Number 4 Volume 17 August 2011 Journal of Engineering

IMPLEMENTATION OF ROOT FINDING ALGORITHM OF MINIMUM PHASE FILTER USING VHDL

Zainab Noori Ghanim **Buthaina Mosa Omran**

Malathe Salah Al-Deen

University of Baghdad College of Engineering Electronics & Communications Department

ABSTRACT

Root-finding is an oldest classical problem, which is still an important research topic, due to its impact on computational algebra and geometry. In communications systems, when the impulse response of the channel is minimum phase the state of equalization algorithm is reduced and the spectral efficiency will improved. To make the channel impulse response minimum phase the prefilter which is called minimum phase filter is used, the adaptation of the minimum phase filter need root finding algorithm. In this paper, the VHDL implementation of the root finding algorithm introduced by Clark and Hau is introduced.

VHDL program is used in the work, to find the roots of two channels and make them minimum phase, the obtained output results are similar in accuracy to the past work results, which is built by using MATLAB program. Using VHDL is necessary in FPGAs for building hardware of the root finding algorithm in lower cost and time. MATLAB program is used only for displaying the input and output discrete signals of tested channels.

الخلاصة

ايجاد الجذور هي مسالة قديمة وتبقى محور بحثي مهم . ان في انظمة الاتصالات عندما تكون عينات الاستجابة للقناة ذات طور اصغر فانه يختصر عمل المعادل ويحسن كفائته . ان مرشحا يستعمل لجعل عينات الاستجابة للقناة ذات طور اصغر ، يسمى بالمرشح ذو الطور الاصغر. عملية تحديث المرشح تحتاج طريقة خاصة لايجاد الجذور .

في هذا البحث فان VHDL تم استعماله لايجاد هذه الجذور باستعمال طريقة Clark and Hau . برنامج VHDL تم استعماله لايجاد جذور قناتين و تحويلهما الى قناتين ذاتي طور اصغر و كانت النتائج مشابهة في دقتها و تحقيقها الغرض المطلوب لنتائج عمل سابق باستخدام برنامج MATLAB . استخدامVHDL هو ضروري في FPGA لبناء دائرة الكترونية لايجاد الجذور بكلفة و وقت اقل. لقد تم استخدام برنامج MATLAB فقط لغرض عرض عينات الاستجابة للقناتين قبل تحويلهما الى طور اصغر و بعد التحويل.

1. Introduction

The minimum-phase and the all-pass filters have over the years attracted much attention due to their broad applicability in signal processing. One area where the minimum-phase filter is widely used is in digital communications over multipath channels where higher-order modulation schemes are employed. In such scenarios the optimal symbol-by-symbol or sequence detector will often require a very high complexity, due to its exponential growth in complexity as a function of the filter length. Furthermore, in multi user detection the complexity grows further, since the number of users will also influence the complexity exponentially. Thus, suboptimal schemes, such as delayed decision feedback or reduced-state sequence estimation will often be applied in such systems instead. However, in order to ensure acceptable performance of these schemes, both the minimum-phase filter and the associated all pass filter are usually needed, since the minimum-phase filter provides the highest possible energy concentration in the beginning of the filter impulse response.[1].

For any suboptimum trellis based equalizer, a discrete time minimum-phase overall impulse response is essential for high performance. In general, this necessitates the introduction of a discrete time prefilter in front of the equalizer, which changes the channel impulse response into its minimum phase equivalent [2]. The, the shape of the impulse response of the channel is very important in the detection process. The first sample of sampled impulse response of the channel must be relatively large. The effect of adaptive linear filter on the channel is such that the resultant response of the channel and filter becomes minimum phase with essence that the energy of the pulse is concentrated towards the front portion of the pulse [3]. The adjustment of the linear filter is the sum of the time taken for the algorithm to find all (or some) of the roots outside the unit circle and the time required to form the filter [3].

In the cases where the linear phase property is not strictly required, much lower time delay can be achieved with the use of minimum phase (MP) filters that satisfy the same amplitude specifications. Having designed the high-order LP filter it is necessary to find its Z-plane roots, and fold all zeros that lie outside the unit circle to their reciprocal radius position [4].

