# Generador de señales con forma de onda arbitraria y ruido usando DDS en FPGA

Trabes, Emanuel; Costa, Diego Esteban; Sosa Páez, Carlos Federico (*)
Laboratorio de Electrónica, Investigación y Servicios
Facultad de Ciencias Físico, Matemáticas y Naturales / Universidad Nacional de San Luis
San Luis, Argentina
e-mail: {etrabes, dec, sosapaez}@unsl.edu.ar

**Resumen—El trabajo consiste en la realización de un generador de señales implementado en FPGA como instrumento virtual que utiliza una computadora como interfaz de usuario. Permite generar ruido blanco y formas de onda arbitrarias ingresadas por el usuario mediante un archivo. El diseño está basado en un Direct Digital Synthesizer (DDS). La descripción del hardware se realizó enteramente en VHDL, y se utilizó WISHBONE como interconexión de los módulos, de manera que los mismos sean fácilmente reutilizables. El software que se ejecuta en la PC se realizó con el ambiente de desarrollo Qt. Se obtuvo una alternativa de bajo costo a los generadores comerciales logrando similar funcionalidad y comportamiento dinámico, con una interfaz de usuario amigable y flexible, y la posibilidad de utilizar la PC como fuente secundaria de procesamiento para aplicaciones que requieran una potencia de cálculo elevada pero no necesiten ser ejecutados en tiempo real (adquisición y procesamiento de señales para usar como fuente de forma de onda arbitraria, presentación gráfica y cómputo de parámetros temporales, espectrales, estadísticos, etc.).**

***Palabras clave —Generadores de señales, instrumentación virtual, DDS, VHDL, FPGA, Qt, WISHBONE.***

## I. INTRODUCCIÓN

Para probar equipos electrónicos se necesita instrumental cada vez más complejo y sofisticado. Adquirir instrumental de última generación puede ser considerablemente caro. Utilizar una PC como alternativa no siempre es factible ya que en muchas aplicaciones se requiere velocidad de procesamiento en tiempo real que sólo se consigue con hardware específico. La tecnología FPGA [1] provee el hardware reconfigurable necesario para generar diseños a la medida de las aplicaciones requeridas, y además, permite actualizar el diseño o configurarlo para que realice tareas completamente nuevas, mediante simples actualizaciones del hardware programado.

Este trabajo consiste en un generador de señales realizado como instrumento virtual [2] [3] [4]. La FPGA se encarga de las tareas en tiempo real y la PC se utiliza como panel de control para el ajuste de los parámetros. Eventualmente, la PC puede ejecutar programas complementarios que requieran grandes cantidades de procesamiento. De esta manera se obtiene lo mejor de ambas tecnologías, la posibilidad de hardware específico y a medida en la FPGA, y la robustez gráfica y gran poder computacional de una PC [5] [6] [7] [8].

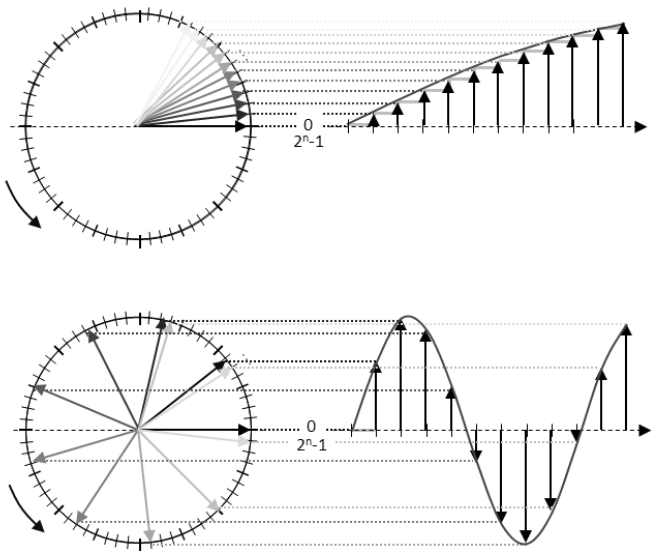


Figura 1. Rueda de fase y señal resultante para diferentes saltos de fase.

