Implementación del algoritmo QRD-RLS sobre FPGA Aplicación a un Sistema Cancelador de Ruido

Miguel Enrique Iglesias Martínez
Departamento de Investigación y Desarrollo
Centro de Desarrollo de la Electrónica y la Automática, CDEA
Pinar del Río, Cuba
Email: mgi@cdea.co.cu

Abstract—**En este artículo se describe una de las tantas aplicaciones de los algoritmos de filtrado adaptativo, en este caso la reducción o cancelación de ruido en particular usando el algoritmo QRD-RLS, llevando a cabo su implementación sobre FPGA y tomando como eje fundamental del trabajo, la flexibilidad de estos dispositivos y las ventajas de la aceleración de algoritmos por hardware en determinadas aplicaciones.**

***Keywords*- FPGA,QRD-RLS, algoritmos**

I. INTRODUCCION

En algunos casos cuando se utilizan filtros digitales, las señales o los sistemas pueden sufrir algunos cambios con el tiempo, y la naturaleza exacta del cambio no es predecible; en tales casos es altamente deseable diseñar un filtro que pueda aprender del proceso mismo, de manera que se pueda adaptar para manejar la situación. Para resolver muchos de estos problemas se propone el uso de filtros adaptativos, cuya característica principal es que estos pueden modificar sus parámetros durante la operación con el fin de lograr un comportamiento deseado [3].

Los sistemas adaptativos en la actualidad han encontrado su lugar en muchas aplicaciones donde la capacidad de aprendizaje del sistema es un factor importante. Estas aplicaciones van desde el modelamiento de plantas con propiedades desconocidas, hasta el uso de estos sistemas en filtros capaces de cancelar el ruido [2].

Existen varios algoritmos para lograr el cálculo de los coeficientes en un sistema dado, los cuales varían en complejidad. Entre los más sencillos se encuentra el algoritmo Least Mean Square (LMS). Este algoritmo es muy usado debido a su facilidad de implementación y baja utilización de recursos computacionales.

Cuando el medio es altamente dinámico se requiere de algoritmos que se adapten rápidamente a los cambios, para estos casos el algoritmo LMS no brinda un buen desempeño. Con esos propósitos se crearon algoritmos de rápida respuesta, tales como el algoritmo RLS [2].

El cual su característica fundamental es que actualiza los coeficientes del filtro en cada iteración, basado en el lema de inversión de matrices, lo cual conlleva a un gran consumo de recursos computacionales siendo esta su mayor desventaja, imposibilitando su uso en aplicaciones de tiempo real en la mayoría de los sistemas basados en microcontroladores y DSP los cuales no cuentan con una arquitectura de procesamiento adecuada para ello.

El presente trabajo muestra la implementación del algoritmo RLS en su variante QRD-RLS sobre una arquitectura reconfigurable para lo cual se lleva a cabo primeramente el diseño del algoritmo en la herramientas de simulación de sistemas Matlab, para luego obtener su sintetización en código HDL mediante la plataforma de trabajo de Xilinx, AccelDSP, y su posterior inclusión en un sistema para la reducción de ruido.

El trabajo se ha organizado de la siguiente manera: La sección 2 explicará los conceptos básicos sobre algoritmos adaptivos en general, además del algoritmo QRD-RLS en particular. La sección 3 muestra la arquitectura y la plataforma de diseño. La sección 4 muestra los resultados obtenidos a partir del diseño planteado y su comparación con los obtenidos en simulación, y finalmente las secciones 5 y 6 recogen las conclusiones obtenidas de este trabajo y las referencias consultadas.

II. ALGORITMOS ADAPTATIVOS

Un sistema adaptativo puede modelarse según lo mostrado en la figura 1. Como se puede apreciar se tiene una planta con características definidas, cuya salida es ingresada al mecanismo adaptativo luego de ser restada de una señal deseada, con lo que dicho mecanismo puede calcular los coeficientes nuevos necesarios para adaptar la respuesta de la planta a la señal deseada.

