Microelectronics Journal 45 (2014) 167–178

Contents lists available at ScienceDirect

Microelectronics Journal

journal homepage: www.elsevier.com/locate/mejo

Design and verification of a frequency domain equalizer

Anis Souari [a], Amjad Gawanmeh [b], Sofiène Tahar [a,*], Mohamed Lassaad Ammari [c]

[a] Department of Electrical and Computer Engineering, Concordia University, Montreal, Quebec, Canada H3G 1M8
[b] Department of Electrical and Computer Engineering, Khalifa University of Science, Technology and Research, P.O. Box 573 Sharjah, United Arab Emirates
[c] École Nationale d'ingénieurs de Sousse, Sousse, Tunisia

ARTICLE INFO

Article history:
Received 20 May 2013
Received in revised form
20 October 2013
Accepted 23 October 2013
Available online 21 November 2013

Keywords:
Error analysis
Theorem proving
Design and verification

ABSTRACT

In this work we provide a methodology for the design and verification of a frequency domain equalizer. The performance analysis of the equalizer is conducted using two methods: simulation based verification in Simulink and System Generator and theorem proving techniques in Higher Order Logic. We conduct both floating-point and fixed-point error estimations for the design in Simulink and System Generator, respectively. Then, we use formal error analysis based on the theorem proving to verify an implementation of the frequency domain equalizer based on the Fast LMS algorithm. The formal error analysis and simulation based error estimation of the algorithm intend to show that, when converting from one number domain to another, the algorithm produces the same values with an accepted error margin caused by the round-off error accumulation. This work shows the efficiency of combining simulation and formal verification based methods in verifying complex systems such as the frequency domain equalizer.

1. Introduction and motivation

With the recent technological growth, electronic devices have invaded all aspects of our lives. These devices are getting more and more compact and consequently more complex. The price of this complexity is the challenge of delivering error-free devices, which require thorough testing and verification at all stages of the design flow. On the other hand, a faulty design can lead to costly delays for the time-to-market. Therefore, design verification is necessary to avoid such situations and is considered a bottleneck in the design process. In order to verify that an implementation meets its specifications, simulation is the most widely used technique in the industry, because it is straightforward and does not need any expertise. This simulation is based on the generation of test patterns, and therefore, it does not provide full coverage for the system under test. On the other hand, formal verification techniques [1] are considered complementary to simulation as they can provide full coverage for the system under test, and in addition, they can catch corner case bugs in the design. Formal verification does require a certain level of expertise to be efficiently used, which may incur a considerable human resources cost.

Equalization is an application of adaptive filtering that can eliminate the inter-symbol interference caused by the noise in the transmission environment. Uncountable adaptive algorithms are used to regulate the filter or the equalizer coefficients in order to match the output to the desired response. To decrease the filtering complexity, the equalizer can be implemented in the frequency domain using the Fast Fourier Transform (FFT) and the Inverse FFT (IFFT), where time convolution is replaced by frequency multiplication. This method offers low complexity growth in comparison with the time domain method. Data processing and filtering require dealing with data at different domains: real numbers, floating-point numbers, and fixed-point numbers. The specification of an equalizer design can be given in the floating-point domain, while the design implementation can be conducted in the fixed-point domain. This conversion generates and accumulates errors due to the different levels of accuracy provided by each number's domain. Therefore, a frequency domain multiplication based system must be tested thoroughly, and error analysis must be conducted to be sure about the correctness of its operation.

Verifying the correctness of an equalizer is very challenging because, firstly, its implementation can be based on an iterative algorithm, secondly, it can contain multiple FFT and IFFT blocks, and finally, it may contain multiple mathematical operators in different number domains. As a result, errors are naturally generated during data conversion between these domains, and can accumulate while performing various algorithmic iterations, FFT and IFFT operations. Therefore, a particular implementation of such a system must be verified in order to be sure that error accumulation is within acceptable limits.

In this paper, we will present a design and verification methodology for a frequency domain equalizer based on a combination of simulation and formal verification. This work is an extension of [2], within which we used theorem proving techniques in order to provide the error analysis for an implementation of the frequency domain equalizer based on the Fast Least Mean Square (Fast LMS) algorithm [3,4]. First, we will develop a model for the Fast LMS algorithm specification at the floating-point

* Corresponding author.
E-mail addresses: asouari@ece.concordia.ca (A. Souari), amjad.gawanmeh@kustar.ac.ae (A. Gawanmeh), tahar@ece.concordia.ca (S. Tahar), mohamed.ammari@eniso.rnu.tn (M. Lassaad Ammari).

0026-2692/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.mejo.2013.10.012

number level of abstraction in Simulink [5]. Then we will conduct simulation based verification in order to measure the floating-point signal-to-noise ratio (SNR) error for this model. Next, we will perform a multi-level formal error analysis to show that, when converting from one number domain to another, the algorithm produces the same values with an accepted error margin caused by the round-off error accumulation. We conduct formal error analysis at the floating-point, fixed-point, and real number domains using the high order logic (HOL) theorem prover [6]. Finally, we will provide an implementation for the Fast LMS algorithm in the frequency domain using the System Generator for DSP [7] at the fixed-point number level of abstraction. Overall, error estimation and analysis for both floating-point and fixed-point models are required to show that the error generated in the implementation of the algorithm conforms with the required accuracy of conversion in the equalizer design to operate properly.

The rest of the paper is organized as follows: Section 2 discusses related work. Section 3 describes the frequency domain equalizer implementation based on the Fast LMS algorithms. Section 4 presents our design and verification methodology. In Section 5 we present the Simulink model for the equalizer and simulation based error estimation. In Section 6, the error analysis of the frequency domain equalizer is formalized in HOL. Section 7 presents the System Generator based model of the equalizer and its error estimation. Finally, Section 8 concludes the paper and presents suggestions for future work directions.

2. Related work

The design and implementation of frequency domain equalizers is considered vital since equalization is a fundamental process in modern communication systems. Wang et al. [8] presented an iterative frequency–time domain equalizer for Advanced Television Systems Committee (ATSC). In this approach, the multipath distortion in the signal is first compensated with a frequency domain equalizer on a block-by-block basis. Then, a time domain interference cancelation algorithm is used to eliminate the inter-block and intra-block interference. These steps are repeated until the desired receiver performance, in terms of SNR and symbol error rate, is achieved. In another work, Luzio et al. [9] proposed a pragmatic iterative and non-iterative Frequency-Domain Equalization design for offset modulation methods in order to optimize the design for low-complexity transmitters and efficient power amplification. Dinis et al. [10] designed a frequency-domain equalizer for the receiver system that is optimized for Offset Quaternary Phase Shift Keying. Both Luzio et al. [9] and Dinis et al. [10] adopted bit error rate (BER) as reference for performance evaluation.

Recently, Sobaihi et al. [11] discussed the performance of an orthogonal frequency division multiple access transceiver with Frequency Domain Equalization based on time domain channel estimation. They also provided an implementation on FPGA platform. The authors measured the received signal constellation before and after equalization and used the bit error rate with SNR for their performance evaluation. Mori et al. [12] presented a method to estimate the average block error rate performance of star 32/64QAM schemes employing a frequency domain equalizer in that is designed for orthogonal frequency division multiple access systems. In addition, Komatsu et al. [13] presented an ASIC hardware implementation of a frequency domain equalizer and measured power consumption and BER for their design using simulation. Mehana and Nosratinia [14,15] provided an analysis of a single-carrier frequency domain equalizer for cyclic delay diversity and Alamouti signaling schemes in order to obtain a threshold rate for the full spatial–temporal diversity. Finally, Li et al. [16] presented a sliding window frequency domain equalizer for multi-mode systems which operates on the time-domain received signal.

The equalizer can be used for equalization in multi-mode systems when different waveforms are supported. This technique shows that equalizers are being developed and enhanced at a fast pace, hence, new testing and verification techniques must also be introduced in order to be sure about their correctness. In addition, the design and the applications of the frequency domain equalizer discussed above were concerned with performance parameters of the equalization process in the frequency domain such as SNR, symbol error rate, and block bit error rate. These parameters were estimated using the simulation environment of MATLAB. However, in all aforementioned methods, the error resulting from data conversion between different number domains is never measured or considered in the analysis. This error is due to handling the design and implementation of the equalizer at different levels of abstraction. Regardless of how small the error is, it can be amplified when introduced into an algorithm with repetitive and accumulative nature, such as frequency domain equalizer algorithms. In this work, we consider the frequency domain equalizer from this perspective and handle errors resulted from data conversions for equalizer models at different levels of abstraction.

On the other hand, the use of formal methods in the analysis of errors resulted from manipulating numbers at different levels of abstraction is not new. Harisson [17] used the HOL-Light theorem prover to approximate floating-point algorithms to their mathematical counterparts. He mainly proved that the floating-point exponential function has a correct overflow behavior and when this overflow is absent, the result is linked to a precise error value. In the analysis done by Harisson, the error represented as an independent random variable, is calculated depending on the arithmetic type and the rounding mode. Then, the mean square error is given after performing the error analysis.