La implementación se basó en un Sintetizador Digital Directo, DDS (*Direct Digital Synthetizer*) [9] [10]. A partir de una frecuencia de reloj fija, estos generadores producen señales de frecuencia ajustable. En general, los DDS poseen un registro de fase, un acumulador de fase, una LUT (*"look-up" table*) y un conversor digital/analógico DAC (*digital to analogue converter*). El registro de salto de fase almacena la palabra de sintonía o control, la cual define el salto que da el acumulador de fase en cada ciclo de reloj. El acumulador de fase es un contador que computa la dirección a ser leída en la LUT. La LUT transforma la fase en amplitud, ya que, para cada una de las direcciones dadas por el acumulador, provee a la salida el valor digital de amplitud de la señal que tiene guardada, en correspondencia con cada valor de fase. Este valor digital de amplitud es convertido en el valor analógico correspondiente de tensión en el DAC. Cuando la palabra de control no cambia, se genera una señal de frecuencia fija. Como se ve en la Figura 1, si el incremento de fase es pequeño, el acumulador de fase recorre la LUT lentamente y tarda más en alcanzar el máximo, generando así una frecuencia más pequeña. Si el incremento de fase es grande, el acumulador de fase recorre rápidamente la LUT y por consiguiente, genera una señal de frecuencia alta.

(*) Pertenecientes al Proyecto "Instrumentación Virtual Reconfigurable" financiado por Ciencia y Técnica de la UNSL.

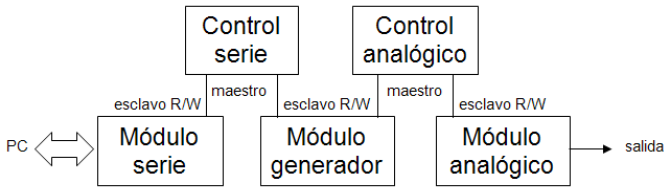


Figura 2. Módulos que componen el sistema diseñado.

En este trabajo se agregó un bloque para generar señales rectangulares y de diente de sierra con gran sencillez, y además, una memoria RAM, que funciona en paralelo con la LUT para generar ondas de forma arbitraria, ingresadas por el usuario mediante una imagen en formato digital. También se incorporó un generador de ruido blanco, función necesaria en muchas tareas y disponible en varios generadores comerciales.

## II. FPGA

En la implementación se utilizó la placa de desarrollo Fusion de Actel [11] con sus herramientas de diseño [12] [13]. La utilización de esta placa es meramente ilustrativa, ya que el diseño está descrito enteramente en VHDL [14] [15] y se puede portar a otras placas fácilmente. El diseño modular hace posible la rápida modificación de las resoluciones de cada etapa si fuera necesario. Para la salida analógica se utilizó la LP Data Conversion Daughter Board desarrollada por el ICTP[1]. Esta placa de expansión posee el DAC LTC1654 con resolución de 14 bits serie y rango dinámico de $3{,}3V_{pp}$.

La Figura 2 muestra el diagrama de los módulos que componen el sistema. Todos los módulos comparten un reloj de 20MHz generado con la primitiva de un PLL a partir del oscilador disponible en la placa de 50MHz. A continuación se explica el funcionamiento de cada módulo.

### *A. Módulo generador*

El módulo generador está compuesto por varios bloques.

El bloque NCO ("numerically controled oscilator") es un oscilador controlado digitalmente que constituye el corazón del módulo generador. Está basado en un *IP core* de un DDS [16] [17]. En la Figura 3 se presenta el diagrama funcional.

En cada flanco ascendente del reloj, cuando la señal de habilitación está en alto, la fase en el acumulador de fase es incrementada en el valor dado por la palabra de sintonía M guardada en el registro de salto de fase. La palabra de sintonía y por ende, el registro de salto de fase, se definió con un ancho de 32 bits. Como es habitual en los DDS y para utilizar el código del *IP core* sin modificaciones, la fase es truncada a un ancho restringido p, en este caso de 12 bits. Esta fase truncada es utilizada como dirección de la LUT que convierte la fase en amplitud de señal. Fue implementada con una primitiva de una ROM definida como un arreglo de 4096x12 bits.

La ecuación de sintonía calcula la frecuencia Fout de la señal de salida controlada por el incremento de fase M y la frecuencia del reloj $F_s$

$$Fout = M\frac{Fs}{2^n} \qquad (1)$$

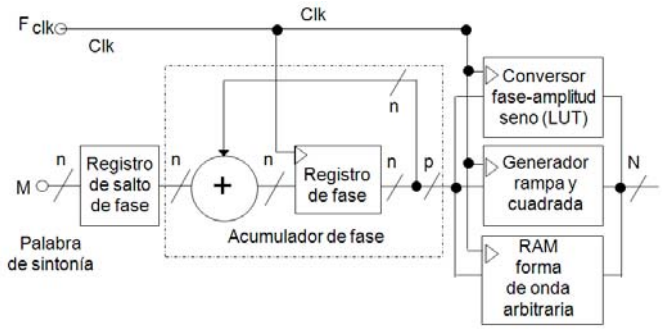


Figura 3. Diagrama de bloques del módulo generador.