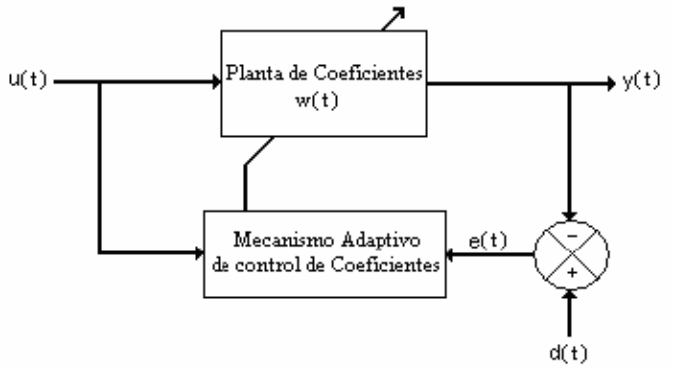

Figura 1 Diagrama de un sistema adaptativo

Las ecuaciones mostradas modelan el funcionamiento de un sistema adaptativo [1].

$$y(t) = u(t) * w(t) \qquad \textbf{(1)}$$
$$e(t) = d(t) - y(t) \qquad \textbf{(2)}$$

Donde ***u(t)*** es la señal de entrada, ***y(t)*** es la señal de salida ya filtrada, ***d(t)*** es la señal deseada a la salida y ***e(t)*** es el error entre ***d(t)*** e ***y(t)***. En este caso, la señal de entrada ***u(n)*** pasa a la Planta que contiene los coeficientes ***w(n)*** (un filtro FIR) y devuelve una señal ***y(t)*** cuyo resultado se muestra en (1). En el momento que la planta entrega el resultado ***y(t)***, esta resta a una señal ***d(t)*** y produce una señal de error ***e(t)*** cuyo resultado se muestra en la ecuación (2), la cual es el parámetro que le permite saber al mecanismo adaptivo que tan "lejos" se encuentra la planta de tener una respuesta similar a la señal deseada ***d(t)***. Conjuntamente con la señal ***u(t)*** se calculan los nuevos coeficientes ***w(t)*** para la planta usando un Mecanismo Adaptativo de Control de Coeficientes [1].

A. Algoritmo QRD-RLS

Dentro de los algoritmos adaptativos existentes el RLS (Recursive Least Square), es generalmente preferido por su rápida convergencia. El método basado en mínimos cuadrados intenta encontrar un juego de coeficientes que minimicen la suma del error cuadrático.

El cálculo directo del nuevo vector de coeficientes involucra la inversión de matrices, lo cual es generalmente indeseado en las implementaciones de hardware debido al alto consumo de recursos. La descomposición de matrices basada en esquemas de mínimos cuadrados tales como, SVD (Singular Value Descomposition) y QR evitan explícitamente la inversión de matrices, son más robustas y más asequible su implementación en hardware [5].

El método de descomposición QR comienza desde la matriz de datos usando transformación unitaria. Para una matriz Q(n) unitaria dada, la función de coste puede ser expresada como:

$$E(n) = \| Q(n)\mathrm{A}^{1/2}(n)e(n) \|^2$$
$$E(n) = \| Q(n)\mathrm{A}^{1/2}(n)d(n) - Q(n)\mathrm{A}^{1/2}(n)\mathrm{A}(n)w(n) \|^2 \qquad \textbf{(3)}$$

Para el problema de minimización definido por la función de coste, la matriz unitaria Q(n) es elegida para triangular la matriz de datos de manera exponencial ponderada tal que:

$$Q(n)\mathrm{A}^{1/2}(n)\mathrm{A}(n) = \begin{bmatrix} R(n) \\ 0 \end{bmatrix} \qquad \textbf{(4)}$$

Donde R(n) es una matriz triangular superior de dimensión KxK y 0 es una matriz nula de dimensión (n-K)xK. El vector de señal deseado, después de estar transformado, esta definido por:

$$Q(n)\mathrm{A}^{1/2}(n)d(n) = \begin{bmatrix} p(n) \\ v(n) \end{bmatrix} \qquad \textbf{(5)}$$