Finding the root of a function is arguably the oldest and the most important problem in numerical mathematics. An interesting situation occurs when we do not know this function and can only observe the values of it with some error. This problem has numerous applications in science and engineering. The problem becomes very complicated when the true relationship is highly nonlinear and the measurements are extremely noisy. Some other applications of stochastic root finding include the quantile estimation problem in bio-assay experiments, quality and reliability improvement, sensitivity experiments and adaptive control and signal processing. [5]

Often it will not be possible to solve nonlinear equation root-finding problems analytically. When this occurs we turn to numerical methods to approximate the solution. The methods employed are usually *iterative*. Generally speaking, algorithms for solving problems numerically can be divided into two main groups: *direct methods* and *iterative methods*. Direct methods are those which can be completed in a predetermined finite number of steps. Iterative methods are methods which converge to the solution over time. These algorithms run until some convergence criterion is met. When choosing which method to use one important consideration is how quickly the algorithm converges to the solution or the method's *convergence rate*.[6].

In this paper the root finding algorithm introduced by Clark and Hau [7] is considered and implemented using VHDL program. FPGA is on the verge of revolutionizing digital signal processing and VHDL program help us to build the components of the FPGAs. Many front-end digital signal processing algorithms are now most often replaced by FPGA. This allows users to by pass the hardware design engineer leading to a significant reduction in development time and cost [8]. Two wireless channels are tested, which they have non-minimum phase response and have roots outside the unit circle. After applying the root finding algorithm these roots are replaced by their complex conjugate and the channels become minimum-phase channels.

2. MINIMUM PHASE FILTER

The block diagram of the system with minimum phase filter is shown in fig. (1). [2] Consider the z-transform of the 'sampled impulse response' of the channel [9]:

$$Y(z) = y_0 + y_1 z^{-1} + \dots\dots\dots\dots + y_g z^{-g} \quad (1)$$

Suppose now that:

$$Y(z) = Y_1(z)\, Y_2(z) \quad (2)$$

where

$$Y_1(z) = \eta\, (1 + \alpha_1 z^{-1})\, (1 + \alpha_2 z^{-1}) \dots\dots\dots\dots (1 + \alpha_{g-m} z^{-1}) \quad (3)$$

and

$$Y_2(z) = z^{-m}\, (1 + \beta_1 z)\, (1 + \beta_2 z) \dots\dots\dots\dots (1 + \beta_m z) \quad (4)$$

It is assume here that no roots (zeros) of Y(z) lie exactly on the unit circle in the z-plane. Also $|\alpha_i| < 1$ and $|\beta_i| < 1$, where α_i is the negative of a root Y(z), β_i is the negative of the reciprocal of a root of Y(z), and η is the appropriate complex value needed to satisfy equation (2), (3) and (4). $|\alpha_i|$ and $|\beta_i|$ are absolute values of α_i and β_i respectively. In applications where Y(z) has one or more roots on the unit circle, these are taken to be roots of $Y_1(z)$, such that in each case $|\alpha_i| = 1$.

The adaptive linear filter is always confined to be an all pass filter and, in its ideal form, adjust the sample impulse response of the channel and filter to be minimum phase such that in the z-transform of the resultant sampled impulse response, all the roots (zeros) lie inside the unit circle in the z-plane [3].

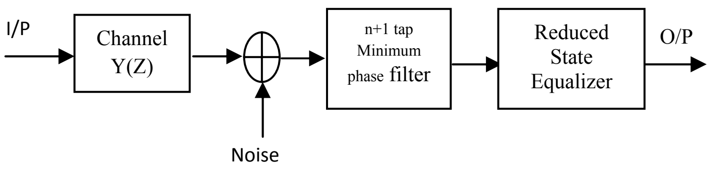

Fig. (1): Block Diagram of the System with Minimum Phase Filter

The adaptive linear transversal filter has n+1 taps. It will, for convenience, be assumed here that the filter is adjusted to its ideal form. The z-transform of the sampled impulse response of the filter is now approximately [3]:

$$M(z) = z^{-n} Y_2^{-1}(z) Y_3(z) \quad (5)$$

where

$$Y_3(z) = (1 + \beta_1^* z^{-1})(1 + \beta_2^* z^{-1}) \dots\dots\dots\dots\dots\dots (1 + \beta_m^* z^{-1}) \quad (6)$$

and β_i^* is the complex conjugate of β_i. Thus, the z-transform of the channel and linear filter is approximately [3]:

$$F(z) = Y(z)M(z)$$

$$= z^{-n}Y_1(z)Y_3(z) \qquad (7)$$

Clearly, all roots of F(z) lie inside the unit circle in the z-plane, which means that the channel and the linear filter together have a response that is minimum phase. The roots of $Y_3(z)$ are the complex conjugate of the reciprocal of the roots of $Y_2(z)$, so that the linear filter replaces all roots of Y(z) that lie outside the unit circle by the complex conjugate of their reciprocals, leaving the remaining roots unchanged. Since the receiver is assumed to know Y(z), it can in principle evaluate M(z) and hence F(z). This involves the determination of the roots of Y(z) that lie outside the unit circle, with no restriction in time or complexity, which would achieved by any conventional root finding algorithm[3].