En base a lo anterior pueden calcularse las frecuencias máximas y mínimas teóricas

$$Fmin = \frac{Fs}{2^{32}} \quad ; \quad Fmax = \frac{Fs}{2} \qquad (2)$$

Con esto, sin tener en cuenta las limitaciones del DAC, para una frecuencia de reloj de 20MHz y un ancho de palabra de fase n de 32 bits, la resolución del generador realizado puede estimarse en 0,00465Hz y la frecuencia máxima en 10MHz.

En este trabajo, la señal diente de sierra se obtiene directamente de la salida del acumulador de fase. La señal cuadrada se genera tomando el bit más significativo de la señal senoidal. A esta implementación se le agregó una RAM, instanciada como una primitiva, la cual trabaja en paralelo con la ROM que contiene la tabla de valores de la salida senoidal. Con una funcionalidad similar a la ROM, la RAM sirve al usuario para almacenar la información de la señal arbitraria.

Este módulo posee también un sumador, para agregar un offset a la señal básica generada por las tablas. Este offset puede tener un valor máximo de 0x3FFF.

También posee un multiplicador, para generar la señal con la amplitud que se requiera. El multiplicador opera sobre un dato de ancho N igual a 10 bits, los más significativos del dato proveniente de la RAM/ROM, y un operando de 4 bits, que selecciona el usuario, por lo que se obtienen 16 niveles de amplitudes de salida posibles para la implementación presente.

La definición de los anchos de palabra se hizo en base a una relación de compromiso entre las diferentes características, la limitación de 14 bits del DAC, los recursos de memoria, la compatibilidad entre bloques, etc.

Para generar la señal de salida, cada etapa realiza un proceso simple siguiendo una estructura de tubería. Primeramente se lee el dato de la tabla de valores, ya sea de la RAM o de la ROM. Luego, a esta señal se la multiplica por la amplitud deseada. En el paso siguiente, se le suma el offset. Este dato es colocado a la salida por la interfaz correspondiente.

[1] Esta placa fue provista por el ICTP (Internacional Center for Teorethical Physics) y fue desarrollada por A. Cicuttin, L. Crespo y A. Shapiro.

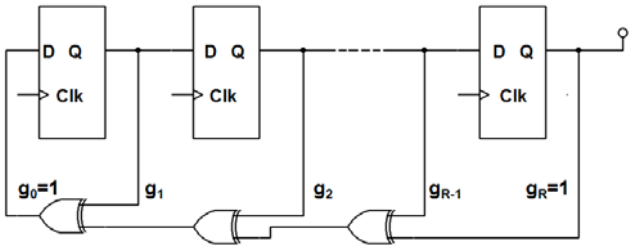


Figura 4. Generador de ruido blanco en configuración Fibonacci.

Para la generación del ruido blanco, se implementó un registro de desplazamiento lineal realimentado LFSR (*linear feedback shift register)* [18]. El valor guardado en este registro de desplazamiento se modifica a sí mismo en cada flanco ascendente del reloj. Gracias a la realimentación y a las compuertas XOR colocadas a la salida de algunos de los registros, el valor rota a través de un conjunto de valores únicos. La secuencia particular que se genera depende del lugar donde se coloquen las compuertas XOR. La ecuación (3) expresa la cantidad L de valores pseudo aleatorios generados

$$L=2^R-1 \quad (3)$$

donde R es la cantidad de registros que contiene el módulo.

Se optó por una configuración Fibonacci que es más simple para implementarla en un lenguaje de descripción de hardware. La misma posee las compuertas XOR entre la salida de algunos registros no consecutivos y es realimentada a la entrada del primer registro, como se observa en la Figura 4. Se usó un *IP core* [19] descrito enteramente en VHDL con una longitud de 14 registros acorde a la longitud de palabra del DAC, lo que da una secuencia de 16383 valores pseudo aleatorios.

### *B. Módulo serie*

Este módulo implementa un transmisor y receptor asíncrono universal UART (*Universal Asynchronous Receiver/Transmitter*) para comunicación serie tipo RS232. Se usó una primitiva de la biblioteca de Actel llamado "coreUART" encapsulado para tener la compatibilidad con WISHBONE [20]. La placa de desarrollo usada posee un puente USB-RS232, por lo que se optó por realizar, tanto del lado de la FPGA como del lado de la PC, una comunicación serie RS232 pero utilizando físicamente el puerto USB. Esto permitió reducir el tiempo de desarrollo usando herramientas conocidas y simples a nivel de software y hardware, pero implementado sobre un puerto moderno y ampliamente disponible como es el USB.