Donde p(n) es un vector de Kx1 elementos y v(n) es un vector de (n-K) x1 elementos, luego se puede reescribir la función de coste de la siguiente manera:

$$E(n) = \| \begin{bmatrix} p(n) \\ v(n) \end{bmatrix} - \begin{bmatrix} R(n) \\ 0 \end{bmatrix} w(n) \|^2$$

$$= \| \begin{bmatrix} p(n) - R(n)w(n) \\ v(n) \end{bmatrix} \|^2 \qquad \textbf{(6)}$$

Es obvio que la estimación de mínimos cuadrados para el vector de pesos debe satisfacer que:

$$p(n) - R(n)w'(n) = 0_K \, x1$$
$$w'(n) = R^{-1}(n)p(n) \qquad \textbf{(7)}$$

La matriz unitaria Q(n), la matriz triangular superior R(n), y el vector p(n) , pueden ser calculados recursivamente usando:

$$\begin{bmatrix} R(n) & p(n) \\ 0(n-K-1)xK & 0(n-K-1)x1 \\ 0_1 \, xK & \alpha(n) \end{bmatrix}$$

$$= Q^{'}(n)\begin{bmatrix} \lambda^{1/2}R(n-1) & \lambda^{1/2}p(n-1) \\ 0(n-K-1)xK & 0(n-K-1)x1 \\ u^{T}(n) & d(n) \end{bmatrix}$$

$$Q(n) = Q^{'}(n)\begin{bmatrix} Q(n-1) & 0 \\ 0 & 1 \end{bmatrix} \quad \textbf{(8)}$$

Por lo tanto el vector de coeficientes óptimos puede ser obtenido. Pero en algunas aplicaciones tales como reducción de ruido y predicción linear e(n), es la señal de salida. Desarrollando la ecuación anterior podemos obtener e(n) directamente sin extraer el vector de pesos explícitamente. Este resultado puede resumirse en la siguiente ecuación:

$$\begin{bmatrix} R(n) & p(n) & R^{-1}(n)u(n) \\ 0_{1xK} & \alpha(n) & (\gamma^{1/2}(n))* \end{bmatrix} \quad \textbf{(9)}$$

$$= Q''(n)\begin{bmatrix} \lambda^{1/2}R(n-1) & \lambda^{1/2}p(n-1) & 0_{Kx1} \\ u^{T}(n) & d(n) & 1 \end{bmatrix} \quad \textbf{(10)}$$

La descomposición QR se puede realizar mediante la utilización de rotaciones de Givens [6], encargadas de diagonalizar la matriz cada vez que un nuevo dato entra, con un coste computacional muy bajo.

III. ARQUITECTURA Y PLATAFORMA DE DISEÑO

A partir de lo mencionado anteriormente se inicia el diseño y simulación del algoritmo QRD-RLS según los parámetros de señal para los cuales el sistema deberá responder de acuerdo a la características de ruido presente en la señal útil, para lo cual se utiliza como señal interferente ruido blanco gaussiano de valor medio cero y varianza constante, y como señal útil de prueba, un tono sinusoidal de frecuencia 1KHz muestreado a 16 KHz. Como se muestra en la figura 2.

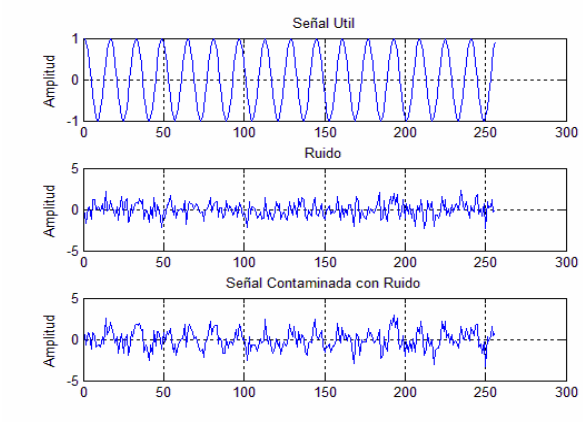

Figura 2 Señales utilizadas en la Aplicación

El diagrama de la arquitectura propuesta se observa a continuación en la figura 3 donde se puede apreciar la forma clásica de un sistema adaptativo utilizado para cancelar ruido, donde en este caso, se toma como referencia del sistema ,la señal contaminada(***d(n)+N(n)***), y como señal de entrada al filtro, una muestra de ruido correlacionado(***N'(n)***) con el presente en la señal, obteniendo como resultado a la salida del filtro una réplica del ruido contaminante de la señal útil , lo que al efectuar la diferencia entre estas dos señales , se obtiene como salida la señal útil inicial, correspondiente al error entre la referencia y la salida del filtro. La diferencia de este sistema con respecto a las demás arquitecturas de sistemas adaptativos es que la salida del mismo es la señal de error [2].

