$\mathcal{DIFU}_{100}ci@$
Revista de Difusión Científica, Ingeniería y Tecnologías
Vol. 13, No. 2, Mayo - Agosto 2019
ISSN:2007-3585
Artículo arbitrado
Publicado: 31 de agosto de 2019

Performance analysis of a FIR filter in RFNoC

Análisis de desempeño de un filtro FIR en RFNoC

Grecia Montoya-Zúñiga[1], **Víktor I. Rodríguez-Abdalá**[*1], **Salvador Ibarra-Delgado**[1], **Remberto Sandoval-Aréchiga**[1], and **Jorge Flores-Troncoso**[1]

[1] *Universidad Autónoma de Zacatecas (UAZ), Unidad Académica de Ingeniería Eléctrica,*
Centro de Investigación, Innovación y Desarrollo en Telecomunicaciones,
Av. López Velarde 801, Col. Centro, Zacatecas , Zac., México, 98000.
grecia.montoya.zu@gmail.com,{abdala,sibarra,rsandoval,jflorest}@uaz.edu.mx

Abstract

The hardware accelerators development in software-defined radio ecosystems, allow increasing the samples per second processing capacity of wireless communication systems, tools such as RFNoC enable the FPGA to perform electronic designs in a distributed network in USRP devices, in this way, these signal processing blocks are seamlessly integrated into the GNU Radio flow diagram with the advantage of hardware processing. Processes such as signal filtering are well known for their high computational load, so their implementation in hardware means less processes performed by the general-purpose computer as well as a better use of the FPGA, which is underused in USRP devices.

***Keywords*—** GNU Radio, RFNoC, FIR

Resumen

La implementación de aceleradores de hardware en ecosistemas de radio definido por software, permiten aumentar la capacidad de procesamiento de muestras por segundo de los sistemas de comunicaciones inalámbricos, herramientas como RFNoC habilita el FPGA para implementar diseños electrónicos en una red distribuida en los dispositivos USRP, de este modo, estos bloques de procesamiento de señales se integran de forma transparente al diagrama de flujo de GNU Radio con la ventaja que implica el procesamiento en hardware. Procesos como el filtrado de señales son bien conocidos por su alta carga computacional, por lo que su implementación en hardware significa una descarga de procesos en la computadora de propósito general así como el aprovechamiento del FPGA es cual es subutilizado en los dispositivos USRP.

***Palabras clave*—** GNU Radio, RFNoC, FIR

I. Introducción

Los ecosistemas de SDR (del inglés Software Defined Radio) son herramientas de software que junto con dispositivos de hardware, permiten implementar sistemas de comunicaciones inalámbricos reconfigurables, con la característica de reemplazar procesos tradicionalmente implementados en hardware, con algoritmos software en una computadora de propósito general.

GNU Radio es un ecosistema de software libre para sistemas de radio enriquecido con numerosos bloques de procesamiento de señal, estos son escritos en lenguajes de programación C++ y Python en general, también incluyen implementaciones en hardware utilizando el lenguaje descriptor de hardware Verilog.

GNU Radio es utilizado principalmente con dispositivos USRP (del inglés Universal Software Radio Peripheral) los cuales son desarrollados y comercializados por Ettus Research, estos pueden funcionar tanto como receptor y transmisor para una amplia gama de frecuencias.

Si bien en GNU Radio, no se puede desarrollar o diseñar un bloque de procesamiento descrito por un lenguaje descriptor de hardware como lo es Verilog, RFNoC (del inglés Radio Frequency Network on Chip) es un entorno que permite el desarrollo de aplicaciones para FPGAs (del inglés Field Programmable Gate Array), de este modo GNU Radio crea un ecosistema compuesto por bloques definidos por software y bloques de RFNoC implementados en hardware de manera transparentes para el usuario, lo cual facilita concentrar el esfuerzo de desarrollo a un nivel más alto de abstracción [1].

[*] Autor de correspondencia

En el presente artículo se muestran los resultados de implementar un filtro FIR en tres diferentes bloques de procesamiento: en C, Python y en el lenguaje descriptor de hardware Verilog, esto con la finalidad de determinar las implicaciones de desarrollar bloques de procesamiento en los diferentes tipos de entornos, y así obtener los costos-beneficios del diseño de hardware contra la implementación en software.