La comunicación WISHBONE soportada es la comunicación estándar monociclo, con la utilización de la señal WISHBONE no obligatoria RTY_I de aviso de error y pedido de re-intento de comunicación por parte del esclavo al maestro. Esto permite independizar la velocidad del diseño general de la velocidad de la comunicación serie. La frecuencia de comunicación se ajustó en 57600 bps. El módulo "coreUART" se configuró para no utilizar bits de paridad y tener una palabra de 8 bits de longitud.

### *C. Control serie*

El control serie es una máquina de estado que, ante una señal de reset, pone todos sus registros internos en 0 y vuelve al estado INI. En el estado INST_FETCH consulta al módulo serie si tiene una instrucción nueva hasta obtener una respuesta afirmativa, para decodificar la instrucción leída en el estado INST_DECODE. Luego de que cualquiera de los estados inferiores detallados en la Tabla 1 completa su tarea, la máquina de estado vuelve a INST_FETCH, para buscar la próxima instrucción a ejecutar. Se desarrolló un protocolo simple donde el código de instrucción está codificado en los 3 MSB del primer byte leído y los datos pueden estar en los 5 bits restantes del byte o en los bytes siguientes. No se implementó ninguna detección de errores porque en las pruebas realizadas nunca hubo problemas en la comunicación.

TABLA 1. CODIFICACIÓN DE LAS INSTRUCCIONES DEL CONTROL SERIE

| Código | Estado | Acción |
|---|---|---|
| 100 | SEL_WAVE | selección de la forma de onda |
| 110 | SEL_FREC | elección de frecuencia de la señal |
| 010 | WRITE_RAM | escritura en memoria RAM |
| 001 | SET_OFFSET | elección del OFFSET de la señal |
| 011 | SET_AMP | elección de la amplitud de la señal |
| 101 | READ_RAM | lectura en memoria RAM |

### *D. Control analógico*

Es el encargado de leer los datos del generador y enviarlos adecuadamente al módulo analógico, de manera que la señal pueda observarse a la salida del DAC. Posee una máquina de estado que verifica si el módulo analógico está listo para recibir un nuevo dato y en caso afirmativo le envía un dato de inicialización para configurar el DAC en modo *slow*, que tiene tiempos de levantamiento y asentamiento lentos pero posee características dinámicas mejoradas respecto al modo *fast* existente en el LTC1654. Luego lee un nuevo dato del módulo generador y se lo envía al módulo analógico para luego empezar nuevamente el bucle. Está diseñado para utilizar un bus WISHBONE y trabajar como maestro en este bus.

### *E. Módulo analógico*

Se encarga de adaptar las tres señales de entrada que necesita el LTC1654 al estándar WISHBONE. Es una máquina de estado que toma el dato enviado por el módulo control analógico. De la cadena de 24 bits enviadas por la salida serie SDI, los primeros 4 bits de control indican la acción a realizar como la configuración y el almacenamiento de un nuevo dato en un registro para generar la salida analógica. Los siguientes 4 bits de dirección indican si el DAC usa la salida A, B o ambas. Los 16 bits siguientes representan el dato a ser convertido a niveles de tensión analógica. Dado que los 2 bits menos significativos son ignorados por el LTC1654, se optó por acomodar el dato sin perder bits de la señal básica, a costa de sacrificar rango de amplitud y offset, cumpliendo los requerimientos del estándar WISHBONE con una descripción genérica para ser re-utilizada si se reemplaza el DAC.

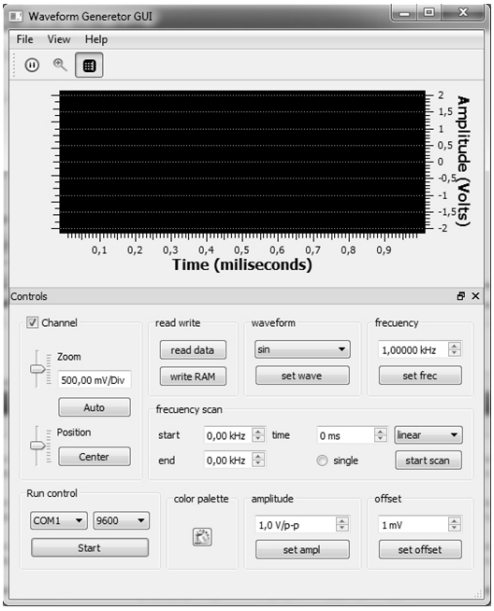


Figura 5. Interfaz gráfica de usuario programada en C++ sobre Qt.

