A processor is provided having an instruction set with user-defined non-linear functions for digital pre-distortion (DPD) and other non-linear applications. A signal processing function, such as DPD, is implemented in software by obtaining at least one software instruction that performs at least one non-linear function for an input value, x, wherein the at least one non-linear function comprises at least one user-specified parameter; in response to at least one of the software instructions for at least one non-linear function having at least one user-specified parameter, performing the following steps: invoking at least one functional unit that implements the at least one software instruction to apply the non-linear function to the input value, x; and generating an output corresponding to the non-linear function for the input value, x. The user-specified parameter can optionally be loaded from memory into at least one register.

![Diagram](image-url)
FIG. 1

BASEBAND TX INTERFACE

CHANNEL FILTERS/CHANNEL DUC

CFR

DPD

EQUALIZER/Q IMBALANCE CORRECTION

TO D/A CONVERTER AND ANTENNA
FIG. 3

FOR \( i = 0; i < 256; i++ \)

\[
\begin{align*}
\{ & \text{LOAD 16 INPUT SAMPLES (16b/16b COMPLEX)} \\
& \text{COMPUTE } |x|^{2} \text{ FOR 16 SAMPLES} \\
& \text{STORE 16 SQUARED SAMPLES } x^2 (16b/16b \text{ COMPLEX})
\}
\]

1 CYCLE FOR 16 SAMPLES, LOAD/MULT/STORE PIPELINED IN PARALLEL

FOR \( m = 0; m < 20; m++ \)

\[
\begin{align*}
\{ & \text{LOAD 128 COEFFICIENTS FOR PIECEWISE POLYNOMIAL } f_m \\
& \text{FOR } i = 0; i < 250; i++16
\}
\]

\[
\begin{align*}
\{ & \text{LOAD 16 INPUT SAMPLES AT OFFSET } u(m) \\
& \text{LOAD 16 SQUARED SAMPLES } x^2 \text{ AT OFFSET } v(m) \\
& \text{COMPUTE NONLINEAR FUNCTION } f_m(x^2(v(m))) \\
& \text{MULTIPLY/ACCUMULATE } x(u(m)) \cdot f_m
\}
\]
FIG. 7

700

\[ \begin{align*}
    x_0 & \rightarrow f_0(\cdot) \rightarrow y_0 \\
    x_1 & \rightarrow f_1(\cdot) \rightarrow y_1 \\
    x_2 & \rightarrow f_2(\cdot) \rightarrow y_2 \\
    x_{15} & \rightarrow f_{15}(\cdot) \rightarrow y_{15}
\end{align*} \]

USER SPECIFICATION

710-1 \rightarrow 710-2 \rightarrow 710-3 \rightarrow 710-N
PROCESSOR HAVING INSTRUCTION SET WITH USER-DEFINED NON-LINEAR FUNCTIONS FOR DIGITAL PRE-DISTORTION (DPD) AND OTHER NON-LINEAR APPLICATIONS

CROSS-REFERENCE TO RELATED APPLICATIONS


FIELD OF THE INVENTION

[0002] The present invention is related to digital signal processing techniques and, more particularly, to techniques for evaluating user-defined non-linear functions.

BACKGROUND OF THE INVENTION

[0003] Digital pre-distortion (DPD) is a technique used to linearize a power amplifier in a transmitter to improve the efficiency of the power amplifier. A power amplifier in a transmitter typically must be substantially linear, so that a signal is accurately reproduced. Compression of the input signal or a non-linear relationship between the input signal and output signal causes the output signal spectrum to spill over into adjacent channels, causing interference. This effect is commonly referred to as spectral regrowth.

[0004] A digital pre-distortion circuit inversely models the gain and phase characteristics of the power amplifier and, when combined with the amplifier, produces an overall system that is more linear and reduces distortion that would otherwise be caused by the power amplifier. An inverse distortion is introduced into the input of the amplifier, thereby reducing any non-linearity that the amplifier might otherwise have.

[0005] Digital pre-distortion is typically implemented using hardwired logic due to the high sampling rate. While such hardware-based DPD techniques effectively linearize a power amplifier they suffer from a number of limitations, which if overcome, could further improve the efficiency and flexibility of DPD circuits. For example, existing hardware-based DPD techniques lack flexibility and it is expensive, time consuming and challenging to modify the DPD design for a new RF design.

