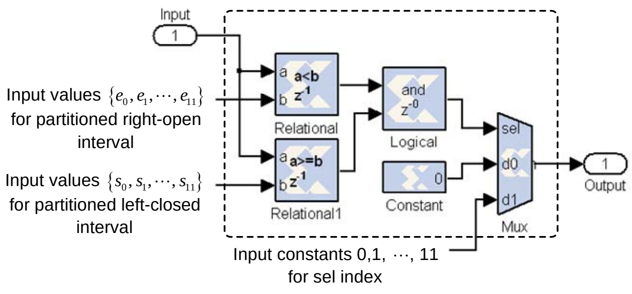
Input
1
Input values $\{e_0, e_1, \cdots, e_{11}\}$ for partitioned right-open interval
Input values $\{s_0, s_1, \cdots, s_{11}\}$ for partitioned left-closed interval
a<b
Relational
a>=b
Relational1
and
Logical
0
Constant
sel
d0
d1
Mux
1
Output
Input constants 0,1, ···, 11 for sel index

Fig. 10. Comparator block.

the same input value is computed into a correction index (m) from the *sel* index for the same interval.

In Fig. 9, in order to illustrate the segment index encoder, the values of $\{s_0, s_1, \ldots, s_{11}\}$ (left-closed interval) and $\{e_0, e_1, \ldots, e_{11}\}$ (right-open interval) are restricted to values that can be produced by a simple combination of twelve comparators in parallel. A more detailed comparator block is shown in Fig. 10. The appropriate taps are obtained from the parallel comparators depending on the choice of the segments and are added to compute the *sel* index and *m* index for an added constant -4. The obtained *sel* index can be assigned for the selection of the multiplexer inputs. Furthermore, using the *m* index can compensate the calculated $\log(u')$ by the CORDIC algorithm for reconstructing the NLF according to (5).

4.3 Evaluation method

The central contribution of this section lies in designing and automating the selection of range reduction for a large dynamic input range and solving a limited interval of the logarithmic function approximation for the CORDIC algorithm based on SG platforms using Xilinx FPGAs.

Table 3. Segment index encoder.

Interval		Bit-shift index	Mux select index (sel)	Correction index (m)
$s_0 \le u < e_0$	$[s_0, e_0) = [a \cdot 2^{-4}, b \cdot 2^{-4})$	4-bit left shift	0	–4
$s_1 \le u < e_1$	$[s_1, e_1) = [a \cdot 2^{-3}, b \cdot 2^{-3})$	3-bit left shift	1	–3
⋮	⋮	⋮	⋮	⋮
$s_4 \le u < e_4$	$[s_4, e_4) = [a, b)$	do not modify	4	0
⋮	⋮	⋮	⋮	⋮
$s_{11} \le u < e_{11}$	$[s_{11}, e_{11}) = [a \cdot 2^{7}, b \cdot 2^{7})$	7-bit right shift	11	7

Note: $[a, b)$ is assigned by a valid interval based on CORDIC to equal $[0.5, 1)$.

Table 4. Resource utilization report.

Device Selected	XC4VSX35		
Features	Used	Avail	Utilization (%)
Slices	8228	15360	53.6
Flip Flops	6322	30720	20.6
LUTs with 4 input	13899	30720	45.2
Bonded IOBs	22	448	4.9
BUFGs	1	32	3.1
RAMB16s	10	192	5.2
DSP48s	6	192	3.1
Max. path delay: 57.8 ns	Max. Clock Freq.: 17.3 MHz		

5 Result of our implementation

The hardware platform used for implementing the channel emulator is Xilinx XtremsDSP DK4 board [23], which hosts a Virtex-4 XC4VSX35 FPGA [7]. We used the top-down design flow based on the Simulink and SG software tools for fast cosimulation. The resource utilization report for this implementation is presented in Table 4.

6 Conclusion

In this paper, an FPGA-based channel estimation system using an optimal TSML algorithm was presented. On the other hand, a novel method for the range reduction and reconstruction and subsequent logarithm function based on the CORDIC method using Xilinx's SG platforms, solution of a limited interval, and integration into the TSML channel estimator was also presented. Based on the module scheme of the dividing system functions, the proposed TSML channel estimator reduces the complexity of the FPGA design. In addition, due to the flexibility of the module in the FPGA, this FPGA-based TSML channel estimator can be easily extended to incorporate other algorithms such as equalizer or adaptive schemes.

The entire system was designed using a modular approach and integrated and downloaded into Xilinx FPGA chips. The modules are reusable and reconfigurable, which can be ported into Matlab/Simulink as Simulink blocks for hardware/software cosimulation and can be utilized in other applications. In future, we will continue to optimize these components such as the sorting data and NLF modules for integration into the TSML channel estimator. More components will be developed if necessary.