El artículo se organiza de la siguiente manera: en la sección II se describe el ecosistema de radio definido por software GNU Radio así como la herramienta de RFNoC para la integración de bloques de procesamiento de señales en hardware dentro GNU Radio, así mismo se describen las especificaciones de la plataforma de hardware. En la sección III se detalla los parámetros de diseño del filtro FIR así como su implementación tanto en software como en hardware. En la sección IV se describen las pruebas a realizar sobre los bloques de procesamiento de señales para tiempo real y post-procesamiento. Finalmente, en las secciones V y VI se muestran los resultados obtenidos y las conclusiones de las pruebas realizadas.

II. Herramientas de Software y plataforma de Hardware

II.1. GNU Radio

GNU Radio es un conjunto de herramientas de desarrollo de software libre y gratuito, que provee bloques de procesamiento de señales para implementarlos en radios definidos por software, puede usarse en conjunto con hardware de RF o sin él, para crear un ambiente de simulación.

Puede realizar todo tipo de procesamiento de señales, entre ellos: filtros, codificadores de canal, demoduladores, decodificadores y muchos otros elementos de procesamiento de señales representados en forma de bloques. La característica más importante es la manera de interconectar estos bloques entre sí a través de un diagrama de flujo, típicamente el diagrama de flujo empieza con una fuente ya sea de hardware o un archivo, los datos obtenidos de la fuente se conectan a una serie de bloques de procesamiento de señales, finalmente se conecta a un bloque “sink” donde la señal obtenida después del procesamiento puede ser mostrada en tiempo real, almacenada o exportada.

Así mismo brinda un conjunto de herramientas para que el usuario pueda crear bloques de procesamiento de señales personalizados, llamada $gr\_modtool$, que es una interfaz estandarizada con las plantillas necesarias para crear módulos OOT (del inglés Out of Tree). El código de los bloques de procesamiento de señales puede ser escrito en los lenguajes de programación Python y C++, cada uno con sus respectivas ventajas, siendo Python un lenguaje de alto nivel puede usarse para aplicaciones que requieren operaciones complicadas pero que no demandan muchos recursos a la computadora; mientras que C++ es más rápido y pueden utilizarse las bibliotecas de C++ incluidas en GNU Radio.

II.2. RFNoC

RFNoC es una herramienta de procesamiento de software libre utilizada para el desarrollo de aplicaciones de SDR en los dispositivos de Ettus Research, generalmente conocidos como USRP. El concepto de NoC (del inglés Network on Chip) se refiere a un sistema integrado en un solo chip usado para intercambiar información y control entre los PEs (del inglés Processing Engines) internos [2].

De manera interna todos los bloques de RFNoC tienen la misma estructura llamada NoC-Shell como muestra la Fig.1, permitiendo la conexión de cualquier IP core que funcione con el protocolo de comunicación AXI-Stream, este se encarga de empaquetar, desempaquetar, enrutar, el control de flujo y que la configuración sea común entre los bloques.

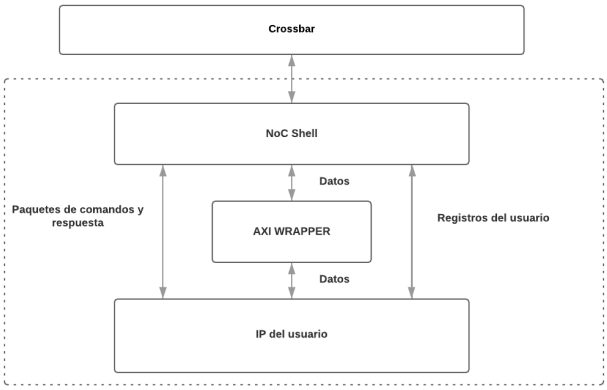

Figura 1: *Estructura interna de un bloque RFNoC*

Un bloque RFNoC tiene tres capas principales, tal como se puede observar en la Fig. 2, las cuales tienen las siguientes funciones:

1. El IP del FPGA, donde se concentra la mayor parte del trabajo realizado, puede ser escrita en Verilog, VHDL, o integrar un IP de Xilinx, con la única condición de que use el protocolo AXI Stream. Se deben de considerar las señales de entrada - salida previamente establecidas y el uso de los registros.
2. El control del bloque representa la lógica digital en el lado del software, declara el flujo de datos entrada - salida, y declara las propiedades reconfigurables del bloque de procesamiento de señales por medio de un archivo XML.

3. La integración a GNU Radio se logra agregando los bloques RFNoC como un modulo OOT. la programación de los archivos de enlace es muy similar a la de GNU Radio.