Akbarpour [18] continued the work of [17] and proposed an error analysis framework based on theorem proving and dedicated specially to DSP algorithms. The methodology is based on the idea of representing the system in the three domains; the real, the floating-point and the fixed-point. Then, he calculated the error in the transitions from real to floating-point and real to fixed-point. Finally, the error in the transition from floating-point to fixed-point is driven by doing a subtraction between the two types of error calculated before. To show the feasibility of his methodology, Akbarpour applied his technique on digital filters [19] as well as on a 16 point radix 2 FFT [20]. Abu Nasser [21,22] adopted the methodology of Akbarpour [20] to study the error analysis of an FFT–IFFT which is a combination of a 64 point radix 4 FFT and IFFT blocks. Our work is also considered as an application of the formal verification framework developed in [18] since it is dealing with the error analysis of a frequency domain equalizer.

The application we verify in this work is considered more complex and error prone than the design in [22], where there is only a single combination of FFT and IFFT blocks, whereas our system is composed of three FFTs and two IFFTs. In addition, the frequency equalizer is based on arithmetic operations that use numbers of real, floating-point and fixed-point types. Hence, the formalization of error expressions and error analysis we intend to perform on the design is based on a theorem for complex numbers of the above different types. Finally, error analysis for the equalizer requires formalizing input vectors to be able to store various symbols in each iteration of the Fast LMS algorithm.

In summary, the contributions of this work compared to the state of the art are summarized as follows: first, the use of formal methods in the verification process of equalizers, in particular, theorem proving technique. Second, the combination of both simulation and theorem proving in the design and verification of frequency domain equalizers. Third, the ability to handle a frequency domain equalizer at multiple levels of abstraction, starting from the algorithmic level, then the time domain level, then the frequency domain level, and finally the FPGA

implementation level. Finally, the ability to make reuse existing theorems in the HOL theorem prover and build upon them to verify the correctness of the Fast LMS algorithm.

3. Frequency domain equalizer

Data alteration between the frequency domain and the time domain requires the use of some tools ensuring the preservation of data during this transition. The most useful tool enabling representation of a signal in the frequency domain is the discrete transform that helps to decrease the computational complexity related to signal processing just like convolution. The Discrete Fourier Transform (DFT) is used for the conversion of a discrete signal from the time domain to the frequency domain.

The DFT has two important properties: *Symmetry*, which means that the two elements $X(k)$ and $X(k+N)$ resulting by applying the DFT on the signal x are the same. This is a periodicity with period N. The second property is *Convolution*, where the time convolution theorem declares that a convolution in time domain is transformed into a simple multiplication in the frequency domain. Convolution is expressed as $x(n)=x_1(m)*x_2(n)=F^{-1}X_1(k)X_2(k)$, where x, x_1 and x_2 are the finite periodic signals, $*$ is the circular convolution, and F^{-1} is the Inverse Discrete Fourier Transform. This property allows using multiplication in the frequency domain instead of using convolution in time domain to reduce the complexity of the transformation. The Fast Fourier Transform (FFT) is an efficient transform algorithm between the two domains. The FFT algorithm computes one cycle of the DFT. Similarly, its inverse computes one cycle of the DFT inverse. The FFT and its inverse are the basic building blocks for frequency equalizers.

One implementation of the frequency domain equalizer [3,4] is based on an adaptive frequency domain algorithm called the Fast Least Mean Square (FLMS) algorithm. Fig. 1 illustrates the FLMS algorithm, where we use lower case variables to represent the signal in the time domain, and upper case variables to represent the signal in the frequency domain. The algorithm works as follows:

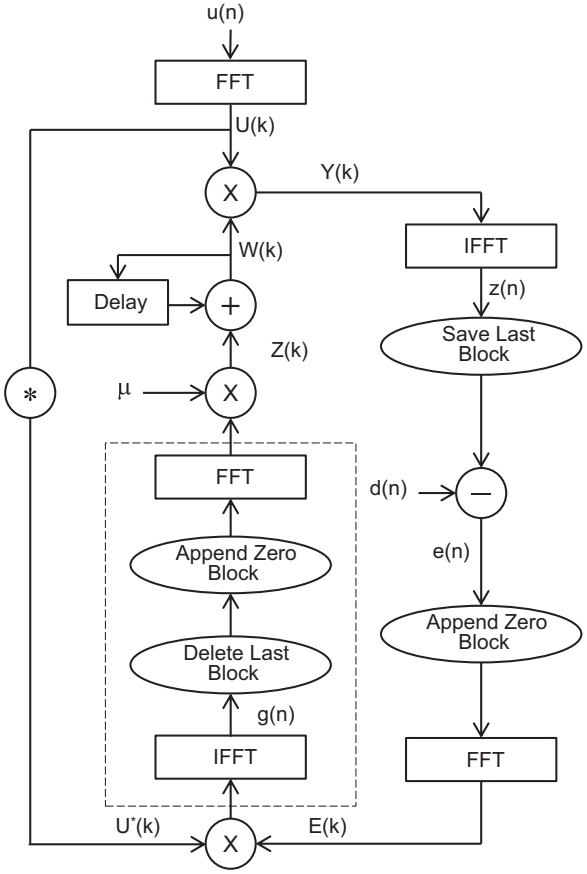

Fig. 1. Frequency domain equalizer design using the Fast LMS algorithm [6].

1. The FFT is applied on a $2N$ input block obtained from the input signal

   $U(k)=FFT\{u(n)\}$

2. The equalizer output in the frequency domain is obtained by multiplying $U(k)$ by the filter coefficients $W(k)$, where these coefficients are adjusted in every round

   $Y(k)=U(k)\cdot W(k)$

   Then, $Y(K)$ is converted into the time domain by applying IFFT

   $y(n)=IFFT\{Y(k)\}$

   Only the last N samples must be kept for circular convolution, hence the Save Last Block operation is applied on

   $y(n)=y(N+1\rightarrow 2N)$

3. Next, error in the signal is calculated by subtracting the current equalizer output $y(n)$ from the desired signal $d(n)$

   $e(n)=d(n)-y(n)$

4. Then, the error is transformed into the frequency domain $E(k)$ by applying FFT on $e(n)$ after appending N zeros to the signal

   $E(k)=FFT\{zeros,e(n)\}$

5. The conjugate of the $U(k)$ is calculated using the conjugate function (*), and it is multiplied by the error in the frequency domain $E(k)$

   $G(k)=E(k)\cdot U_*(k)$

6. Then, an IFFT is applied on $G(k)$ to transfer it to the time domain

   $g(n)=IFFT\{G(k)\}$

7. Since only the first N samples of this result are kept for the circular convolution, the last block is delated

   $g(n)=g(1\rightarrow N)$

8. Next, zeros are appended to $g(n)$ so that $2N$ point FFT can be applied on $g(n)$ and the result is multiplied by the step size parameter μ

   $Z(k)=\mu\cdot FFT\{g(n),zeros\}$

   This message, $Z(k)$, is used in order to update the equalizer coefficients for next round, $W(k+1)$, by adding it to the filter coefficients of this round, $W(k)$

   $W(k+1)=W(k)+Z(k)$

9. The next input block, $u(n+1)$ is then processed using the updated equalizer coefficients, $W(k+1)$. The iterative algorithm will cause the error to decrease because the coefficients are being updated progressively.