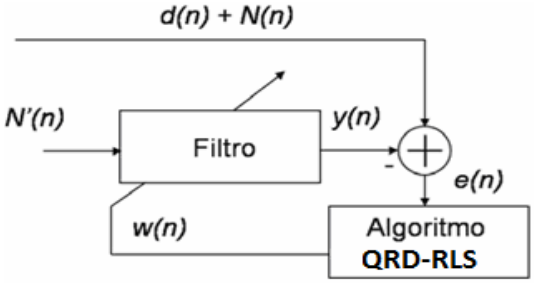

Figura 3 Filtro Adaptativo como reductor de ruido

La elección de la estructura de cancelación adaptativa mostrada en la figura 3 y usada para la verificación del algoritmo QRD-RLS pudiera no ser la mas óptima en cuanto a la cantidad de información que necesita el sistema para adaptarse, pudiendo usarse otro tipo de estructura de cancelación de ruido, como la predictiva, la cual no requiere el conocimiento previo de la naturaleza del ruido aditivo a la señal, ni tampoco que el mismo deba estar correlacionado. Sin embargo, esto puede implicar un aumento en el consumo de recursos hardware del algoritmo QRD-RLS imposibilitando la fiabilidad de su sinterización sobre una arquitectura reconfigurable.

Que el ruido que se toma como muestra de entrada al filtro, este correlacionado con el incidente en la señal útil, implica mejores resultados y un menor número de operaciones para la convergencia del algoritmo, lo cual se traduce en un ahorro de recursos computacionales, siendo este último punto, y la síntesis del algoritmo QRD-RLS, los objetivos fundamentales del trabajo.

A. AccelDSP

AccelDSP es una herramienta de síntesis proporcionada por Xilinx, la cual permite transformar un diseño en punto flotante desarrollado en Matlab, en un módulo hardware, que puede ser implementado en un FPGA.

Posee una interfaz de usuario fácil de usar que controla un ambiente integrado con otras herramientas de diseño, tales como: Matlab, Xilinx ISE, System Generator y otras como simuladores de código HDL y sintetizadores lógicos [4].

B. *Capacidades de AccelDSP*

AccelDSP proporciona las siguientes capacidades:

- Lee y analiza un diseño en punto flotante desarrollado en Matlab.
- Crea automáticamente el diseño equivalente en punto fijo.
- Invoca a Matlab para simular y verificar el diseño tanto en punto fijo como en punto flotante.
- Crea un modelo HDL sintetizable con su fichero de simulación correspondiente (Testbench) [4].

C. *Flujo de diseño AccelDSP*

En la figura 4 se puede observar el flujo de diseño de AccelDSP cuando se utiliza Xilinx System Generator como herramienta de diseño, para adicionar el sistema generado como una librería más y reutilizar esta en posteriores diseños, aunque se puede emplear también Xilinx ISE .