[0006] Digital pre-distortion and other non-linear applications must often process one or more non-linear functions that include one or more parameters specified by a user, such as filter coefficient values or values from a look-up table. A need therefore exists for a processor having an instruction set with one or more user-defined non-linear functions for digital pre-distortion (DPD) and other non-linear applications to enable, for example, a high performance software implementation of DPD.

SUMMARY OF THE INVENTION

[0007] Generally, a processor is provided having an instruction set with user-defined non-linear functions for digital pre-distortion (DPD) and other non-linear applications. According to one aspect of the invention, a signal processing function, such as DPD, is implemented in software by obtaining at least one software instruction that performs at least one non-linear function for an input value, x, wherein the at least one non-linear function comprises at least one user-specified parameter; in response to at least one of the software instructions for at least one non-linear function having at least one user-specified parameter, performing the following steps: invoking at least one functional unit that implements the at least one software instruction to apply the non-linear function to the input value, x; and generating an output corresponding to the non-linear function for the input value, x.

[0008] In addition, the user-specified parameter can optionally be loaded from memory into at least one register. The user-specified parameter may comprise a look-up table storing values of the non-linear function for a finite number of input values. The user-specified parameter may comprise one or more coefficients employed by the non-linear function to perform polynomial interpolation between entries of the look-up table.

[0009] A more complete understanding of the present invention, as well as further features and advantages of the present invention, will be obtained by reference to the following detailed description and drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

[0010] FIG. 1 illustrates portions of an exemplary transmitter in which aspects of the present invention may be employed;

[0011] FIG. 2 illustrates portions of an alternate exemplary transmitter in which aspects of the present invention may be employed;

[0012] FIG. 3 illustrates exemplary pseudo code to implement a DPD function in software on a vector processor of 16 component vectors using a user-defined non-linear instruction \( f_{\text{DPD}} \);

[0013] FIGS. 4A and 4B are graphical illustrations of exemplary functional block diagrams;

[0014] FIG. 5A illustrates an individual user-defined non-linear function \( f_{\text{non-linear}} \) as a function of \( x(n) \);

[0015] FIG. 5B illustrates an exemplary approximation of the individual user-defined non-linear function \( f_{\text{non-linear}} \) of FIG. 5A;

[0016] FIG. 6 illustrates a Taylor Sum computation block; and

[0017] FIG. 7 is a schematic block diagram of an exemplary vector-based digital processor that evaluates a user-defined non-linear function for one or more complex numbers simultaneously in accordance with an embodiment of the present invention.

DETAILED DESCRIPTION

[0018] FIG. 1 illustrates portions of an exemplary transmitter 100 in which aspects of the present invention may be employed. As shown in FIG. 1, the exemplary transmitter portion 100 comprises a channel filter and digital up conversion (DUC) stage 110, a crest factor reduction (CFR) stage 120, a digital pre-distortion (DPD) stage 130 and an optional equalization and/or IQ imbalance correction stage 140. Generally, the channel filter and digital up conversion stage 110 performs channel filtering using, for example finite impulse response (FIR) filters and digital up conversion to convert a digitized baseband signal to an intermediate frequency (IF). As indicated above, the crest factor reduction stage 120 limits the peak-to-average ratio (PAR) of the transmitted signal. The digital pre-distortion stage 130 linearizes the power amplifier.
to improve efficiency. The equalization stage 140 employs RF channel equalization to mitigate channel impairments.

[0019] According to one aspect of the invention, digital pre-distortion non-linear processing and other non-linear applications are performed in software on a processor having an instruction set with one or more user-defined non-linear functions pre-distortion (DPD) and other non-linear applications. A user-defined non-linear instruction is used to compute a non-linear function having at least one parameter that must be specified by a user. The user-defined non-linear function receives with an input scalar or vector and produces an output scalar or vector. In the event of an input vector, the output vector comprises output samples that are non-linear function of the input samples.

[0020] While the present invention is illustrated in the context of digital pre-distortion, the present invention can be used for any non-linear application employing one or more user-defined non-linear functions.