The above algorithm is based on the overlap-save convolution algorithm [3], where updating the equalizer coefficients in the

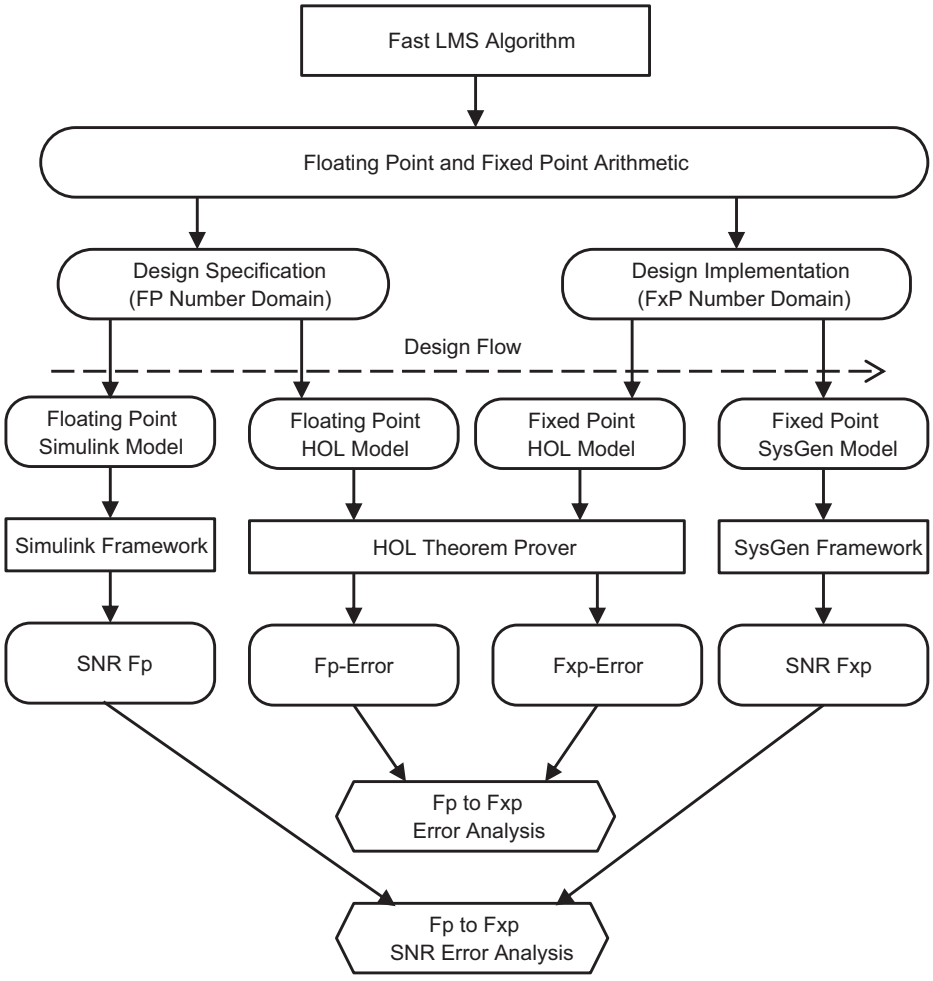

Fig. 2. Design and verification methodology for the Fast LMS algorithm.

frequency domain is done block by block instead of sample by sample.

4. Problem definition

To decrease the filtering complexity, an equalizer can be implemented in the frequency domain using the Fast Fourier Transform (FFT) and Inverse FFT (IFFT), where time convolution is replaced by frequency multiplication. This method offers low complexity growth in comparison with the time domain method. Data processing and filtering requires dealing with data at different domains: real number, floating-point number, and fixed-point number domains. The specification of an equalizer design can be given in the floating-point domain, while the design implementation can be conducted in the fixed-point domain. This conversion generates and accumulates errors due to the different level of accuracy provided by each number's domain.

In order to understand the problem, assume a real number, x, that is represented at the floating and fixed point levels by x_{fp} and x_{fxp}, respectively, in the equalization process described above. A limited number of bits must be used in each of these representations, hence, a rounding error is generated. The rounding error for floating-point representations can be defined as δ such that $x_{fp} = x(1+\delta)$. Similarly, the rounding error for fixed-point representations can be defined as ε such that $x_{fxp} = x(1+\varepsilon)$. When applying an equalization algorithm such as the Fast LMS, numbers are handled at the frequency domain level using fixed-point, rather than floating-point in the time domain, or real number domain in the actual design specification. Therefore, an error is generated due to the difference in the precision between these number domains. In addition, this error can accumulate when arithmetic operations such as addition and multiplication are applied in a repetitive manner [23].

In summary, a frequency domain multiplication based system must be tested thoroughly, and error analysis must be conducted to be sure about the correctness of its operation. In this work, we will use both simulation and formal analysis in order to measure this error and verify that the rounding error is within the design requirements during the implementation of the Fast LMS algorithm.

5. Design and verification methodology

In order to achieve a certain level of assurance we use a theorem proving based verification in order to provide formal error analysis for the equalizer in the frequency domain. The equalizer implementation is based on an iterative algorithm that contains multiple FFT and IFFT blocks. In addition, multiple mathematical operations are needed in different number domains: real, floating-point and fixed-point. As a result, errors are naturally generated during data conversion between these domains, and can accumulate while performing various algorithmic iterations, which are FFT and IFFT operations. However, the Fast LMS algorithm requires accuracy of conversion between different number domains in order to operate properly. Therefore, the formal error

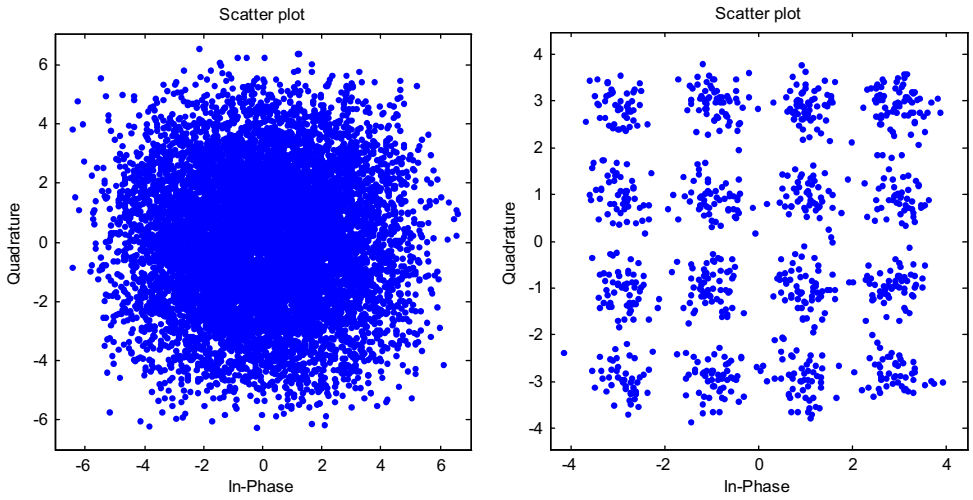

Fig. 3. Signal constellation before and after the Fast LMS-based equalization.

analysis of the algorithm intends to show that, when converting from one number domain to another, the algorithm produces the same values with an accepted error margin caused by the round-off error accumulation.

Then, we perform the formal error analysis to verify an implementation of the equalizer based on the Fast LMS algorithm. HOL theorem proving is used in order to provide formal error analysis for the design at different number domains. The expressiveness of HOL allows us to model the equalizer in all three number domains: floating-point, fixed-point, and real number domains. Errors are formally estimated by approximating the floating-point and the fixed-point designs to the real domain values.

Fig. 2 illustrates the design for verification methodology for the frequency domain equalizer based on the Fast LMS algorithm. In the first step, the formal specifications for the FLMS algorithm are obtained based on floating-point arithmetics. Then, the design is modeled in Simulink using a floating-point arithmetic toolbox. Simulink is used in order to estimate the SNR for the design in the frequency domain based on the floating-point arithmetic model. Hence, the SNR error estimation is obtained for the floating-point level design.

Next, we conduct formal error analysis at the floating-point, fixed-point, and real number domains. First, we obtained a floating-point HOL model for the design specification of the frequency domain equalizer. Then, we obtained a fixed-point HOL model based on the design implementation. Thereafter, we use HOL in order to calculate both the floating-point and fixed-point errors compared to the real number domain values. These two errors were used to obtain floating-point to fixed-point error analysis. On the other hand, we obtained a floating-point to fixed-point error analysis based on simulated SNR floating-point and fixed-point errors obtained by simulating Simulink and System Generator models, respectively.

Then, we build a System Generator design implementation model for the equalizer in the frequency domain for the Fast LMS algorithm. This design implementation is obtained using fixed-point arithmetics. Similar to the floating-point level design, the System Generator is used to provide a DSP implementation in order to estimate the SNR for the design in the frequency domains, where the error was estimated based on the fixed-point arithmetic model. Hence, the SNR error estimation is obtained for the fixed-point level design. Finally, we provide an FPGA based implementation for the Fast LMS algorithm.

Finally, we performed SNR error analysis between the Simulink model at the floating-point number abstraction level and the System Generator models at the fixed-point number abstraction level. The error analysis should be confirmed by the error analysis conducted using the formal analysis. In summary, the steps that we followed in our design and verification methodology were as follows:

- Model the frequency domain equalizer in Simulink in the floating-point domain (FLMS specification).
- Obtain SNR error estimation for the design specification at the floating-point domain abstraction level in Simulink.
- Model the frequency domain equalizer in HOL in the floating-point domain (FLMS HOL specification).
- Model the frequency domain equalizer in HOL in the fixed-point domain (FLMS HOL implementation).
- Perform formal error analysis in HOL.
- Implement the frequency domain equalizer in System Generator for DSP in the fixed-point domain (FLMS implementation).
- Obtain SNR error estimation for the design implementation at the fixed-point domain abstraction level in System Generator.
- Simulation based SNR error analysis between fixed-point and floating-point.
- Validate the SNR error analysis with formal error analysis.

This above approach shows the efficiency of combining simulation and formal verification based methods in the design and verification of complex systems such as the frequency domain equalizer.