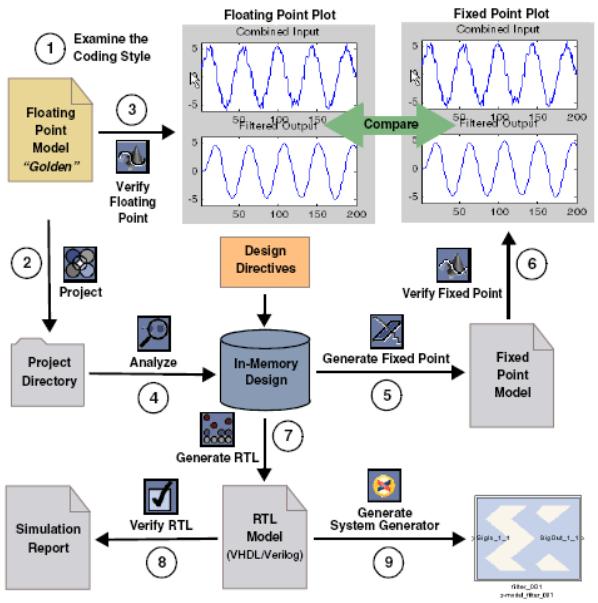

Figura 4 Flujo de Diseño AccelDSP-XSG [4].

Una vez generado el bloque del algoritmo QRD-RLS a través de la herramienta de síntesis AccelDSP y exportado a System Generator, se adiciona este a un sistema diseñado para comprobar su funcionamiento. El sistema de comprobación del algoritmo se diseñó en System Generator, mediante el cual se generan, tanto la señal útil correspondiente al tono sinusoidal, como el ruido blanco gaussiano. Posteriormente la señal obtenida pasa a través de un conversor DAC para ser visualizada y comparada con los resultados obtenidos en la simulación.

Todo el sistema fue implementado sobre la arquitectura FPGA SPARTAN-3AN, la cual posee 700K compuertas y 20 multiplicadores embebidos, de los cuales solo se usa el 60 % de los mismo. Teniendo en cuenta que este es un punto crítico a la hora de implementar el algoritmo, es este un buen resultado con respecto al consumo de recursos del sistema. La figura 5 muestra el diagrama completo del sistema.

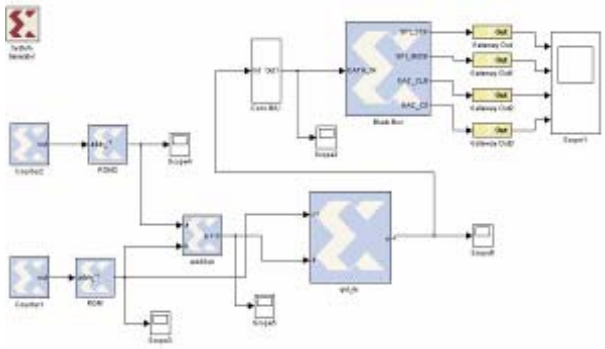

Figura 5 Arquitectura del Sistema en System Generator

IV. RESULTADOS

En orden de comparar los resultados obtenidos en la simulación del sistema cancelador de ruido, con los mostrados en la implementación del algoritmo QRD-RLS, se presentan a continuación una serie de gráficas para la comparación de los mismos, la cuales detallan la eficiencia del algoritmo en tareas de reducción o cancelación de ruido, utilizando como secuencia de adaptación, una señal de referencia.

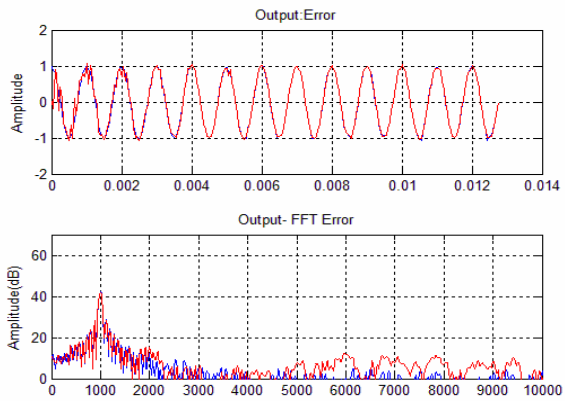

Figura 6 Salida del sistema y comportamiento de frecuencias

Como se puede apreciar en la figura anterior la salida del sistema generada en el modelo de punto flotante

descrito en Matlab y la salida del modelo punto fijo generado por la herramienta coinciden, así mismo el comportamiento de frecuencias, lo cual demuestra la convergencia de los coeficientes del filtro a la atenuación de las frecuencias fuera del rango dinámico de señal útil.

A continuación se puede ver la comparación entre la señal de entrada contaminada y salida del sistema, con lo cual se muestra la coincidencia del modelo de simulación en Matlab con el modelo real implementado sobre el FPGA.