## III. SOFTWARE

### A. *Interfaz gráfica de usuario*

La interfaz gráfica de usuario GUI (*Graphic User Interface*) que se ejecuta en la PC fue programada en el lenguaje C++ utilizando el ambiente de desarrollo Qt [21]. Este es un ambiente multiplataforma, con unas potentes librerías gráficas [22] usado en múltiples aplicaciones de instrumentación virtual [23] [24]. Para facilitar la reutilización del código, éste fue diseñado por capas, cada una con una función clara y específica según se detalla a continuación.

#### 1) *Capa "PUERTO"*

Esta capa implementa la comunicación de más bajo nivel. Tiene funciones para la comunicación básica en un entorno Windows con el puerto serie, como leer byte, escribir byte, abrir el puerto y cerrar el puerto.

#### 2) *Capa "COMTHREAD"*

Esta capa implementa un hilo de ejecución o *thread*. Se encarga de enviar datos correctamente a la placa de desarrollo, como también de recibirlos. Básicamente, esta capa intercala escritura con lectura del puerto, de a un dato por vez. Para la lectura guarda los datos en un buffer. Cuando está lleno, esta capa le da la señal a la capa superior para proseguir con la lectura de los mismos. Cuando los datos son leídos por la capa superior, el buffer se borra, comenzando con la captura de datos nuevamente. Para la escritura, envía los datos por el puerto uno a uno hasta que todos hayan sido enviados.

#### 3) *Capa "BOARDTOOLS"*

Esta capa se encarga de tomar las selecciones que realiza el usuario en el entorno gráfico, y pasárselas a la clase anterior para enviar la instrucción necesaria al hardware. Implementa los controles para el usuario, y se encarga de acomodar los datos, tanto para mostrarlos por la interfaz gráfica, como para su correcto envío por el puerto hacia la FPGA.

#### 4) *Capa de Usuario*

Esta capa crea la ventana principal de la aplicación, instancia la clase anterior, crea el gráfico, los botones de cerrado del programa, envía mensajes de error o notificación hacia el usuario y realiza todas las tareas para la correcta representación de la ventana. En la Figura 5 puede observarse el aspecto de la ventana creada en este trabajo.

### B. *Procesamiento de la imagen*

Para ejemplificar las ventajas de usar una PC, se realizó un programa que procesa una imagen y extrae de la misma la información para generar una señal arbitraria. La imagen puede estar guardada en una gran variedad de formatos con cualquier resolución. Puede provenir de una fotografía, una captura de pantalla o alguna aplicación en software. Se hicieron pruebas con dibujos realizados sobre papel con excelentes resultados como se muestra en la Figura 6. Para un análisis correcto, el trazo debe ser oscuro sobre fondo blanco. Si las condiciones de iluminación son óptimas, no es necesario tener alto contraste. El programa lee la imagen de un archivo y la escala automáticamente. Luego la transforma a escala de grises. Después se itera en el eje "x" de la imagen, buscando en cada valor de "x" el menor valor posible de "y". Estos representan los 4096 valores de la amplitud de la señal que son guardados en la RAM. Las muestras son normalizadas de manera que el mayor valor de amplitud corresponda a "0x3FF" y que el valor más pequeño corresponda con 0x000. Posteriormente, los datos son guardados en un archivo ".dat" para ser utilizado por el programa principal que lee los datos y los envía uno a uno al hardware para almacenarlo en la RAM.

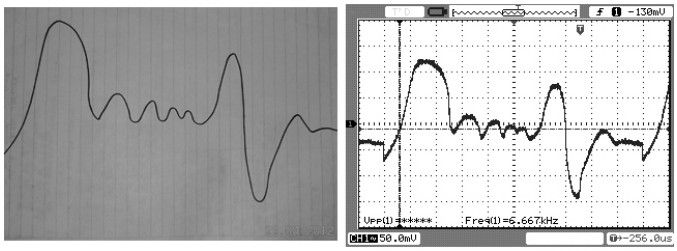


Figura 6.Trazo de forma de onda arbitraria dibujado sobre papel rayado y señal generada mediante la herramienta de procesamiento de imagen.

## IV. RESULTADOS OBTENIDOS

Como criterio de diseño, se buscó independizar la implementación en la FPGA de las prestaciones del DAC que posee ciertas limitaciones, y a su vez, se trató de mantener la compatibilidad con otros trabajos y el estándar WISHBONE.