[0021] The present invention can be applied in handsets, base stations and other network elements.

[0022] FIG. 2 illustrates portions of an alternate exemplary transmitter 200 in which aspects of the present invention may be employed. As shown in FIG. 2, the exemplary transmitter portion 200 comprises two pulse shaping and low pass filter (LPF) stages 210-1, 210-2 and two digital up-converters 220-1, 220-2 which process a complex signal, Q. The exemplary transmitter portion 200 of FIG. 2 does not include the crest factor reduction stage 120 of FIG. 1, but a CFR stage could optionally be included. The complex input (I,Q) is then applied to a digital pre-distorter 230 of FIG. 2 and is the focus of the exemplary embodiment of the invention. The digital pre-distorter 230 of FIG. 2 is discussed further below, for example, in conjunction with FIGS. 3 and 4.

[0023] The output of the digital pre-distorter 230 is applied in parallel to two digital analog converters (DACs) 240-1, 240-2, and the analog signals are then processed by a quadrature modulation stage 250 that further up converts the signals to an RF signal.

[0024] The output 255 of the quadrature modulation stage 250 is applied to a power amplifier 260, such as a Doherty amplifier or a drain modulator. As indicated above, the digital pre-distorter 230 linearizes the power amplifier 260 to improve the efficiency of the power amplifier by extending its linear range to higher transmit power.

[0025] In a feedback path 265, the output of the power amplifier 260 is applied to an attenuator 270 before being applied to a demodulation stage 280 that down converts the signal to baseband. The down applied to an analog to digital converter (ADC) 290 to digitize the signal. The digitized samples are then processed by a complex adaptive algorithm 295 that generates parameters w for the digital pre-distorter 230. The complex adaptive algorithm 295 is outside the scope of the present application. Known techniques can be employed to generate the parameters for the digital pre-distorter 230.

[0026] Non-Linear Filter Implementation of Digital Pre-Distorter

[0027] A digital pre-distorter 230 can be implemented as a non-linear filter using a Volterra series model of non-linear systems. The Volterra series is a model for non-linear behavior in a similar manner to a Taylor series. The Volterra series differs from the Taylor series in its ability to capture “memory” effects. The Volterra series can be used to approximate the response of a non-linear system to a given input if the output of this system depends strictly on the input at that particular time. In the Volterra series, the output of the non-linear system depends on the input to the system at other times. Thus, the Volterra series allows the “memory” effect of devices to be captured.

[0028] Generally, a causal system with memory can be expressed as:

$$y(t) = \sum_{\tau=0}^{\infty} g_{\tau}(t)$$

[0029] In addition, a weakly non-linear system without memory can be modeled using a polynomial expression:

$$y(t) = \sum_{\tau=0}^{\infty} a_{\tau} b(t)^{\tau}$$

[0030] The Volterra series can be considered as a combination of the two:

$$y(t) = \sum_{\tau=0}^{\infty} g_{\tau}(t)$$

$$y(t) = \sum_{\tau=0}^{\infty} a_{\tau} b(t)^{\tau}$$

[0031] In the discrete domain, the Volterra Series can be expressed as follows:

$$y(n) = \sum_{\tau=0}^{\infty} g_{\tau}(n)$$

$$y(n) = \sum_{\tau=0}^{\infty} a_{\tau} b(n)^{\tau}$$

[0032] The complexity of a Volterra series can grow exponentially making its use impractical in many common applications, such as DPD. Thus, a number of simplified models for non-linear systems have been proposed. For example, a memory polynomial is a commonly used model:

$$y_{mp}(n) = \sum_{k=0}^{K} \sum_{m=0}^{M} h_{km}(m^{k}x(n-m))$$

[0033] Another simplified model referred to as a Generalized Memory Polynomial Model, can be expressed as follows (where NI indicates the memory depth and K indicates the polynomial order):

$$y(n) = \sum_{k=0}^{K} \sum_{m=0}^{M} h_{km}(n^{k}x(n-m))$$

[0034] An equivalent expression of the Generalized Memory Polynomial with cross-products, can be expressed as follows:

$$y(n) = \sum_{m=0}^{M} x(n-m) \cdot f_{mp}(n-m)$$

[0035] where:

$$f_{mp}(m) = \sum_{k=0}^{K} h_{km}(n^{k}x(n-m))$$
where \( f(x) \) is a non-linear function having one or more user-specified parameters assumed to be accelerated in accordance with an aspect of the invention using the user-defined non-linear instruction \text{vec}_n, discussed below. It is noted that other basis functions other than \( x^k \) for non-linear decomposition are possible.