Clark and Hau introduced an algorithm to determine the wanted roots of Y(z) and at the same time evaluate M(z) and F(z) [7]. The algorithm is discussed in the next section.

3.Root finding algorithm [3, 7, 9]

- First the receiver holds in store the sequence Y and an estimate λ_i of the quantity β_1. The first estimate of β_1 at the start of the process is one of a number of different starting points. The nine starting points given in Table (A1) are having been found to be the optimal starting points.
  Having determined λ_i the receiver appropriately adjusts the one tap feedback transversal filter shown in fig. (2) with z-transform:

$$A_i(z) = (1 + \lambda_i z)^{-1} \qquad (8)$$

- The stored sequence Y is now reversed in order, so that it starts with the component y_g and it is fed through the feedback transversal filter. The sequence Y passing though the filter in reverse order, is taken to be moving backwards in time starting with the component y_0 at time t=0. The delay of one sampling interval T in the feedback filter now becomes an advance of T with z-transform z. Thus, the effective z-transform of the feedback filter becomes $A_i(z)$ and the output from the filter is the sequence of the $\{e'_{i,h}\}$, only the g+1 components of this sequence are in fact generated. An improved estimate of β_1 is now given by:

$$\beta_1 \cong \lambda_i + \frac{e'_{i,0}}{\varepsilon_i} \qquad (9)$$

where

$$\varepsilon_i = e'_{i,1} - e'_{i,2}\lambda_i + e'_{i,3}\lambda_i^2 \ldots\ldots\ldots\ldots. e'_{i,g}(-\lambda)^{g-1} \qquad (10)$$

This means that if c is a positive constant in the range of 0 to 1, then

$$\lambda_{i+1} = \lambda_i + c\frac{e'_{i,0}}{\varepsilon_i} \qquad (11)$$

This gives a new tap feedback transversal filter, with λ_i replace by λ_{i+1}. The effective z-transform of this filter, when operating on the sequence Y in reverse order is:

$$A_{i+1}(z) = (1 + \lambda_{i+1} z)^{-1} \qquad (12)$$

and the coefficients of z^{-h} in $Y(z)A_{i+1}(z)$ is $e'_{i+1,h}$. The iterative process continues in this manner until the term $e'_{i,0}/\varepsilon_i$ in equation (11) satisfies:

$$\left|\frac{e'_{i,0}}{\varepsilon_i}\right|^2 < d \qquad (13)$$

where d is an appropriate small positive real constant or else until either i=40 or $|\lambda_i| < 1$ and in each case the process is terminated. When equation (13) is satisfied the iterative process is taken to have converged. Let the value of i at convergence be k so that:

$$\lambda_k \approx \beta_1 \qquad (14)$$

β_1 is the negative of the reciprocal of the first root of Y(z) to be processed by the system and of course $|\beta_1|$ <1. Since the filter with z-transform is not limited to zero and negative power of z. At the end of iteration process, when i =k, the z-transform of the filter is:

$$A_k(z) = (1+\beta_1 z)^{-1} \qquad (15)$$

- The receiver next appropriately adjusts the two-tap feedforward transversal filter shown in fig. (3), which has the z-transform:

$$B_k(z) = 1+\lambda_k^* z^{-1} \qquad (16)$$

The sequence of the {e'$_{i,k}$}for h=0, 1,…..g is now fed through this filter in the correct order. This gives the output sequence (g+2) components with the z-transform:

$$f_{1,-1} + f_{1,0}z^{-1} + \dots\dots + f_{1,g}z^{-g-1} \qquad (17)$$

Which is approximately equal to Y(z)A$_k$(z)B$_k$(z) and where $f_{1,-1} = 0$. The resultant effect on the sequence Y of the two filters giving the sequence of the $\{f_{1,h}\}$ approximately to that of a single filter with Z-transform:

$$C_1(z) = A_k(z)B_k(z) \approx (1+\beta_1 z)^{-1}(1+\beta_1^* z^{-1}) \qquad (18)$$