La distorsión por el error de cuantización de amplitud, es consecuencia de la resolución finita del DAC y se puede cuantificar mediante la relación de potencia de señal a potencia de ruido de cuantización SQNR dada por

$$SQNR_{[dB]} = -1.76 - 6.02B \qquad (4)$$

siendo B la resolución del DAC expresada en bits. El DAC de 14 bits en el presente trabajo da una SQNR de -86,05dB.

Otra fuente de error es el truncamiento de fase previo a la conversión de fase a amplitud, que produce componentes espectrales espurias. La diferencia entre el nivel de la portadora y el máximo nivel de las señales espurias es llamado rango dinámico libre de espurios SFDR [25] [26]. Cuando el nivel de la portadora es 0 dB, el SFDR puede estimarse como

$$S_{máx[dB]} = -SFDR_{[dB]} = -6.02P + 3,92dB \quad (5)$$

siendo P la cantidad de bits de la fase luego de ser truncados. En este trabajo, como se dejan 12 bits para la fase, el valor puede estimarse en -68,32dB. Como la distorsión por cuantización de amplitud es pequeña gracias a la resolución del DAC, la distorsión espuria total puede aproximarse con la distorsión por cuantización de fase. Del análisis realizado con el Osciloscopio y Analizador de Espectro Tektronix TDS 2022C, de 200 MHz y 20 Gs/s, se observó una SFDR real menor a -40 dB en todo el rango de frecuencias, como se ve en la Figura 7 para el caso de 10 kHz. También se analizó la distorsión armónica total para una señal de 1 kHz con máxima amplitud arrojando 1,31% de THD hasta el armónico 12.

Respecto a otras fuentes de error, el DAC utilizado tiene una distorsión no lineal integral de 1,2 bits menos significativos y una diferencia de 0,3 bits menos significativos.

La máxima salida de tensión posible es 3,3V, ya que el DAC está alimentado con esa tensión. El módulo generador tiene un dato de salida de 16 bits de longitud. Esto permitiría 65536 valores diferentes de amplitud. Como el DAC es de 14 bits, la resolución efectivamente se ve reducida a 16384 valores. La resolución de amplitud requerida por el diseño es de 0,3V, por lo que se decidió utilizar solamente 10 de los 12 bits de amplitud generados por el NCO, de manera de obtener a la salida una señal mínima (sin offset y con un multiplicador configurado en 1) de $0,2V_{pp}$. Por esto, del NCO se descartan los 2 bits menos significativos. Entonces estos 14 bits están compuestos por 10 bits de la señal básica guardada en el NCO, que son 1024 valores desde 0 hasta 0,2V con pasos de 0,2mV. Para configurar la amplitud de la señal se utilizan 4 bits, dando un total de 16 valores de amplitud posibles con un paso de 0,2V. Para el offset se utilizan los 16 bits completos. El máximo offset está dado con el valor 0x3FFF, con el cual una señal de amplitud de 0 V tendría un offset de 3,3V utilizando todo el rango del DAC. Para evitar sobrepaso de la tensión máxima, los valores máximos de amplitud y offset que pueden ser aplicados simultáneamente son restringidos por el software.

Para analizar la linealidad en la amplitud se verificó el funcionamiento en todo el rango, registrándose el peor caso a la mayor frecuencia permitida, dando 1,18% de error a fondo de escala para un rango dinámico del DAC ajustado en 1,1V.

Según (2), la resolución en frecuencia es de de 0,00465 Hz y la frecuencia máxima es de 10MHz. Considerando la frecuencia de conversión del DAC, la frecuencia a generar es precisa hasta la centésima de Hz y la máxima frecuencia de una onda senoidal está limitada por la frecuencia de Nyquist de 384 kHz aunque la frecuencia máxima real se ve limitada a 100 kHz por los requerimientos de distorsión para otras formas de onda por la composición armónica. En las pruebas realizadas, se utilizó un HP 5315A Universal Counter de 7 dígitos y ½ disponible en el laboratorio y se observó una resolución en frecuencia de 0,01Hz coincidente con el valor esperado.

Los tiempos de subida y de establecimiento están dados por las características del DAC. El tiempo de subida es de 0,9µS/V y el de establecimiento es de 8µS/V para la configuración *slow* según su hoja de datos. Las mediciones con una señal cuadrada de 1, 10 y 100 kHz arrojaron un tiempo de subida similar en todos los casos de 1,1µS con una variación de ±0,3µS/V y un tiempo de establecimiento de 1,2µS±0,2µS/V para las tres frecuencias mencionadas, observándose un ciclo de trabajo porcentual de 50% con una variación menor a ±0,1%. El resumen de prestaciones obtenidas se consigna en la Tabla 2.