6. Error estimation in Simulink

Error estimation is conducted for the Fast LMS algorithm using simulation in the Simulink environment. The simulation is based on error estimation for the 4-tap frequency domain equalizer, which converges after almost 200 symbols to reach the value of -40 dB. On the other hand, an FPGA based implementation is simulated for a 2-tap equalizer on the one million gate Spartan 3 FPGA board. The results obtained from the Simulink model were better than those obtained from the System Generator model because Simulink uses a floating-point description while System Generator uses a fixed-point one. The Simulink model for the Fast LMS algorithm block diagram can be found in [24].

Testing based verification for the Fast LMS algorithm is obtained by estimating the error generated in every step. The accuracy of the verification process was affected thoughtfully by

the method used in the framework to model numbers, be it floating-point or fixed-point. In addition, the verification process was based on applying a specific number of iterations, therefore in order to get more assurance about generated error, more simulation is required. For a certain level of assurance, specifically, in safety critical applications, simulation time becomes tremendous, and hence, we use theorem proving based verification in order to provide formal error analysis of the Fast LMS algorithm.

The implementation of the frequency domain equalizer is based on an adaptive frequency domain algorithm (Fast LMS). In this section we use Simulink in order to test the performance of the equalizer. The following simulation parameters for channel characteristics are used throughout all our simulations: *SNR*=40 dB, *Transmitted symbols*= 10 000, and $\mu = 0.002$. Fig. 3(a) and (b) presents the constellation of the input signal to the equalizer and the constellation of its output signal. Part (a) shows that the input signal to the equalizer is very noisy. After equalization, a constellation of 16-QAM was obtained, which means that the frequency domain equalizer eliminates the inter-symbol interference.

The error estimation curve for the Fast LMS equalizer shows that it converges within only 200 symbols to reach a steady error rate equal to −40 dB for an SNR of 40 dB. This implies that the frequency domain equalizer is more efficient than the time domain equalizer. The Fast LMS algorithm as the adaptive algorithm for the equalizer gives an error rate equal to 1.83%. Fig. 4 illustrates the equalizer error estimation for the Fast LMS in the frequency domain.

The architecture of the Fast LMS-based equalizer is modeled in the Simulink environment as a transmission chain. Fig. 5 shows the building blocks of a Fast LMS frequency domain equalizer where the 4-tap frequency domain is used. In this equalizer architecture, three FFT blocks and two IFFT blocks are used to allow the alternation between the time domain and the frequency domain.

Fig. 6 shows the constellation of the equalizer output signal. The constellation shows no noise which implies that the equalizer is working perfectly. The 4-tap frequency domain equalizer converges after around 200 symbols to a steady value of −40 dB. The error estimation curve for the 4-tap equalizer is shown in Fig. 7.

7. Formal error analysis

Higher-order-logic (HOL) theorem proving [25] is a formal method that is used to conduct precise analysis of various systems.

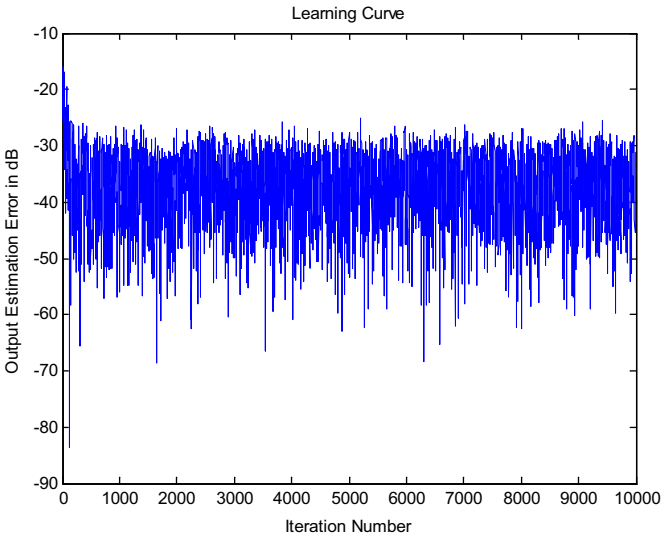

Fig. 4. Fast LMS equalizer error estimation in the frequency domain.

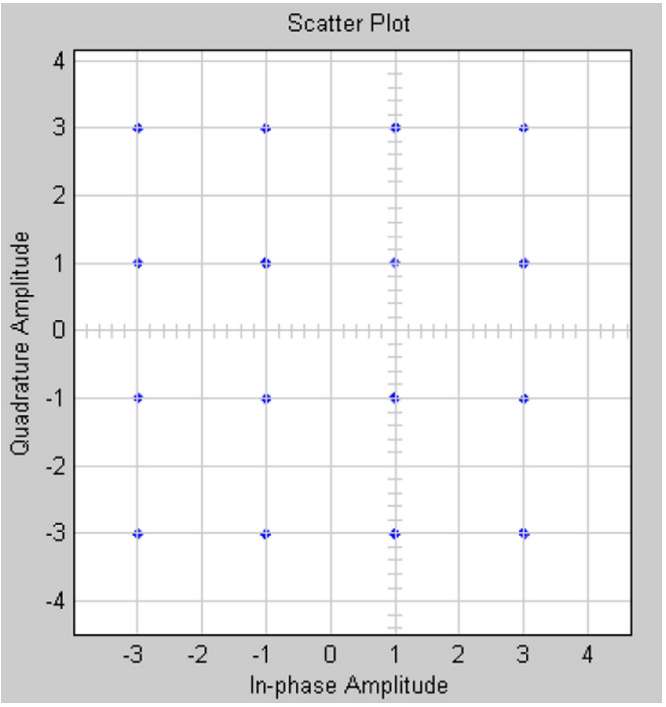

Fig. 6. Signal constellation after frequency domain equalization in Simulink.

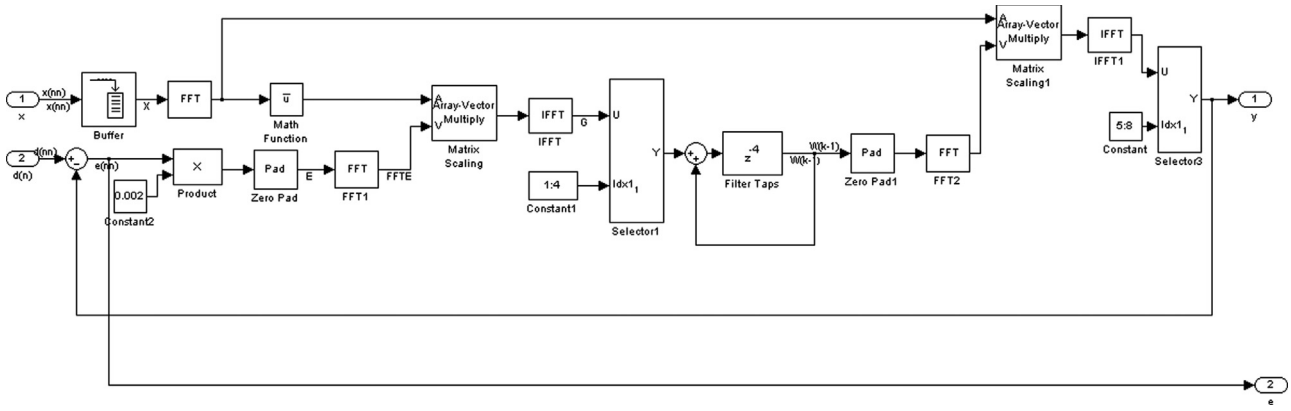

Fig. 5. Simulink model for the Fast LMS equalizer in the frequency domain.

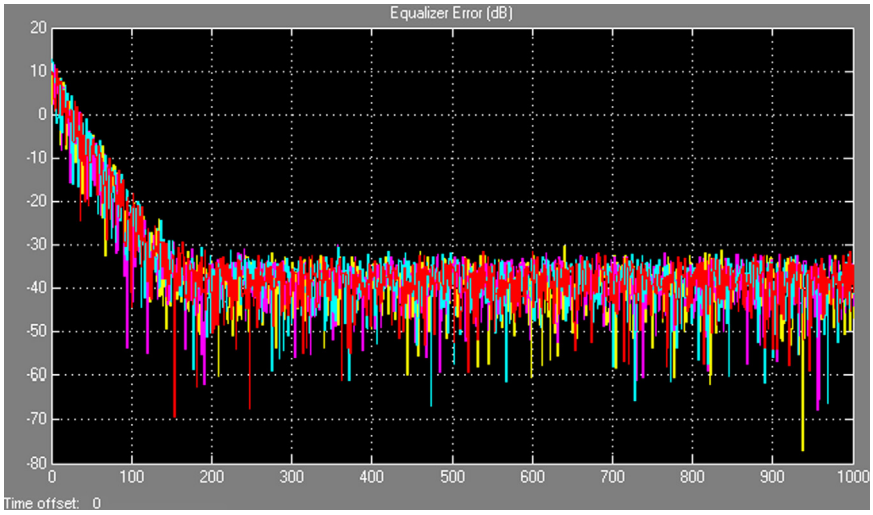

Fig. 7. Error estimation in the frequency domain.