- The output sequence of $\{f_{1,h}\}$is advanced by one place (sampling interval) and the first component $f_{1,-1}$ discarded to give the sequence F$_1$ with z-transform:

$$F_1 = f_{1,0} + f_{1,1}z^{-1} + \dots\dots + f_{1,g}z^{-g} \approx zC_1(z)Y(z) \qquad (19)$$

For practical purposes the linear factor $(1+\beta_1 z)$is replaced in F(z) by the linear factor $(1+\beta_1^* z^{-1})$. Thus, the root (-1/β$_1$) of Y(z) is replaced by the root (-β_1^*), which is the complex conjugate of its reciprocal and lies inside the unit circle. F$_1$(z) contains, in addition an advance of one sampling interval. The sequence F$_1$ (with z-transform F$_1$(z)) is an estimate of the sampled impulse response of the channel and adaptively linear transversal filter, when the z-transform of the latter is zC$_1$(z). The iterative process is repeated with Y replaced by F$_1$ for the tracking of moor roots.

- The whole procedure is now repeated, but using F$_1$(z) in place of Y(z). At the end of the iterative process $\lambda_k \approx \beta_2$ and the value of λ$_k$ and F$_1$(z) determines F$_2$(z) which is used in place of F$_1$(z) for processing β$_3$.

- The system continues in this way until, on the hth repetition of the whole procedure, no roots of F$_h$(z) outside the unit circle are found, starting from any of the nine possible values of λ_i. In the case where, from each starting point, $|\lambda_i| > 1$ for some value of i, or i reaches 40, it will be assumed that h=m. This being so, all the roots of Y(z) that lie outside the unit circle are replaced by the complex conjugate of their reciprocals, in the z-transform of the channel and adaptive filter.

The estimate of the sampled impulse response of the channel and the adaptive filter that are employed by the detector is the sequence F_m with z-transform:

$$F_m(z) = f_{m,0} + f_{m,1}z^{-1} + \dots\dots\dots + f_{m,g}z^{-g}$$

$$\approx zF(z) \approx Y_1(z)Y_3(z) \qquad (20)$$

Table (1): The Nine Possible Starting Points

Root No.	Root Value
No.1	0.01
No.2	0.909
No.3	-j0.909
No.4	j0.909
No.5	-0.909
No.6	0.643-j0.643
No.7	0.643+j0.643
No.8	-0.643-j0.643
No.9	-0.643+j0.643

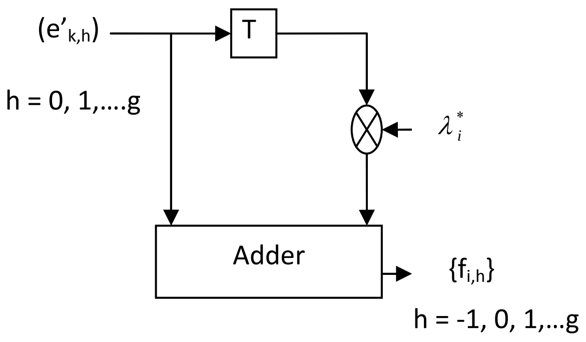

Fig. (2): One-Tap Feedback Transversal Filter

4. Design of Finding Roots in Minimum Phase

VHDL program provides language constructs for parametrizing and customizing designs, and for definition and usage of design libraries. These constructs enable a designer to generate a functional design independent of the specific technology and customize this generic design at a later stage. Specifically, library, use clause, package, and configuration declarations of VHDL are used for grouping or categorizing various components into design libraries and for customizing designs to use components in these libraries [10, 11].

The design is processed by using VHDL program. The main finding roots part of minimum phase component is shown in fig.(4).

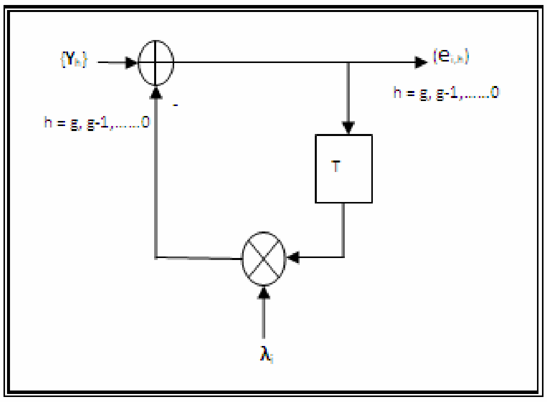

Fig. (3):Two-Tap Feedforward Transversal Filter.