TABLA 2.PRESTACIONES

| | Parámetro | Valor medido |
|---|---|---|
| ***General*** | Resolución en Frecuencia | 0,01 Hz |
| | Rango de amplitud | 0~3,3Vpp |
| | Control de Amplitud | 200 mV |
| | Rango de Offset | 0~3,3 V |
| | Impedancia de Salida | 50 Ω |
| ***Seno*** | Frecuencia máxima | 100KHz |
| | Distorsión armónica total | 1,31% |
| | Distorsión espuria | -68,32dB |
| ***Cuadrada*** | Frecuencia máxima | 10kHz |
| | Tiempo de subida | 1,1µS ±0,3µS/V |
| | Tiempo de establecimiento | 1,2µS ±0,2µS/V |
| | Overshoot | 0,2% |
| | Ciclo de trabajo | 50 %±0,1% |
| ***Arbitraria*** | Frecuencia máxima | 10kHz |
| | Longitud | 4096 muestras |
| | Resolución | 10 bits |
| | Frecuencia de muestreo | 750 kHz |

Como se observa en la Tabla 3, se utilizaron pocos recursos de los disponibles en la FPGA, permitiendo la integración de nuevas funciones e instrumentos. El consumo de potencia en la FPGA es de 19,32 mW y el reporte temporal acusa una frecuencia límite de 43,027 MHz superior a la especificada.

TABLA 3. REPORTE DE RECURSOS DE LA FPGA

| Recursos utilizados | Usados/Disponibles (%) |
|---|---|
| CORE Used | 3573/38400 (9,30%) |
| IO (W/ clocks) Used | 100/223 (44,84%) |
| GLOBAL (Chip+Quadrant) Used | 11/18 (61,11%) |
| PLL Used: 1 Total | 1/2 (50,00%) |
| RAM/FIFO Used | 12/60 (20,00%) |

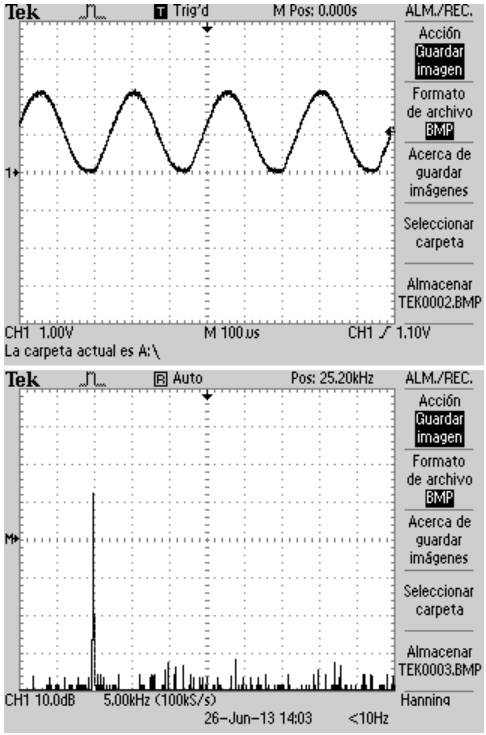


Figura 7. Osciloscopio y analizador de espectro mostrando una señal senoidal generada de 10 kHz como ejemplo.

## V. CONCLUSIONES

Se ha logrado desarrollar un generador de señales con funciones implementadas en una FPGA cuyo control es realizado a través de un software para PC. Se ha utilizado un diseño de hardware modular, facilitando futuras mejoras a partir del desarrollo. Se han aprovechado los elementos incorporados en la Fusión Development kit sin necesidad de utilizar componentes adicionales. La interfaz gráfica en Qt es simple e intuitiva, y permite operar el instrumento como uno tradicional. Se han utilizado las bibliotecas disponibles para varias plataformas y el código está íntegramente escrito en un lenguaje portable estructurado adecuadamente para que sea fácil su lectura, realización y traspaso a otra plataforma.

## VI. POSIBLES MEJORAS Y TRABAJOS FUTUROS

Actualmente se está trabajando en el agregado de modulación AM, FM y PSK. El DDS elegido es apto también para implementar un modulador Spread Spectrum. Exceptuando los tres módulos definidos como primitivas, el diseño del hardware descrito en VHDL es portable por lo que se está trabajando para probar el diseño en otras FPGAs. Además de mejorar la frecuencia máxima cambiando el DAC, se podría optimizar el módulo generador pues el reporte temporal lo señala como el de mayor limitación. Aun a costa de un diseño más complejo, sería posible migrar de la configuración DDS a una *Point Per Clock* PPC utilizando el PLL reconfigurable de la FPGA. Por otra parte, se podría compilar la GUI a un sistema operativo de código abierto como Linux apto para Qt.