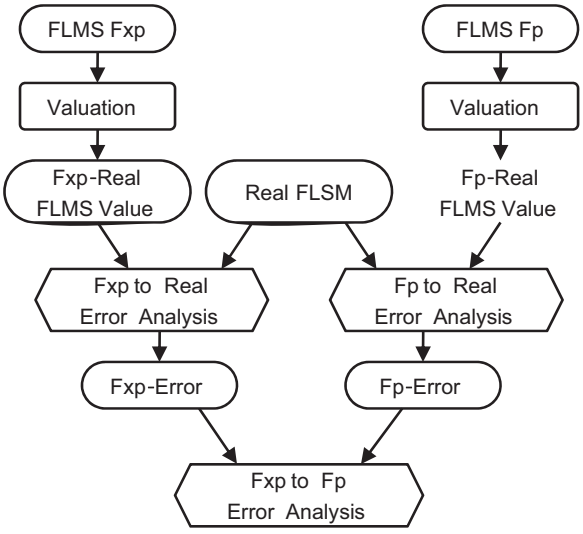

Fig. 8. Error analysis verification methodology in HOL.

It is based on a system of deduction with precise semantics and is expressive enough to be used for the precise specification of systems such as the frequency domain equalizer. The HOL theorem prover framework [6] is an interactive tool dedicated to conduct proofs in higher-order logic. There are only four types of terms in HOL: variables, constants, function application and lambda terms [6]. The main core of HOL consists of five axioms and eight inference rules. All the theories existing in HOL are built on the top of them. All the theories should be proved before they are added to the HOL inference system.

7.1. Methodology

The verification methodology, as depicted in Fig. 8, is based on a formal model for numbers in three different domains: fixed-point (FXP), floating-point (FP), and real domains and a valuation procedure for numbers conversion. Based on this conversion, error analysis is performed between the actual real values obtained and the converted ones from both floating-point and fixed-point domains. Finally, further analysis is performed to show the error analysis between the fixed-point and the floating-point.

In this verification methodology, the Fast LMS algorithm should be formalized in the three number domains in the same way it is defined above. The valuation functions should be used in order to return real approximations of the floating-point and fixed-point algorithms. In our error analysis we obtain the rounding error between the FXP to Real and FP to Real error analysis. These are defined as the difference between the approximations of the fixed-point and the floating-point algorithms and the real specification.

To perform the error analysis of the Fast LMS algorithm, we used the existing theories in HOL and built on top of them the necessary theories to reason about error generation and accumulation in the equalizer. For the floating-point and the fixed-point modeling of the design, we used, respectively, the formalization of the IEEE 754 standard based floating-point arithmetic [19] and the fixed-point arithmetic HOL theorems developed by Akbarpour et al. [23].

7.2. Modeling the frequency domain equalizer in HOL

To formalize the error due to the floating-point rounding in HOL, we use two fundamental theorems, the first one deals with floating-point rounding error, and states that *if x is a real number within the floating-point range, then* $x_R = x(1+\delta), |\delta| \le 2^{-p}$, while the second one deals with fixed-point rounding error, and states that *if x is a real number within the fixed-point range, then* $x_R = (x+\varepsilon), |\varepsilon| \le 2^{-fracbits(X)}$, where p is the precision of the floating-point format, x is the real number, and x_R is the floating-point value. The rounding error in the floating-point domain is multiplicative, while it is additive in the fixed-point domain [23]. Evaluating arithmetic operations, denoted as $\star$, in the floating-point domain is defined as $fl(x \star y) = (x \star y)(1+\delta)$, where $|\delta| \le 2^{-p}$. Similarly, in the fixed-point domain it is defined as $fl(x \star y) = (x \star y) + \varepsilon$, where $|\varepsilon| \le 2^{-fracbits(X)}$.

In order to model the error analysis of the frequency domain equalizer, existing HOL theories developed by Abdullah [18] were used. In addition, theories for complex numbers and many other required functions like the complex sum in all three number domains had to be defined as described in the sequel.

7.2.1. Real-numbers domain modeling

Modeling the frequency domain equalizer using HOL in real domain requires formalizing complex numbers. We defined complex

as a new datatype based on a pair of real numbers as shown below:

```
Complex = ⊢def complex of (real # real)
```

The real and imaginary parts of a complex number are also formalized in HOL:

```
Re_def = ⊢def Re (complex (a,b)) = a

Im_def = ⊢def Im (complex (a,b)) = b
```

We also define properties on complex numbers that are needed to model the equalizer in HOL. For instance, the conjugation is formalized using this definition:

```
CNJ = ⊢def ∀ z.CNJz = complex(Rez, ¬Imz)
```

where ¬ in the above expression represents the conjugate of the imaginary part of the complex number, i.e., its negative value.

Arithmetic operations on complex numbers such as addition, subtraction and multiplication were also defined in HOL and properties about these operations were proved, such as complex multiplication and complex addition commutativity. The principal *n*-roots of unity is a fundamental building block for these arithmetic operations, hence it was defined in HOL using Euler's identity as follows:

```
principal_root
= ⊢def ∀nk.principal_rootnk
= complex (cos n*k*Π/¬2, sin n*k*Π/¬2)
```

One fundamental function that is used in both FFT and IFFT is the complex summation which is defined in HOL recursively as follows:

```
rec_sum_def
= ⊢def rec_sum(n, 0)f
  = complex(0, 0) ∧ rec_sum (n, SUCm)f
    = rec_sum (n,m)f + f (n + m)
```

where *n* and 0 are the upper and lower indices, respectively, and *f* is a function. Based on these definitions, we define both FFT and IFFT in real time domain using HOL as follows:

```
real_FFT_def
= ⊢def ∀xk.real_FFTxk
  = rec_sum (0,3) (\n.principal_rootnk*ELnx)

real_IFFT_def
= ⊢def ∀Lk.real_FFTxk
  = complex_4 *rec_sum (0,3) (\n. principal_root n
    k*EL n L)
```

where complex constant was defined as

```
complex_4_def = ⊢def complex_4 = complex(1/4, 0)
```

We adopted the function *EL* from the *pairTheory* in order to extract the *n*th element of a complex list *L*. The term \n in the *real_FFT_def* and *real_IFFT_def* theorems is a *lambda-abstraction* which shows that *sum* is a function of *n*.

7.2.2. Floating-point domain modeling

Complex numbers modeling in this domain is similar to the real domain except that here the complex numbers are represented as pair of floats

```
complex = ⊢def complexof (float#float)
```

Floating-point complex summation is defined using as follows:

```
float_complex_sum_def
= ⊢def float_complex_sum (n, 0) f
  = float_complex (float (0,0,0), float (0,0,0)) ∧
  float_complex_sum (n, SUC m)   f = float_complex_sum
   (n,m) f + f (n + m)
```

This definition explains how to add two complex numbers in the floating point domain. This addition creates a rounding error, which shall be modeled in HOL as well. Therefore, we built on top of the IEEE floating-point rounding formalization by Harrison [17] in order to define a floating-point complex rounding function using the predefined *round* function to calculate the rounding value of a floating-point complex number as follows:

```
float_complex_round_def
= ⊢def ∀z.float_complex_roundz
  = float_complex (float (round float_format
    To_nearest (Re z)),
   float (round float_format To_nearest (Im z)))
```

The inverse function of rounding is called valuation, it gives the equivalent real of any floating-point number. It is defined in HOL as *Val*. For the valuation of the complex numbers of type *float* we define the function *float_complex_val*

```
float_complex_val_def
= ⊢def ∀z.float_complex_valz
  = complex (Val (float_Re z), Val (float_Im z))
```

In order to model both FFT and IFFT blocks in the floating-point domain, we used the functions *float_principal_root* and *float_principal_root_1* to define *float_FFT* and *float_IFFT*, respectively. These two former functions provide the rounding results of the two functions *principal_root* and *principal_root_1* defined in the previous section above. The definition of FFT and IFFT in HOL is given as

```
float_FFT_def
= ⊢def ∀xk.float_FFTxk
  = float_complex_sum (0,3) (\n. float_principal_root n
    k * EL n x)

float_IFFT_def
= ⊢def ∀Lk.float_IFFTxk
  = float_complex_4*float_complex_sum (0,3)
  (\n. float_principal_root_1 n k* EL n L)
```

These two definitions provide the implementation of the FFT and IFFT in HOL based on floating-point complex numbers summation.