This component reads the channel impulses from a text file (input data text file) and converts the complex numbers of channel impulses to binary values by specified procedure. This main level gives binary values of channel to the second level and receives edited channel impulses from the second level to be the output of this component. The edited channel impulses are converted from binary values to complex numbers and written in a text file (output data text file).

The second level has three components as shown in fig.(5). These components are:

A. **First Part Finding Roots component** :

The internal structure of this component is shown in fig.(6) and its algorithm as follows :

- Take the channel impulses (Y) from the first level with clock to start new operation with each clock.
- Read lamda (λ_i) initial value in each time from look up table, which has nine initial values for lamda.
- The Y and λ_i are used as inputs to three internal components in the first part component, see fig.(6). The internal components compute error matrix ($e'_{i,h}$) , epsilon (ε_i) and new lamda (λ_{i+1}) as output data.
- Check if $\left|\frac{e'_{i,0}}{\varepsilon_i}\right|^2 < 10^{-10}$ .If the condition is true the first part is stopped and gives output data (error matrix, lamda value and clock) to other second level components (Find Conjugate and Two Tap) to begin its work. But if the condition is false the first part repeats the operation of computing new error matrix, epsilon and new lamda, where new lamda is used as input with channel impulses to the three components in new iteration. The condition is checked in each iteration. This continues iteration process is stopped if the absolute value of lamda is more than one or the number of iterations be fourty, where new initial value of lamda is taken in this case to repeat the same process by using the second value of look up table of initial values until the first condition will be true or ending the look up table without finding any roots.
- Give restart signal to the three components at each new iteration.

The three internal components of the First Part component are:

1. One Tap: this component applied eq.(8) in its work. Channel impulses Y, λ_i, clock and restart are the inputs for this component. Error matrix $e'_{i,h}$ and clock to the second component are the output of it. Its internal structure is shown in fig.(7) and its algorithm as follows:

- Start work when clock signal, Y and λ_i are received.
- Subtract feedback value from each value of Y in each time to obtain error value related to this value of Y until the error matrix is obtained with the same size of Y. Subtraction component is used for this purpose.
- Each value of error will be delayed by D_latch component.
- The old value of error is multiplied with λ_i by a multiplier component to obtain feed back value.
- The error matrix and a clock signal will be the output of One Tap component and are the inputs to the next component (Epsilon Computing component).

2. Epsilon Computing: this component finds the value of epsilon ε_i as in eq.(10). The component internal structure is shown in fig. (8). Its algorithm is as follows:

- Start new work when clock signal and new error matrix are received from One Tap.
- Find the value of powered lamda by computing lamda to the power **i** as in eq. (10) by a specified procedure.

- Multiply powered lamda with each value of error in error matrix (0-9) by a multiplier component.
- Add the result value of multiplier to the recent value of epsilon to obtain a new value of epsilon by using an adder component.
- The previous steps are repeated for each value of error to find the final value of epsilon to be an output of Epsilon Computing component.
- Clock signal and ε_i are given to the next component (Lamda Computing).

3. Lamda Computing: this component finds the new value of lamda λ_{i+1}, where the new value of lamda will be an old value in the next iteration as in eq.(11). Its internal structure is shown in fig.(9) and its algorithm is as follows :

- Divide the first value of error matrix (the output of One Tap) by the epsilon (output of Epsilon Computing) by using a divider component.
- The divider output will be multiplied with the constant c as in eq.(11). A multiplier component is used to find the multiplication result.
- The output of multiplier will be added to the recent value of lamda by an adder component to find new lamda. New lamda λ_{i+1} will be the output of Lamda Computing component.

B. Find Conjugate component:

This component is designed to find the conjugate value of any complex number and it's used here to find the conjugate of lamda, which realizes the previous condition. The output conjugate value will be the input to the Two Tap component.

C. Two Tap component:

This component has three inputs, which are error matrix, conjugate of lamda λ^*_i and clock and has one output. The edited channel impulses matrix (F) is the output of this component. Eq.(16) explains the work of this component. The internal structure of this component is shown in fig. (10) its algorithm is as follows:

- Multiply the lamda conjugate with the first value of error matrix by a multiplier component.
- Add the result of multiplication to the next value of error matrix by an adder component to find the first value of edited channel impulses matrix.
- Repeat the previous steps until the whole matrix of channel is found by taking the whole error matrix.
- The Two Tap begins new work with each received clock

The main finding roots component repeat the work ten times as minimum by taking Y matrix from the text value at first time, but when the root is found and Two Tap component works as a result, F the output of two tap will be the input in the next time instead of Y from file.