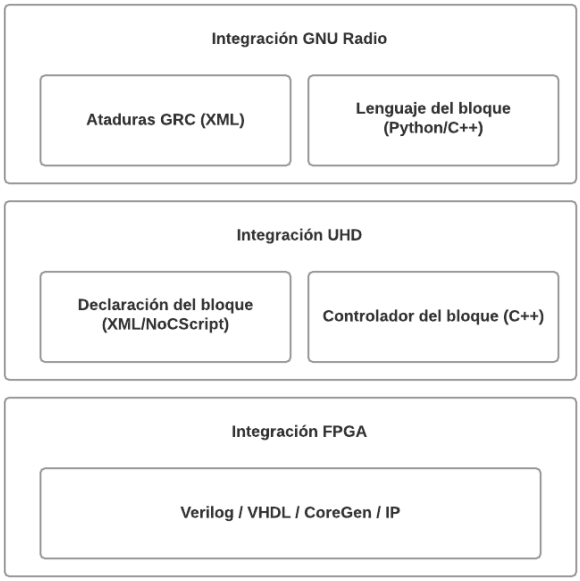

Figura 2: *Componentes para el desarrollo de un bloque RFNoC.*

II.3. USRP X300

Ettus Research USRP X300 es una plataforma de SDR de alto rendimiento para diseñar e implementar sistemas de comunicación inalámbricas de próxima generación, este cuenta con las siguientes características [3]:

- Dos ranuras para las tarjetas hijas de RF de banda ancha con las siguientes características:
  - Hasta 160 MHz de ancho de banda por cada una.
  - Cubre desde CD (corriente directa) hasta 6 GHz.
- Consta de un FPGA Kintex-7 de Xilinx de alto rendimiento para procesamiento digital de señales y diseño de hardware.
- Varias interfaces de alta velocidad:
  - Dual Ethernet de 10 Gigabit: 2x RX a 200 MSps (del inglés Samples per second) por canal.
  - Dual Ethernet de 10 Gigabit: 4x RX a 80 MSps por canal.
  - PCIe Express (Escritorio) - 200 MSps Full Duplex.
  - ExpressCard (Laptop) - 50 MSps Full Duplex.
  - Ethernet Dual 1 Gigabit - 25 MSps Full Duplex.
- La arquitectura UHD (del inglés USRP Hardware Driver) proporciona compatibilidad con:
  - GNU Radio.
  - C++/Python API (del inglés Application Programming Interface).
  - Amarisoft LTE 100.
  - OpenBTS.
- OCXO (del inglés Oven Controlled Crystal Oscillators) opcional controlado por GPS para sincronización con otras tarjetas USRP.

III. Filtro FIR

III.1. Coeficientes del filtro FIR

El filtro se diseñó utilizando la función de transferencia de un filtro pasa bajas de $M = 16$ taps, es decir de orden $N = 15$, una frecuencia de corte de 100 kHz y una frecuencia de muestreo de 256 kSps.

El cálculo de los coeficientes del filtro se define multiplicando un filtro ideal pasa bajas $h_{faselineal}(n)$ por la ventana Hamming $W_{Hamming}(n)$.

De este modo se determinó que los taps o coeficientes del filtro que cumplen con los parámetros de diseño son los que se muestran en la Tabla 1 :

Tabla 1: *Coeficientes del filtro FIR.*

Tap	Coeficiente
h[-8]=h[8]	-0.00145482
h[-7]=h[7]	-0.00142821
h[-6]=h[6]	0.01081722
h[-5]=h[5]	-0.02816919
h[-4]=h[4]	0.03971448
h[-3]=h[3]	-0.01441649
h[-2]=h[2]	-0.09972695
h[-1]=h[1]	0.59466396

Finalmente, en la Figura 3 se muestran los coeficientes del filtro FIR.

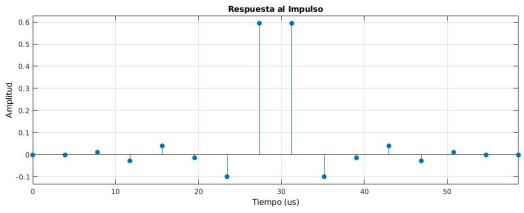

Figura 3: *Respuesta al impulso unitario.*

III.2. Filtro en Software (C++ y Python)

Para el correcto funcionamiento del filtro en su implementación en software, se debe inicializar el bloque para asegurar el número de elementos disponibles en la convolución del filtro, por ello este debe tener el mismo tamaño que la función de transferencia del filtro $M = 16$. Esto significa que los elementos de salida se calculan a partir del elemento de entrada actual y de los elementos de entrada anteriores. La función $set\_history(16)$ dentro del bloque de procesamiento de GNU Radio escrito en C permite asegurar la cantidad de elementos en el bloque de procesamiento.