7.2.3. Fixed-point domain modeling

The formalization of the Fast LMS algorithm in HOL in the fixed-point domain is different from the formalization in the other two domains. This is due to the use of the primitive parameters for arbitrary attributes related to fixed-point numbers. We used *fxp* to define complex numbers in the fixed-point domain. Retrieving the real and imaginary parts of a complex number of type *fxp* is achieved using *fxp_Re* and *fxp_Im*, respectively. Complex addition,

complex subtraction and complex multiplication are defined using functions *fxp_complex_add*, *fxp_complex_sub* and *fxp_complex_mul*, respectively. As we mentioned, the definition of the complex summation for the fixed-point domain is given as

```
fxp_complex_sum_def
⊢def fxp_complex_sum (n,0) X f
  =fxp_complex (fxp (WORD (REPLICATE (streamlength
   (X)) F),X),
  fxp (WORD (REPLICATE (streamlength (X)) F),X)) ∧
  fxp_complex_sum (n,SUC m) X f
  =fxp_complex_add X (fxp_complex_sum (n,m) X f) (f
   (n + m))
```

The function *Fxp_round* converts a real number into its fixed-point equivalent number. To perform rounding of a complex number of type *fxp*, we defined the function *fxp_complex_round_def* as follows:

```
fxp_complex_round_def
⊢def ∀ X z. fxp_complex_round X z
  =fxp_complex (Fxp_round X Re(z), Fxp_round X Im
   (z))
```

To obtain the real value of a fixed-point number, we use the function *value* that is defined in the *fxpTheory* in HOL. The function *fxp_complex_value* is defined to perform the valuation of fixed-point complex numbers as follows:

```
fxp_complex_value
⊢def ∀ z. fxp_complex_value z
  =complex (value fxp_Re (z), value fxp_Im (z))
```

Similar to the floating-point domain, we formalize the FFT and IFFT blocks in the fixed-point domain as follows:

```
fxp_FFT_def
⊢def ∀ X x k. fxp_FFT X x k
  =fxp_complex_sum (0,3) X (\n. (fxp_complex_mul X
  (fxp_principal_root X n k) (EL n x) ))
```

```
fxp_IFFT_def
  ⊢def ∀ X L k. fxp_IFFT X L k
  =fxp_complex_mul X (fxp_complex_4 X)
   (fxp_complex_sum (0,3) X
  (\n.(fxp_complex_mul X (fxp_principal_root_1 X n
   k) (EL n L) )))
```

where *fxp_complex_4 X* refers to the term $1/N$ in the IFFT equation. It is a constant complex number which value is equal to 1/4 since the number of taps, N, adopted for our design is equal to 4. These definitions provide the implementation of the FFT and IFFT in HOL based on fixed-point complex numbers summation function.

7.3. Error analysis of the frequency domain equalizer in HOL

In order to perform error analysis in HOL, a theorem for every building block of the design must be defined and proved. Then, one comprehensive theorem for the whole design is required to show the validity of rounding and error accumulation. In this section, we will discuss the major theorems we defined for error analysis of the design. The structure and relationship between these theorems is shown in Fig. 9.

We start by formalizing errors in the floating-point domain for the designs using the lemma *float_complex_val*, denoted as Val_{f_p}.

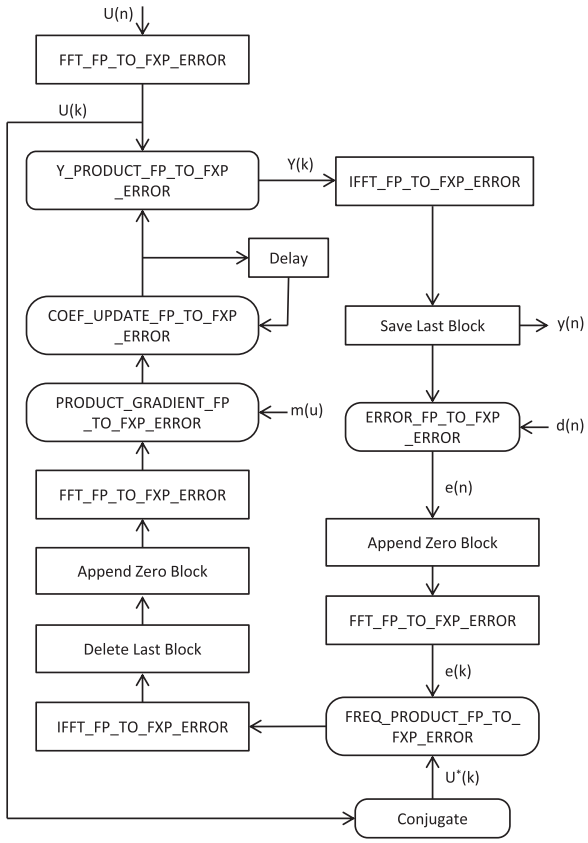

Fig. 9. Structure of HOL theorems for the equalizer.

The lemma is formalized in HOL as follows:

```
float_complex_val
⊢lemma ∀ z. float_complex_val z
=complex(Val(float_Re z),Val (float_Im z))
```

Similarly, we formalize errors in the fixed-point domain using the lemma *float_complex_val*, denoted as Val_{f_x}. This lemma is formalized in HOL as follows:

```
fxp_complex_value
⊢lemma ∀ z. fxp_complex_value z
=complex(value(fxp_Re z),value(fxp_Im z))
```

The effect of these functions on the arithmetic operations is inherited from the effect of the function *Val* and *value*. The function *Val* is used to define the rounding error due to the valuation of the floating-point in real number. The effect of the *Val* function on arithmetic operations is defined as

$$\forall ab \cdot \exists e \cdot Val(a+b) = (Vala + Valb) * (1+e)$$
$$\forall ab \cdot \exists e \cdot Val(a-b) = (Vala - Valb) * (1+e)$$
$$\forall ab \cdot \exists e \cdot Val(a * b) = (Vala * Valb) * (1+e)$$

where a and b are the two floating-point numbers and e a real number. e in the above lemmas defines the error caused by the valuation of the fixed-point in real number. The effect of the *value* function on arithmetic operations is given as

$$\forall abX \cdot \exists e \cdot value(FxpAddXab) = valuea + valueb + e$$
$$\forall abX \cdot \exists e \cdot value(FxpSubXab) = valuea - valueb + e$$
$$\forall abX \cdot \exists e \cdot value(FxpMulXab) = valuea * valueb + e$$

These lemmas are dedicated to complete the error analysis for rounding numbers between different domains, and will be used to formalize FFT and IFFT in the next section.

7.4. Fast LMS error analysis

The major theorems we defined for the equalizer design are shown in Fig. 9. In order to perform complete error analysis for the whole design, each block of the Fast LMS algorithm described above in Fig. 1 must be formalized in HOL. As shown in the figure below, the theorems for every block intend to show the validity of rounding and error accumulation.

The first theorem deals with error analysis between the real and the floating-point representations of the FFT block. It consists of a simple subtraction between the floating-point valuation and the real valuation of FFT. Using C for composing a complex number, P_{f_p} for the float principal root function, P for the real principal root, and $\sum_{n=0}^{k}$ for `rec_sum_def`, this theorem, denoted as EF_{RF_p}, is formalized as follows:

$$\begin{aligned}&\vdash_{thm} \forall x,y,k\cdot \exists e1,e2,e3\cdot EF_{RF_p}(x,y,k)\\&\quad = Val_{f_p}(P_{f_p}(0,k)*EL(0,y)*C(1+e3,0))\\&\qquad + Val_{f_p}(P_{f_p}(1,k)*EL(1,y)*C(1+e2,0))\\&\qquad + Val_{f_p}(P_{f_p}(2,k)*EL(2,y)*C(1+e1,0)) - \sum_{n=0}^{3}(P(n,k)*EL(n,x))\end{aligned}$$

This HOL theorem defines the rounding error for the FFT block between real and floating point number domains, and is based on the valuation function, complex numbers, and summation definitions. This rounding error represents the error in the equalization process at the floating-point number domain contributed by the FFT block.