## REFERENCIAS

[1] A. Cicuttin, M. L. Crespo, A. Shapiro y N. Abdallah, "A block-bassed open source approach for a reconfigurable virtual implementation platform ussing FPGA technology", IEEE International Conference on Reconfigurable Computing & FPGA's ReConFig 2006, 2006, pp 1 a 8.

[2] A. Manuel, D. Biel, J. Olivé, J. Prat. F. J. Sánchez, "Instrumentació virtual. Adquisició, processament i anàlisi de senyals", Ediciones UPC, junio 2001, ISBN 84-8301-473-4

[3] H. Goldberg."What is Virtual Instrumentation", IEEE Instrumentation & Measurement Magazine, dic. 2000, pp. 10-13.

[4] V. Smieško, K. Kovač. "Virtual Instrumentation And Distributed Measurement Systems", Journal of Electrical Engineering, Vol. 55, No. 1-2, 2004, pp. 50-56.

[5] G.R. Tsai, M.C. Lin, G.S. Sun, Y.S. Lin. "Single chip FPGA-based reconfigurable instruments". The International Conference on Reconfigurable Computing and FPGAs, sep. 2004.

[6] J.W. Hsieh, G.R. Tsai, M.C. Lin. "Using FPGA to implement a n-channel arbitrary waveform generator with various add-on functions". 2nd IEEE International Conference on Field-Programmable Technology, dic. 2003, pp. 296–298.

[7] M.J. Moure, M.D. Valdes, E. Mandado. "Educational Applications of Reconfigurable Hardware Based Virtual Instruments". 29th ASEE/IEEE Frontiers in Education Conference, nov. 1999, pp. 12C6/17.

[8] M.D. Valdes, M.J. Moure, C. Quintans, E. Mandado. "A Data Acquisition Reconfigurable Coprocessor for Virtual Instrumentation Applications". International Conference on Field-Programmable Logic and Applications, sep. 2003, pp. 1107-1110. 1993. www.tek.com.

[9] L. Cordesses. "Direct Digital Synthesis: A Tool for Periodic Wave Generation (Part 1)". IEEE Signal Processing Magazine, jul. 2004.

[10] E. Murphy, C. Slattery. "All about Direct Digital Synthesis". Analog Dialogue 38-08, ago. 2004.

[11] Documentos de Actel Corp. www.microsemi.com.

[12] Manuales de uso del Libero, Designer, Flash Pro y ModelSim.

[13] Synplify® DSP User Guide.

[14] K. Skahill, "VHDL for programmable logic", Addison-Wesley, 1996.

[15] J. Bhasker, "VHDL primer", 3th edition, Prentice Hall, 1999.

[16] Simon Doherty, Waveform Gen, OpenCores. www.opencores.org/project,waveform_gen.

[17] Simon Doherty, Waveform Gen, OpenCores. www.zipcores.com/opencores/waveform_gen.pdf.

[18] Linear Feedback Shift Register, Xilinx logic core DS257, mar. 2003.

[19] VipinLal, LFSR, Open Cores. www.opencores.org/project,lfsr_randgen.

[20] WISHBONE System-on-Chip (SoC) Interconnection Architecture for Portable IP Cores, OpenCores Organization, sep. 2002. www.opencores.org.

[21] Qt–A cross-platform application and UI framework. www.qt.nokia.com.

[22] Qwt-Qt Widgets for Technical Applications. www.qwt.sourceforge.net.

[23] F. Aguilera, C. Sosa Páez, D. Costa, "Implementación de un osciloscopio en una plataforma de instrumentación virtual recofigurable". Congreso de Microelectrónica Aplicada 2010. Tercer Milenio. ISBN 978-978-9374-65-8, pp. 110 a 112.

[24] E. Trabes, C. Sosa Páez, V. Yelpo, D. Costa. "Instrumentación virtual en tiempo real con FPGA: Analizador de Espectro". Congreso de Microelectrónica Aplicada 2011. ISBN 978-950-34-0749-3, pp. 115 a 119.

[25] V.F. Kroupa, V Cizek, J. Stursa, H. Svandova. "Spourious signals in direct frecuency synthetizers due to the phase truncation". IEEE trnas. Ultrason, ferroelect, freq contru, vol 47, No 5, sep. 2000.

[26] F. Curticapean, J. Niittulahti. "Exactanalysis of spurious signals in direct digital frecuency synthesizers due to phase truncation". Electron Lett. Vol. 39 No. 6, mar. 2003.