La función $work$ del bloque de procesamiento de señal en C es donde se realiza el procesamiento de la señal y regresa el resultado del filtro. De manera muy similar se realiza el procesamiento de señal en el bloque de GNU Radio escrito en Python.

III.3. Filtro en Hardware (Verilog)

RFNoC crea su propio entorno de desarrollo en conjunto con la herramienta para el análisis y diseño de circuitos digitales de Xilinx Vivado. A modo demostrativo existen imágenes para cargar al FPGA con algunos bloques básicos, además consta de una interfaz gráfica para configurar una nueva imagen para el FPGA a través del script $uhd\_image\_builder.py$.

RFNoC cuenta con un bloque FIR de ejemplo, el cual puede ser reconfigurado cambiando los valores de los coeficientes del filtro por parte del usuario. Este bloque acepta sólo números de tipo entero, por lo cual se debe de hacer una transformación del tipo de dato flotante a una representación binaria de 16 bits con signo.

$$\begin{aligned} coef_{int} &= [coef_{float} * 2^{15}] \\ &= [-48, -47, 354, -923, 1301, -472, \ldots] \end{aligned} \tag{1}$$

Al cambiar el tipo de dato este genera un error por redondeo en los coeficientes del filtro implementado en hardware, para determinar su impacto en la señal filtrada este se calcula de la siguiente manera:

$$coef_{fixed} = \frac{coef_{int}}{2^{15}} \tag{2}$$

$$\%error = |\frac{coef_{fix} - coef_{float}}{coef_{float}}| * 100 \tag{3}$$

$$\%error_{acumulado} = \sum \%error = 2.7383\,\% \tag{4}$$

El error acumulado por redondeo de los coeficientes del filtro puede ser considerado despreciable, ya que estos niveles de error no afectan de forma sustancial a la señal filtrada.

IV. Implementación del filtro FIR en la plataforma SDR

IV.1. Cama de prueba de Hardware

La computadora de propósito general utilizada para los experimentos tiene las siguientes especificaciones: 8 GB de memoria RAM y procesador Intel Xeon Bronze 3104 a 1.7Ghz de 6 núcleos. El dispositivo USRP se comunica en la capa IP / UDP a través de Gigabit y 10 Gigabit Ethernet según sea el caso. Las conexiones del dispositivo USRP 300 se muestran en la Fig. 4.

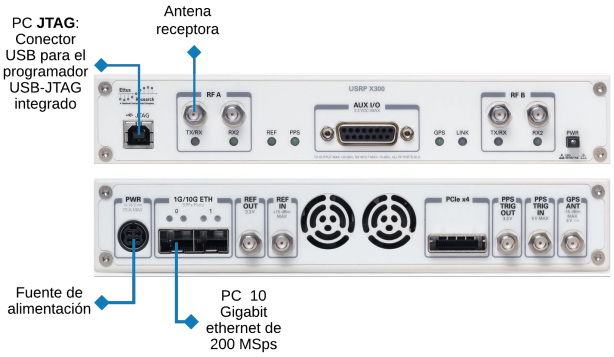

Figura 4: *Conexiones del USRP X300.*

IV.2. Banco de filtros

En la Fig.5 se muestra la implementación de los tres filtros en un diagrama de flujo de GRC (del inglés GNU Radio Companion), que es la interfaz donde se implementa de manera gráfica los sistemas de comunicaciones en GNU Radio. Estos tienen como entrada la suma de dos señales sinusoidales generadas por los bloques *Signal Source* con las frecuencias de 100 kHz y 120 kHz respectivamente. Debido a que el bloque de RFNoC FIR tiene una amplificación de 2^{15} por el redondeo de los coeficientes, es conveniente multiplicarlo por su inverso para realizar una comparación justa del comportamiento de los filtros y así se muestren con la misma ganancia.