As discussed hereinafter, the user-defined non-linear instruction \( f_{\text{m},n} \) can be processed, for example, by a vector processor. The \( f_{\text{m},n} \) is an \text{mol} array of non-linear functions. Each non-linear function can have a user-specified parameter, such as a look-up table or coefficients. The look-up table can be a polynomial approximation of the user-defined non-linear instruction \( f_{\text{m},n} \). As discussed further below in conjunction with FIG. 7, the look-up table for each user-defined non-linear instruction \( f_{\text{m},n} \) in the \text{mol} array can be stored in memory and loaded into a register associated with a functional unit when the instruction is processed by the processor. The input samples can then be evaluated on the individual non-linear instruction \( f_{\text{m},n} \) in the \text{mol} array.

The exemplary pseudo code 400 comprises a first portion 310 to compute a magnitude of the input \( x \). In line 320, the look-up table for an individual non-linear instruction \( f_{\text{m},n} \) in the \text{mol} array can be loaded into a register. Thereafter, the exemplary pseudo code 330 comprises a portion 330 to implement equation (1) (e.g., input samples, perform a square operation on the samples, compute the non-linear function and then multiply accumulate the result).

FIG. 4A is a graphical illustration of an exemplary functional block diagram 400 that implements equation (1). In the exemplary embodiments described herein, \( |x|^2 \) is used instead of \( x^2 \). As shown in FIG. 4A, the exemplary circuit 400 comprises a plurality of delay elements, such as delay elements 405-1 through 405-5 to generate the \( x(n-m) \) term of equation (1) and delay elements 405-6 through 405-9 to generate the \( x(n-m)^2 \) term of equation (2) by delaying the output of a squaring operation 410. In addition, the exemplary functional block diagram 400 comprises an array of functional units 420-1,1 through 420-4,4 that receive the appropriate \( x(n-m)^k \) term and implements equation (2). The exemplary functional block diagram 400 also comprises a plurality of multipliers \( (x) \) that receive the appropriate \( x(n-m)^k \) term and multiply it with the output of the corresponding \( m,l \) functional unit 420. The outputs of the multiplication each row are added by adders \((+)+430\) and the outputs of each adder 430 in a given row are summed by a corresponding adder 440 to generate the output \( y(n) \).

FIG. 4B provides a graphical illustration 450 of an alternate exemplary functional block diagram 450 that implements equation (1) with a reduced number of multiply operations. As shown in FIG. 4B, the exemplary circuit 450 comprises a plurality of delay elements, such as delay elements 455-1 through 455-5 to generate the \( x(n-m) \) term of equation (1) and delay elements 455-6 through 455-9 to generate the \( x(n-m)^2 \) term of equation by delaying the output of a squaring operation 460. In addition, the exemplary functional block diagram 450 comprises an array of functional units 470-1,1 through 470-4,4 that receive the appropriate \( x(n-m)^k \) term and implements equation (2). Adders 480 compute the non-linear gains (sum of non-linear functions of magnitude of the input).

The exemplary functional block diagram 450 also comprises a plurality of multipliers \( (x) \) that receive the appropriate \( x(n-m) \) term and multiply it with the output of the squared output of a column of corresponding \( m,l \) functional units 470. In this manner, the non-linear gains from adders 480 are applied to the input data (complex multiply-accumulate (CMA) operations). The outputs of the multiplication added by adders \((+)+485\) to generate the output \( y(n) \).