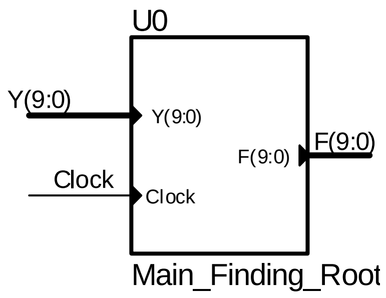

Fig.(4): Main Finding Root component architecture

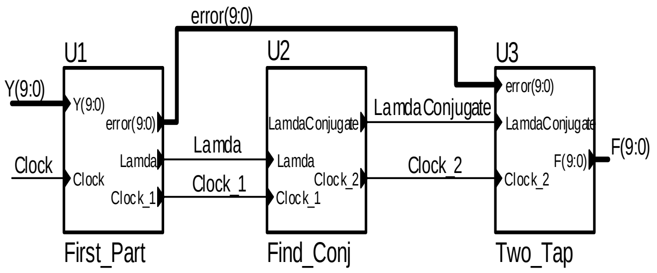

Fig.(5): Internal structure of Main Finding Root component

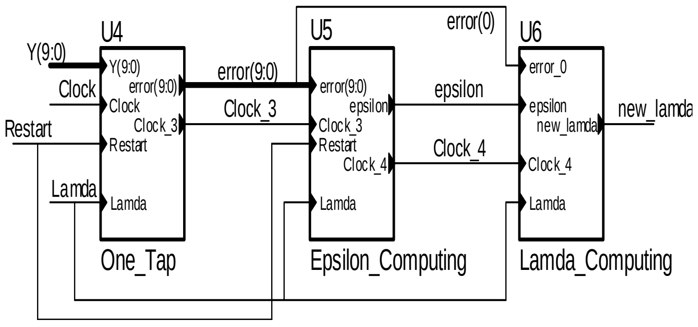

Fig.(6): Internal structure of First Part component

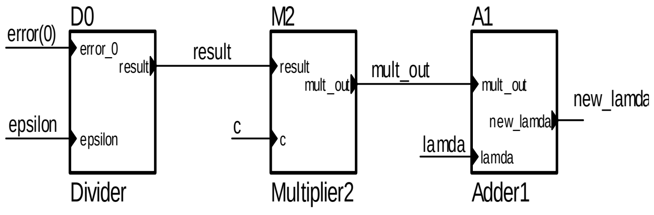

Fig.(9): Internal structure of Lamda Computing component.

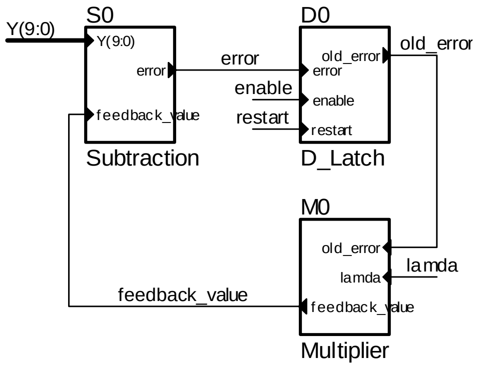

Fig.(7): Internal structure of One Tap.

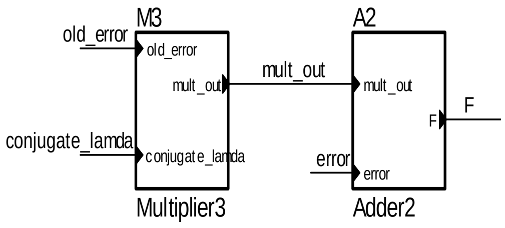

Fig.(10): Internal structure of Two Tap.

5. Results

Two non-minimum phase channels are used in the test, each of them has ten taps. The impulse response of them is shown in figs. (11) and (13). The roots of them are shown in figs. (15) and (17). All roots of the first channel are outside the unit circle, while the second channel has eight roots outside the unit circle and only one root inside the unit circle. The impulse responses of the two channels after using the minimum phase filter are shown in figs. (12) and (14). The proposed VHDL circuit reflects the roots of the two channels from outside the unit circle to inside it as shown in figs. (16) and (18). The input and output impulse response, which are written in input data text file and output data text file are plotted using MATLAB program.