The real to fixed-point error analysis of the FFT is established by proving that the produced error is equivalent to subtraction between the valuated fixed-point FFT expression and the real one. Using *MUL* for complex multiplication, and P_{f_x} for the fixed-point principal root function, this error, denoted as EF_{RF_x}, is formalized as follows:

$$\begin{aligned}&\vdash_{thm} \forall x,y,X,k\cdot \exists e1,e2,e3\cdot EF_{RF_x}(x,y,X,k)\\&\quad = Val_{f_x}(MUL(X,P_{f_x}(X,0,k),EL(0,y))) + C(e3,e3)\\&\qquad + Val_{f_x}(MUL(X,P_{f_x}(X,1,k),EL(1,y))) + C(e2,e2)\\&\qquad + Val_{f_x}(MUL(X,P_{f_x}(X,2,k),EL(2,y))) + C(e1,e1)\\&\qquad - \sum_{n=0}^{3}(P(n,k)*EL(n,x))\end{aligned}$$

Once the real to fixed-point error analysis is achieved, the fixed-point to floating-point error analysis of the FFT block is obtained by deducting the results of the real to floating-point and the real to fixed-point error expressions as given in Fig. 8 above. The following theorem, denoted as $EF_{F_pF_x}$, is defined in order to formalize the fixed-point to floating-point error analysis of the FFT. This error is formalized as follows:

$$\begin{aligned}&\vdash_{thm} \forall x,x_p,x_f,X,k\cdot \exists e1,e2,e3,e4,e5,e6\cdot EF_{F_pF_x}(x,x_p,x_f,X,k)\\&\quad = Val_{f_x}(MUL(X,P_{f_x}(X,0,k),EL(0,x_f))) + C(e3,e3)\\&\qquad + Val_{f_x}(MUL(X,P_{f_x}(X,1,k),EL(1,x_f))) + C(e2,e2)\\&\qquad + Val_{f_x}(MUL(X,P_{f_x}(X,2,k),EL(2,x_f))) + C(e1,e1)\\&\qquad - (Val_{f_p}(P_{f_p}(0,k)*EL(0,x_p))*C(1+e6,0)\\&\qquad + Val_{f_p}(P_{f_p}(1,k)*EL(1,x_p))*C(1+e5,0)\\&\qquad + Val_{f_p}(P_{f_p}(2,k)*EL(1,x_p))*C(1+e4,0))\end{aligned}$$

Similar theorems were defined and proven for the IFFT block. For illustration purposes, the theorem for real to floating-point error, denoted as EI_{RF_p}, is formalized as follows:

$$\begin{aligned}&\vdash_{thm} \forall L,L_x,k\cdot \exists e,e1,e2,e3\cdot EI_{RF_p}(L,L_x,k)\\&\quad = Val_{f_p}(C_{4_f}*C(1+e,0)*((Val_{f_p}(Pi_{f_x}(0,k),EL(0,L_x))\\&\quad *C(1+e3,0) + Val_{f_p}(Pi_{f_x}(1,k),EL(1,L_x)))*C(1+e2,0)\\&\quad + Val_{f_p}(Pi_{f_x}(2,k),EL(2,L_x)))*C(1+e1,0))\\&\quad - C_4*\sum_{n=0}^{3}(P_{f_x}(n,k)*EL(n,x))\end{aligned}$$

Having an HOL theorem defined and proved for every building block of the design, one comprehensive theorem for the whole algorithms that defines the blocks of the design together. The HOL theorem [24] is used to obtain the rounding error from each block and accumulate it together with the error produced by the successor block in the design as depicted in Fig. 9. Eventually, the rounding error is obtained for the whole design and validated for the algorithm. This theorem is proved in HOL, which verifies that the rounding and accumulated error produced by steps of the algorithm are within the accepted range given in the specification of the design. Theorems for error analysis of the blocks of the frequency domain equalizer that were formalized and proved in HOL are presented in [24].

7.5. Discussion

Many existing theories in HOL, e.g., *arithmeticTheory*, *realTheory*, *listTheory*, *pairTheory*, *realLib*, *numLib*, *floatTheory*, *fxpTheory*, *ieeeTheory*, ..., were used to derive the rounding error analysis of the frequency domain equalizer. The definitions of the Fast LMS algorithm were formalized and proved based on these theories. All the definitions were formalized first in the real domain, and then all the arithmetic operators were overloaded to build the design in the floating-point and fixed-point domains using *floatTheory* and *fxpTheory*, respectively.

The equalizer application shows that formal error analysis is applicable on larger scale systems such as the one we have analyzed, which is traditionally analyzed with paper and pencil or simulation based techniques based on estimating the error. Formal analysis proves that the implementation meets its specification with 100% coverage, something that is not feasible in simulation. In addition compared to the classical analytical technique, this method is computerized and has been conducted using an interactive tool, and, the theorems can be efficiently reused to verify other designs that make use of the same algorithm.

Some specific error analysis theorems that were used in this work were defined and proven by Akbarpour [18] and Abdullah [21]. However, we had to build our own theorems on top of these in order to formalize and verify every block of the Fast LMS algorithm, and consequently define one single theorem for the whole design that we included in [24]. This shows that relevant theorems that are proven in HOL can be reused efficiently in order to verify complex systems of similar properties. In fact, scalability of HOL theorems is one of its best features, since all proven theorems can be reused efficiently to verify other designs, which reduces time and effort, in particular while using the interactive HOL framework environment.

8. Error estimation in system generator

In order to provide an implementation for the equalizer, we first build a System Generator design implementation model in the frequency domain for the Fast LMS algorithm. This design implementation is obtained using fixed-point arithmetics. The system generator is used to provide a DSP implementation in order to estimate the SNR for the design in the frequency domains, where the error was estimated based on the fixed-point arithmetic

model. Hence, the SNR error estimation is obtained for the fixed-point level design.

The design contains elementary blocks from the Xilinx block set which are essentially multipliers, adders, FFT, IFFT and sub-systems performing complex multiplication. The System Generator model for the Fast LMS algorithm block diagram can be found in [24]. The signal constellations as well as the error estimation curve given by Figs. 10 and 11, respectively, show that the equalizer eliminates the inter-symbol interference from the noisy input signal. The error estimation curve shows that the equalizer converges after around 100 symbols to a steady value of −15 dB, while Simulink simulations showed that the equalizer converges into its steady state value of −40 dB.

The SNR error analysis is calculated by comparing the SNR error given by the fixed-point frequency domain equalizer that was implemented using System Generator for DSP blocks and the estimated SNR error given by the floating-point equalizer that was implemented using Simulink blocks. In our case, the SNR error given by the fixed-point equalizer is equal to −15 dB while the SNR error generated by the floating-point equalizer is valued to −40 dB. The main reason for this difference is the change of the number domain from the floating-point at the specification level of abstraction into the fixed-point at the implementation level of abstraction. Hence, this error produced by the equalizer is due to the conversion between these two number domains. This error confirms the definition of the rounding error that was defined above for both floating-point and fixed-point domains compared to the real number domain:

$$x_{Fp} = x(1+\delta), \quad |\delta| \le 2^{-p}$$

$$x_{Fxp} = (x+\varepsilon), \quad |\varepsilon| \le 2^{-fracbits(X)},$$

where p is the precision of the floating-point format, x is the real number, x_{Fp} is the floating-point value, and x_{Fxp} is the fixed-point value.

In these representations, we have three errors that were recognized by our results: the error in equalization due to representing the signal in the floating-point domain, i.e., δ, the error in equalization due to representing the signal in the fixed-point domain, i.e., ε, and the error in representing the signal with two different precisions, i.e., $\delta-\varepsilon$. The accumulation of the latter error leads to a difference between the two error rate values equals to −25 dB, where the error rate value in the floating-point domain is equal to −40 dB, and the error rate value in the fixed-point domain is equal to −15 dB. On the other hand, the HOL formal error analysis proved that the signal value in the floating-point domain is equal to its fixed-point counterpart plus an error within a certain margin, hence, verifying that $\delta-\varepsilon$ was within the acceptable error range according to the design specifications.

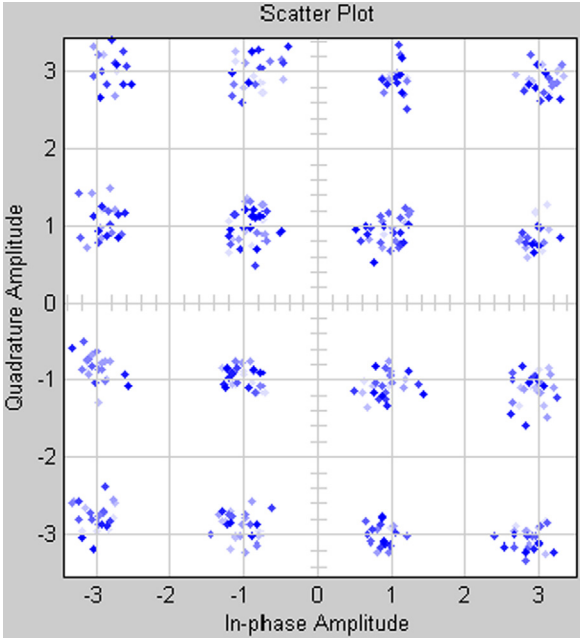

Fig. 10. Signal constellation after frequency domain equalization in System Generator.