References:

[1] *IEEE standard for local and metropolitan area networks Part 16: Air interface for fixed broadband wireless access systems*, IEEE Std. 802.16^{TM}-2004, Oct. 2004, pp. 1–857.

[2] J.K. Hwang, R.L. Chung, Low-complexity algorithm for tap-selective maximum likelihood estimation over sparse multipath channels, *Proc. 50th GLOBECOM*, 26–30 Nov. 2007, pp. 2857–2862.

[3] J.K. Hwang, Y.P. Li, Modular design and implementation of FPGA-based tap-selective maximum-likelihood channel estimator, *Proc. IEEE ICCSC 2008*, May 26–28, 2008, pp. 658–662.

[4] M. Wax, T. Kailath, Detection of signals by information theoretic criteria, *IEEE Trans.*

Acoustics, Speech, and Signal Processing, Vol. 33, Apr. 1985, pp. 387–389.

[5] Digital video broadcasting (DVB); Framing structure, channel coding and modulation for terrestrial television, European Standard (EN) 300 744 V1.5.1, European Telecomm. Standards Institute (ETSI), Nov. 2004.

[6] W.F. Schreiber, Advanced television systems for terrestrial broadcasting: Some problems and proposed solutions, *Proc. IEEE*, Vol. 83, June 1995, pp. 958–981.

[7] Virtex-4 Family Overview, Xilinx Inc., 2004, http://www.xilinx.com.

[8] Xilinx System Generator for DSP, v 8.1 User's Guide, Xilinx, Inc., San Jose, CA.

[9] D. Turney, et al., Modeling and Implementation of DSP FPGA Solutions, White Paper, 2000. Available at: http://www.xilinx.com.

[10] T.S. Rappaport, *Wireless Communications: Principles and Practice*, second ed., Prentice Hall, New Jersey, 2002.

[11] D.C. Chu, Polyphase codes with good periodic correlation properties, *IEEE Trans. Inform. Theory*, Vol. IT-18, July 1972, pp. 531–532.

[12] J.K. Hwang, Y.C. Chen, A combined detection-estimation algorithm for harmonic-retrieval problem, *Signal Processing*, Vol. 30, Jan. 1993, pp. 177–197.

[13] J.K. Hwang, Y.C. Chen, Superresolution frequency estimation by alternating notch periodogram, *IEEE Trans. Signal Processing*, Vol. 41, Feb. 1993, pp. 727–741.

[14] Fast Fourier Transform v3.1 Xilinx Core. DS260 Nov. 2004, Xilinx Product Specifications [Online]. Available at: http://www.xilinx.com.

[15] R. Andraka, A survey of CORDIC algorithms for FPGA-based computers, *Proc. ACM/SIGDA Int. Symp. Field-Program. Gate Arrays*, 1998, pp. 191–200.

[16] Y.H. Hu, CORDIC-based VLSI architectures for digital signal processing, *IEEE Signal Processing Magazine*, July 1992, pp. 17–34.

[17] W. Ligon, G. Monn, D. Stanzione, F. Stivers, K. D. Underwood, Implementation and analysis of numerical components for reconfigurable computing, *Proc. IEEE Aero. Conf.*, Vol. 2, 6-13, Mar. 1999, pp. 325–335.

[18] A. Boudabous, F. Ghozzi, M.W. Kharrat, N. Masmoudi, Implementation of hyperbolic functions using CORDIC algorithm, *Proc. ICM 16th Int. Conf. Microelectronics*, 6-8, Dec. 2004, pp. 738–741.

[19] J. Martinez, R. Cumplido, C. Feregrino, An FPGA-based parallel sorting architecture for the Burrows Wheeler transform, *Proc. IEEE Int. Conf. Reconfigurable Computing and FPGAs*, 28-30 Sept. 2005, pp. 7.

[20] M.J. Schulte, E.E. Swartzlander Jr., Hardware designs for exactly rounded elementary functions, *IEEE Trans. Computers*, Vol. 43, No. 8, 1994, pp. 964–973.

[21] D. Lee, A.A. Gaffar, O. Mencer, W. Luk, Optimizing hardware function evaluation, *IEEE Trans. Computers*, Vol. 43, No. 8, Aug. 1994, pp. 964–973.

[22] X. Hu, R.G. Harber, S.C. Bass, Expanding the range of convergence of the CORDIC algorithm, *IEEE Trans. Computers*, Vol. 40, No. 1, 1991, pp. 13–21.

[23] XtremeDSP Development Kit-IV User Guide, Xilinx Inc., Issue 2, 2005. Available at: http://www.xilinx.com.