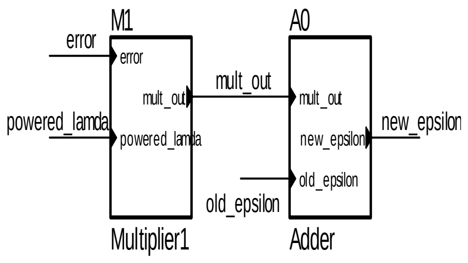

Fig.(8): Internal structure of Epsilon Computing component

6. Conclusion

The work with VHDL program gives good output signals by reflecting the roots from outside to inside the unit circle. The synchronization among the components is necessary in this work and the VHDL program offers the methods to synchronize the system.

As in any high level language, VHDL allows the definition and usage of functions and procedures. In addition to the important hardware implications of subprograms, these language constructs greatly improve readability and organization of a hardware description.

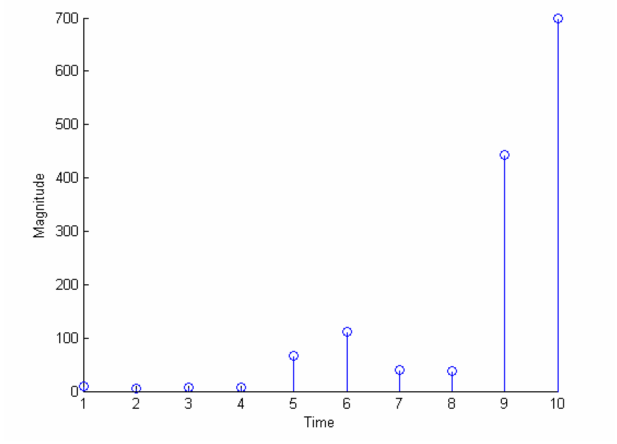

Fig.(11) Impulse response of channel 1

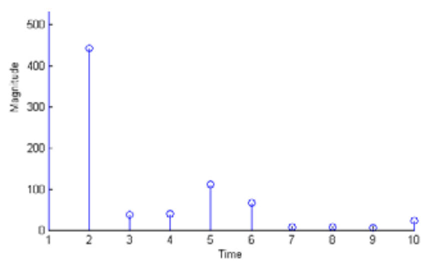

Fig. (12): Impulse response of combined channel 1 and minimum phase filter.