FIG. 5A illustrates an individual user-defined non-linear function \( f_{\text{m},n} \) as a function of \( x(n) \). FIG. 5B illustrates an exemplary approximation 550 of the individual user-defined non-linear function \( f_{\text{m},n} \) of FIG. 5A. The exemplary approximation 550 of FIG. 5B uses segmented Taylor series look-up tables. The non-linear function \( f_{\text{m},n} \) is decomposed into \( j \) segments. The samples 560-1 through 560-\( j \) associated with each segment is stored in a look-up table. If a sample is stored in the look-up table for a given \( x \), the sample can be retrieved from the look-up table and directly employed in the non-linear function evaluation. If a desired \( x \) is between 2 values in the look-up table, then a linear interpolation or more generally a Taylor series-based interpolation is performed in hardware within the functional unit to obtain the result, as discussed further below in conjunction with FIG. 6. In this manner, the non-linear digital pre-distortion operation can be described by Taylor series coefficients in different segments of the input signal 550. In one exemplary implementation having 32 segments, for coefficients represented using 4 cubic polynomial approximations coefficients, in the look-up table there are 128 complex entries (16 bit complex and 16 bit real). In a further variation having 128 segments, and one coefficient per segment, there are 128 complex coefficients for linear interpolation (16 bit complex and 16 bit real). Alternatively, 32 complex entries for segments with 4 coefficients per segment for cubic interpolation.

As indicated above, if a desired \( x \) value is not in the look-up table but rather is between 2 values in the look-up table, then a linear interpolation is performed in hardware within the functional unit to obtain the result. A Taylor series computation can be performed as a cubic interpolation to evaluate the small cubic polynomial, as follows:

\[
f(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3\]

where the coefficients \( a \) are obtained from the look-up table. The complexity of this expression, however, is significant (with a number of multipliers to perform the multiplications and squaring operations).

The complexity can be reduced using the Homer algorithm (factorization), such that \( f(x) \) can be computed as follows. See, also, U.S. patent application Ser. No. 12/324, 934, filed Nov. 28, 2008, entitled "Digital Signal Processor With One or More Non-Linear Functions Using Factorized Polynomial Interpolation," incorporated by reference herein.

\[
f(x) = (b_3 x + b_2) x + b_1 x + b_0\] (3)

The complexity in equation (3) has been reduced to only 3 multiplication and 3 addition operations. \( f(x) \) is an offset from the value stored in the look-up table.

FIG. 6 illustrates a Taylor sum computation block 600 that implements equation (3). The coefficients \( b_0, b_1, b_2, b_3 \) are retrieved from the look-up table 650. The Taylor sum computation block 600 implements equation (3) with only 3 multiplication (610) operations and 3 addition (620) operations.

FIG. 7 is a schematic block diagram of an exemplary vector-based digital processor 700 that evaluates a user-de-
defined non-linear function for one or more complex numbers simultaneously in accordance with an embodiment of the present invention. Generally, the vector-based implementation of FIG. 7 performs different processes concurrently. Thus, the vector-based digital processor 700 contains plural functional units 710-1 through 710-N for evaluating user-defined non-linear functions.

Generally, the vector-based digital processor 700 processes a vector of inputs \( x \) and generates a vector of outputs, \( y(n) \). The exemplary vector-based digital processor 700 is shown for a 16-way vector processor nl instruction implemented as:

\[ \text{vec.nl } (x_1, x_2, \ldots, x_{16}), \text{range of } x[k] \text{ from } 0 \text{ to } 1 \]

In this manner, the vector-based digital processor 700 can perform 16 such non-linear operations and linearly combine them in a single cycle. For example, the user-defined non-linear function can be expressed as:

\[ f(x) = \sum_{k=0}^{15} a_k x^k \]

It is noted that in the more general case, different functions may be applied to each component of the vector data of the vector processor.

As shown in FIG. 7, the functional units 710 receive the user-specification, such as the look-up tables or coefficients, from memory for storage in a register.

CONCLUSION

While exemplary embodiments of the present invention have been described with respect to digital logic blocks and memory tables within a digital processor, as would be apparent to one skilled in the art, various functions may be implemented in the digital domain as processing steps in a software program, in hardware by circuit elements or state machines, or in combination of both software and hardware. Such software may be employed in, for example, a digital signal processor, application specific integrated circuit or micro-controller. Such hardware and software may be embodied within circuits implemented within an integrated circuit.