9. Conclusion and future work

In this work, we proposed a design and verification methodology for a frequency domain equalizer implemented using the Fast LMS algorithm in the frequency domain where a number of mathematical operations are performed on numbers in three different domains: floating-point, fixed-point and real numbers. As a result, errors are naturally generated during data conversions between these domains, and can accumulate while performing various algorithmic iterations, such as FFT and IFFT operations. The proposed methodology for the design and verification of the frequency domain equalizer spreads over various design flow

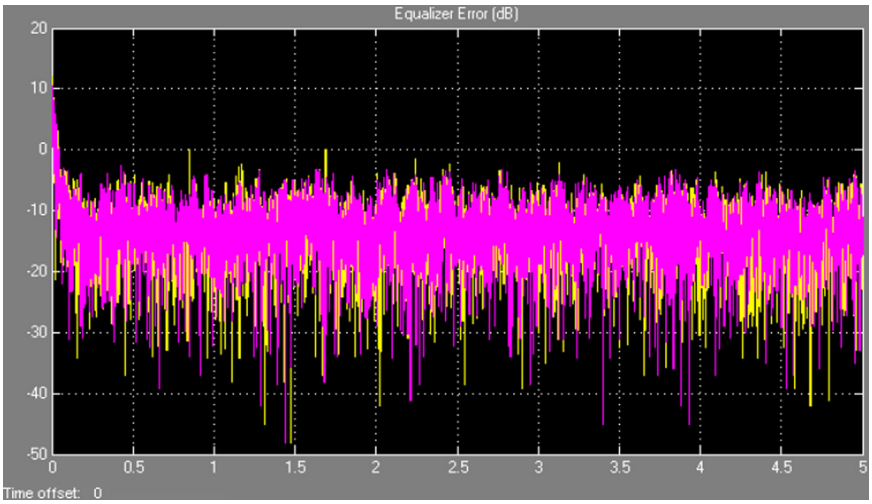

Fig. 11. Error estimation curve.

levels considering different number domains: real, floating-point, and fixed-point.

In this paper, we first implemented the frequency domain equalizer in Simulink in the floating-point domain based on the Fast LMS algorithm specifications. This model is used to estimate the SNR in the design. The equalization error generated in the simulation is estimated to be around −40 dB. This error is due to the rounding of the signal at the floating-point number domain. Next, we used formal techniques in order to perform formal error analysis between a floating-point HOL model and a fixed-point HOL model of the frequency domain equalizer. To achieve this, theorem proving was used to provide formal error analysis for the frequency domain equalizer. This required data is to be converted between the fixed-point and the floating-point domains, which in turn produces errors that can accumulate during the several iterations of the algorithm. To perform this analysis, basic formal definitions and theorems were required in order to handle iterative nature of the algorithm and the multiple FFT and IFFT blocks in the equalizer. The formal error analysis was used to show that errors in the equalizer algorithm that occur while converting from one number domain to the another are within the accepted range based on the design specification of the equalizer. However, this was not a straightforward task and required building on existing HOL theorems for error analysis as well as the derivation of new expressions for the accumulation of round-off error in the algorithm.

Finally, in order to cover all the design flow levels, we implemented the frequency domain equalizer in the System Generator for DSP based on the Fast LMS design implementation at the fixed-point domain level of abstraction. This implementation is used to conduct SNR error estimation by simulating the design model in the System Generator. Simulation curves showed that equalization generated an error of −15 dB. This error is due to the rounding of the signal at the fixed-point number domain compared to the real value. We then conducted simulation based SNR error analysis between fixed-point and floating-point for above specifications and implementations. In fact, the error generated at the fixed-point number domain is larger than the error generated at the floating-point number domain. This is due to the lower precision of the fixed-point domain. This difference was formally proved to be within the acceptable precision as stated in the design specification. This application showed that performance analysis for the equalizer can be conducted using both formal error analysis and simulation based error analysis in order to cover various levels of the design flow.

As future work, we plan to extend the current work and perform error analysis using the GAPPA framework developed by Melquiond [26]. Another interesting approach is to use first order theorem proving to verify properties about the functional behavior of the equalizer.

References

[1] J. Boca, J. Bowen, Siddiqi, Formal Methods: State of the Art and New Directions, Springer-Verlag, London Limited, 2010.

[2] A. Souari, S. Tahar, A. Gawanmeh, Formal error analysis and verification of a frequency domain equalizer, in: IEEE International New Circuits and Systems Conference, IEEE Computer Society Press, 2012, pp. 189–192.

[3] M.O. Fril, Frequency Domain Adaptive Filtering, Master's Thesis, National University of Ireland, UK, 2005.

[4] P.A. Dmochowski, Frequency Domain Equalization for High Data Rate Multi-path Channels, Master's Thesis, Queen's University, Kingston, Ontario, Canada, 2001.

[5] Matlab, 2012.

[6] HOL Sourceforge Project. The HOL System Reference, ⟨http://hol.sourceforge.net⟩, 2011.

[7] Xilinx, 2008.

[8] X. Wang, Y. Wu, C. Nadeau, G. Gagnon, Design and implementation of frequency domain equalizer for ATSC system, in: IEEE International Symposium on Broadband Multimedia Systems and Broadcasting, IEEE Computer Society Press, 2008, pp. 1–6.

[9] M. Luzio, R. Dinis, P. Montezuma, A pragmatic design of frequency-domain equalizers for offset modulations, in: IEEE Vehicular Technology Conference, IEEE Computer Society Press, 2012, pp. 1–5.

[10] R. Dinis, M. Luzio, P. Montezuma, On the design of frequency-domain equalizers for OQPSK modulations, in: IEEE Sarnoff Symposium, IEEE Computer Society Press, 2010, pp. 1–5.

[11] K. Sobaihi, A. Hammoudeh, D. Scammell, FPGA implementation of frequency domain equalizer with time domain channel estimation for millimetre-wave OFDM system, in: Wireless Telecommunications Symposium, IEEE Computer Society Press, 2012, pp. 1–6.

[12] C. Mori, T. Kawamura, N. Miki, M. Sawahashi, Link-level performance of star 32/64QAM schemes using frequency domain equalizer for DFT-precoded OFDMA, in: International Symposium on Wireless Personal Multimedia Communications, IEEE Computer Society Press, 2012, pp. 266–270.

[13] K. Komatsu, S. Kameda, M. Iwata, S. Tanifuji, N. Suematsu, T. Takagi, K. Tsubouchi, ASIC implementation of frequency domain equalizer for single carrier transmission, in: General Assembly and Scientific Symposium, IEEE Computer Society Press, 2011, pp. 1–4.

[14] A. Mehana, A. Nosratinia, Performance of MIMO single-carrier frequency domain zero-forcing equalizer, in: IEEE International Symposium on Information Theory, IEEE Computer Society Press, 2012, pp. 3023–3027.

[15] A. Mehana, A. Nosratinia, Single-carrier frequency-domain equalizer with multi-antenna transmit diversity, IEEE Trans. Wireless Commun. 12 (1) (2013) 388–397.

[16] J. Li, E. Bala, R. Yang, Sliding window-frequency domain equalization for multi-mode communication systems, in: IEEE Long Island Systems, Applications and Technology, IEEE Computer Society Press, 2013, pp. 1–6.

[17] J. Harrison, Floating Point Verification in Hol Light: The Exponential Function, Technical Report 428, Computer Laboratory, University of Cambridge, Cambridge, UK, 1997.

[18] B. Akbarpour, Formal Verification Methodology of DSP Designs, Ph.D. Thesis, ECE Department, Concordia University, Montreal, Quebec, Canada, 2005.

[19] B. Akbarpour, S. Tahar, Error analysis of digital filters using theorem proving, in: Theorem Proving in Higher Order Logics, Lecture Notes in Computer Science, vol. 3223, Springer-Verlag, Berlin, Heidelberg, 2004, pp. 1–16.

[20] B. Akbarpour, S. Tahar, A methodology for the formal verification of FFT algorithms in HOL, Formal Methods Comput. Aided Des. 3312 (2004) 37–51.

[21] N. Abdullah, Formal Analysis and Verification of an OFDM Modem Design, Master's Thesis, ECE Department, Concordia University, Montreal, Quebec, Canada, 2006.

[22] N. Abdullah, B. Akbarpour, S. Tahar, Error analysis and verification of an IEEE 802.11 OFDM Modem using theorem proving, Electron. Notes Theor. Comput. Sci. 242 (2) (2009) 3–30.

[23] B. Akbarpour, A. Dekdouk, S. Tahar, Formalization of fixed-point arithmetic in HOL, Formal Methods Syst. Des. 27 (1–2) (2005) 173–200.

[24] A. Souari, A. Gawanmeh, S. Tahar, M. Lassaad, Design and Verification of the Frequency Domain Equalizer, Technical Report, Electrical and Computer Engineering Department, Concordia University, Montreal, Quebec, Canada, 2013.

[25] M. Gordon, T. Melham, Introduction to HOL: A Theorem Proving Environment for Higher Order Logic, Cambridge University Press, 1993.

[26] G. Melquiond, De l'arithmétique d'intervalles à la certification de programmes, Ph.D. Thesis, École Normale Supérieure de Lyon, France, 2006.