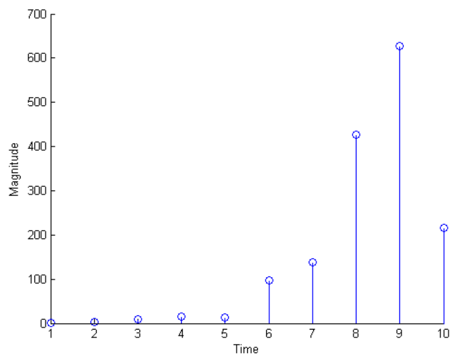

Fig. (13): Impulse response of channel 2

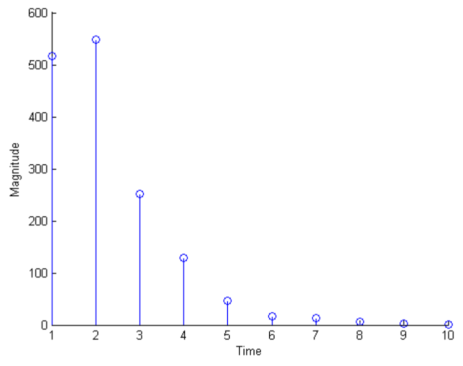

Fig. (14) Impulse response of combined channel 2 and minimum phase filter

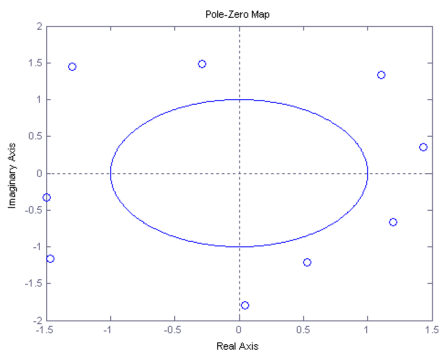

Fig. (15): Roots of channel 1

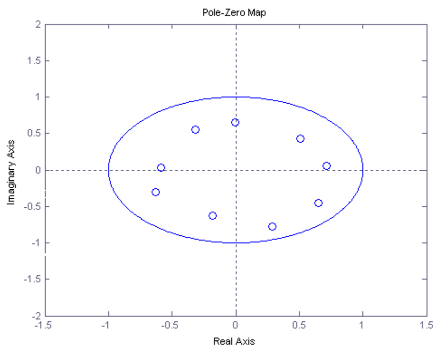

Fig. (16): Roots of channel 1 after minimum phase filter

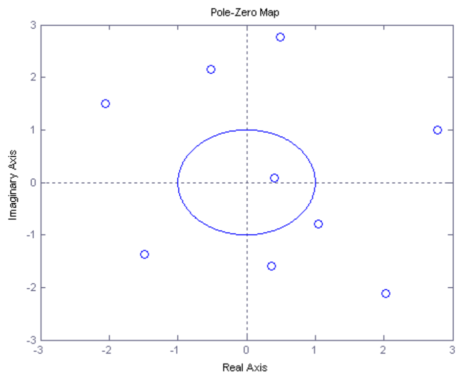

Fig. (17): Roots of channel 2.

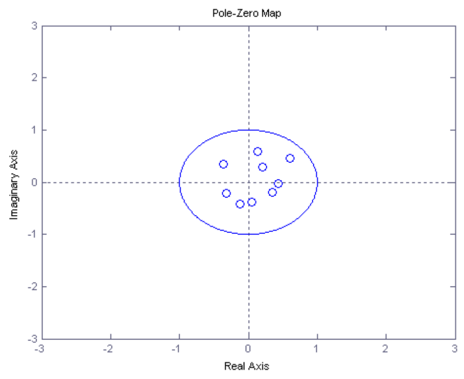

Fig. (18): Roots of channel 2 after minimum phase filter

References:

[1] Morten Hansen, Lars P. B. Christensen, and Ole Winther **"Computing the Minimum-Phase Filter Using the QL-Factorization"**, IEEE Transaction on Signal Processing, Vol. 58, No. 6, June 2010.

[2] Wolfgang H. Gerstacker, Frank Obernosterer, Raimund Meyer, and Johannes B. Huber, " **On Prefilter Computation for Reduced-State Equalization**", IEEE Transaction on Wireless Communications, Vol. 1, No. 4, October 2002'

[3] Hau,S.F. **"Adaptive Adjustment of Modern Receiver for HF Radio Links"**, Ph.D. Thesis, Loughborough University of Technology, 1986.

[4] Gordana Jovanovic-Dolecek and Vlatko Dolecek, **"Narrowband Minimum Phase Filter Design Using Sharpened Comb Filter",** ECCTD'01 - European Conference on Circuit Theory and Design, August 28-31, 2001, Espoo, Finland.

[5] V. Roshan Joseph, Yubin Tian nd C. F. Jeff Wu,**" Adaptive Designs For Stochastic Root-Finding",** Statistica Sinica 17(2007), 1549-1565**.**

[6] KAREN A. KOPECKY," **Root-Finding Methods**", Lecture Notes ECO 613/614 FALL 2007.

[7] Clark, A.P and Hau, S.F, **"Adaptive adjustment of receiver for distorted digital signals"**, IEE Proc. 1984,131, pt. F. pp.526-536.

[8] Uwe Meyer-Baese[a], A. Vera[b], A. Meyer-Baese[a], M. Pattichis[b], R. Perry[a a], FAMU-FSU, **"Discrete Wavelet Transform FPGA Design using Matlab/simulink",** ECE Dept., 2525 Pottsdamer Street, Tallahassee, FL

USA-32310; [b] University of New Mexico, ECE Dept., Albuquerque, NM 87131, 2006.

[9] Buthaina Mosa Omran, **"Improvement Techniques for Satellite Digital Video Broadcasting Using OFDM",** Ph.D. Thesis, University of Baghdad, 2007.

[10] Zainalabedin Navabi, **"VHDL Analysis and Modeling of Digital Systems",** McGraw-Hill Series in Electrical and Computer Engineering, 1993.

[11]Mark Zwolinski, "**Digital System Design with VHDL**", Prentice Hall, 2000.

List of Abbreviations:

FPGAs : Field Programmable gate arrays.
VHSIC : Very High Speed Integrated Circuit.
VHDL : VHSIC Hardware Description Language.