Thus, the functions of the present invention can be embodied in the form of methods and apparatuses for practicing those methods. One or more aspects of the present invention can be embodied in the form of program code, for example, whether stored in a storage medium, loaded into and/or executed by a machine, wherein, when the program code is loaded into and executed by a machine, such as a processor, the machine becomes an apparatus for practicing the invention. When implemented on a general-purpose processor, the program code segments combine with the processor to provide a device that operates analogously to specific logic circuits. The invention can also be implemented in one or more of an integrated circuit, a digital processor, a microprocessor, and a micro-controller.

It is to be understood that the embodiments and variations shown and described herein are merely illustrative of the principles of this invention and that various modifications may be implemented by those skilled in the art without departing from the scope and spirit of the invention.

We claim:

1. A method performed by a processor for implementing a signal processing function in software, comprising:
   obtaining at least one software instruction that performs at least one non-linear function for an input value, \( x \), wherein said at least one non-linear function comprises at least one user-specified parameter;
   in response to at least one of said software instructions for at least one non-linear function having at least one user-specified parameter, performing the following steps: invoking at least one functional unit that implements said at least one software instruction to apply said non-linear function to said input value, \( x \); and generating an output corresponding to said non-linear function for said input value, \( x \).

2. The method of claim 1, wherein said signal processing function comprises digital predistortion.

3. The method of claim 1, further comprising the step of loading said at least one user-specified parameter from memory into at least one register.

4. The method of claim 1, wherein said user-specified parameter comprises a look-up table storing values of said non-linear function for a finite number of input values.

5. The method of claim 4, wherein said user-specified parameter comprises one or more coefficients employed by said non-linear function to perform polynomial interpolation between entries of said look-up table.

6. The method of claim 1, wherein said processor comprises a vector processor.

7. The method of claim 1, wherein said input value, \( x \), comprises a vector and wherein said output comprises a vector.

8. A processor configured to implement a signal processing function in software, comprising:
   a memory; and
   at least one hardware device, coupled to the memory, operative to:
   obtain at least one software instruction that performs at least one non-linear function for an input value, \( x \), wherein said at least one non-linear function comprises at least one user-specified parameter;
   in response to at least one of said software instructions for at least one non-linear function having at least one user-specified parameter, performing the following:
   invoke at least one functional unit that implements said at least one software instruction to apply said non-linear function to said input value, \( x \); and
   generate an output corresponding to said non-linear function for said input value, \( x \).

9. The processor of claim 8, wherein said signal processing function comprises digital predistortion.

10. The processor of claim 8, wherein said at least one hardware device is further configured to load said at least one user-specified parameter from memory into at least one register.

11. The processor of claim 8, wherein said user-specified parameter comprises a look-up table storing values of said non-linear function for a finite number of input values.

12. The processor of claim 11, wherein said user-specified parameter comprises one or more coefficients employed by said non-linear function to perform polynomial interpolation between entries of said look-up table.

13. The processor of claim 8, wherein said processor comprises a vector processor.
14. The processor of claim 8, wherein said input value, x, comprises a vector and wherein said output comprises a vector.

15. A method performed by a processor for implementing digital pre-distortion in software, comprising:
   obtaining at least one software instruction that performs at least one non-linear function for an input value, x,
   wherein said at least one non-linear function comprises at least one user-specified parameter,
   in response to at least one of said software instructions for at least one non-linear function having at least one user-specified parameter, performing the following steps:
   invoking at least one functional unit that implements said at least one software instruction to apply said non-linear function to said input value, x; and
   generating an output corresponding to said non-linear function for said input value, x.

16. The method of claim 15, further comprising the step of loading said at least one user-specified parameter from memory into at least one register.

17. The method of claim 15, wherein said user-specified parameter comprises a look-up table storing values of said non-linear function for a finite number of input values.

18. The method of claim 17, wherein said user-specified parameter comprises one or more coefficients employed by said non-linear function to perform polynomial interpolation between entries of said look-up table.

19. The method of claim 15, wherein said processor comprises a vector processor.

20. The method of claim 15, wherein said input value, x, comprises a vector and wherein said output comprises a vector.