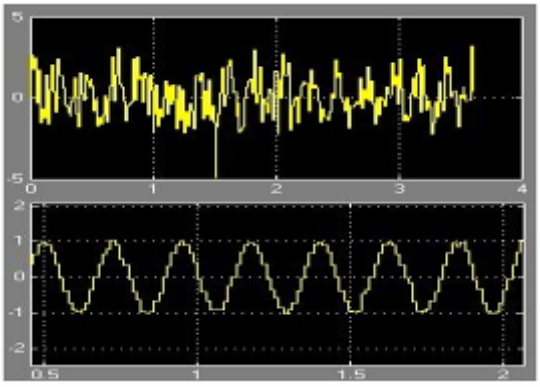

Figura 7 Entrada contaminada y señal a la salida del sistema en System Generator

En cuanto a los recursos utilizados en la implementación del algoritmo QRD-RLS, la tabla 1 muestra el consumo que originó la implementación del mismo. Cabe destacar que el procesamiento de los datos mediante el algoritmo QRD-RLS se hizo tomando como longitud máxima, 24 bits, forzando el diseño del mismo a esta longitud de datos, teniendo en cuenta los recursos del dispositivo con el cual se llevó a cabo la implementación, y sin comprometer la eficiencia del algoritmo en cuanto a presentar a su salida, un resultado óptimo.

Device Utilization Summary (xc3s700an-4fgg484)			
Logic Utilization	Used	Available	Utilization
Number of Slices	2987	5888	50%
Number of Slice Flip Flops	789	11776	6%
Number of 4 input LUTs	5819	11776	49%
Number of bonded IOBs	57	372	15%
Number of MULT18X18SIOs	12	20	60%
Number of GCLKs	1	24	4%

Tabla 1 Consumo de Recursos del Algoritmo QRD-RLS

CONCLUSIONES

El algoritmo QRD-RLS es una excelente manera de filtrar una señal cualquiera usando una señal de referencia ***d(t)*** como modelo, pero, lamentablemente, su implementación se traduce en un costo computacional elevado, aunque se pueden llegar a diseños como el mostrado aquí, donde se prefije la longitud de los datos sin comprometer los resultados esperados, lo que puede representar un ahorro de recursos y área de trabajo , si este representa un punto clave en la implementación, ya que los mayores problemas de este algoritmo, son los cálculos de la matriz de autocorrelación inversa de la señal de entrada ***P(n)***, la cual es de NxN, y del vector de ganancia de Kalman ***K(n).*** Es importante notar que dado que las operaciones involucradas son operaciones entre vectores y matrices, significa que existe un gran consumo de elementos dedicados, en este caso multiplicadores como caso crítico en el diseño. No obstante se logró sintetizar el algoritmo QRD-RLS sobre una arquitectura paralela y reconfigurable lo que posibilita su reutilización en posteriores diseños.

REFERENCIAS

[1] Jorge Benavides Aspiazu, Walter Calienes Bartra and Carlos Silva Cárdenas,"Diseño de una arquitectura para la implementacion de un filtro adaptativo rls sobre un fpga", XV Workshop Iberchip, Buenos Aires - Argentina,Marzo 2009.

[2] Saeed V. Vaseghi, Advanced Digital Signal Processing and Noise Reduction, Third Edition, 2006, John Wiley & Sons Ltd.

[3] S. Haykin,"Adaptive Filter Theory", 3rd ed. Upper Saddle River, NJ:Prentice-Hall, 1994.

[4] AccelDSP Synthesis Tool User Guide obtenido de: http://www.xilinx.com/acceldsp_user.pdf.

[5] Implementation of CORDIC-Based QRD-RLS Algorithm on Altera Stratix FPGA with Embedded Nios Soft Processor Technology obtenido de: www.altera.com/literature/wp/wp_qrd.pdf

[6] B. Yang and J. F. Bohme, "Rotation-based RLS algorithms: Unified derivations, numerical properties, and parallel implementations," *IEEE Trans. Signal Processing*, vol. 40, pp. 1151–1167, May 1992.