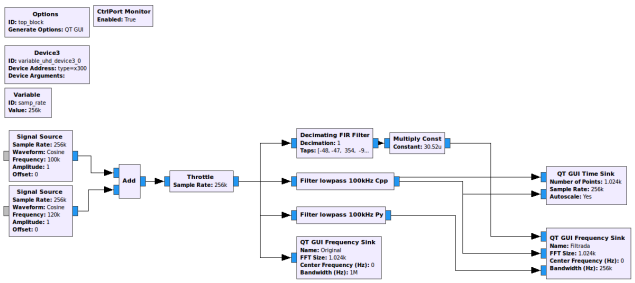

Figura 5: *Diagrama de flujo del banco de filtros.*

La Fig.6 es la interfaz gráfica del diagrama de flujo implementado en GRC, en el primer recuadro se muestran tanto la señal original (señal azul) como la señal filtrada (señal verde) en el dominio del tiempo, en esta comparación se puede observar que la señal filtrada no tiene los mismos valores que la señal original, esto por los efectos del filtrado en el dominio del tiempo; en el segundo recuadro se muestra la señal original en el dominio de la frecuencia, claramente se observa los dos componentes espectrales de la señal de entrada; y finalmente, en el tercer recuadro se muestra la salida de los tres filtros, donde tienen el mismo comportamiento en frecuencia, lo que permite determinar que a pesar de ser implementados en diferentes lenguajes de software y hardware, este tiene el comportamiento deseado, de igual manera se puede observar que la frecuencia de 100 kHz tiene una atenuación de -6 dB y la frecuencia de 120 kHz tiene una atenuación de -26 dB, de acuerdo a los parámetros de diseño del filtro.

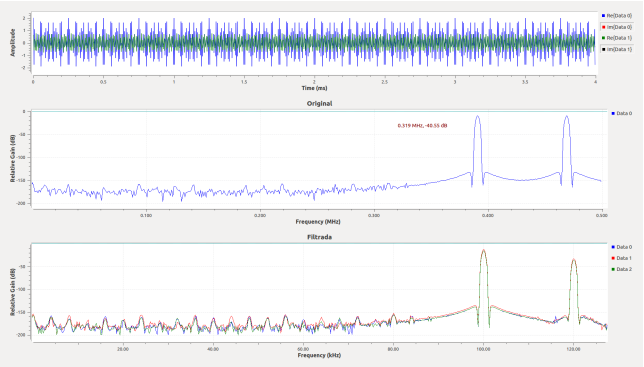

Figura 6: *Señal original y filtrada.*

IV.3. Implementación de un receptor de frecuencia modulada (FM)

En el diagrama de flujo que muestra la Fig. 7 se implementó un demodulador FM con dos de los filtros, el de RFNoC y el de C++, el filtro diseñado con Python presentó una latencia alta al momento de procesar las muestras en tiempo real, por lo que se descartó en este diagrama de GRC.

El receptor de FM permite interactuar al usuario por medio de la interfaz gráfica que se muestra en la Fig. 8, en ella se observa el espectro de la señal de la estación seleccionada en tiempo real.

El bloque FIR implementado en Python no puede procesar las muestras en tiempo real ya que no es implementado en un lenguaje con compilador sino en un lenguaje intérprete, se realizó un diagrama en GRC donde la señal demodulada es almacenada en un archivo de audio WAV para que sea reproducida en post-procesamiento, tal como se muestra en la Fig. 9.

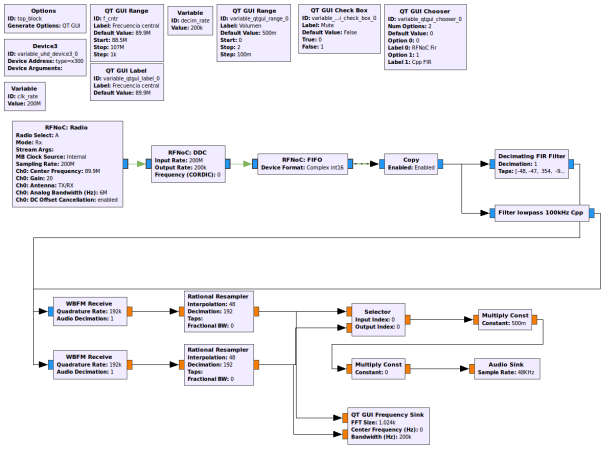

Figura 7: *Diagrama de flujo del receptor FM.*

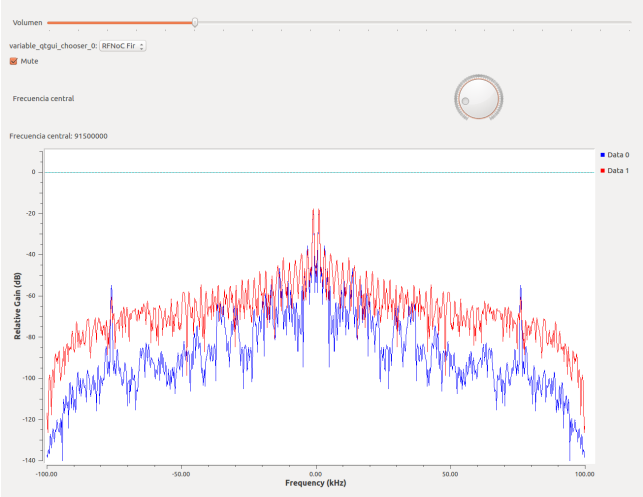

Figura 8: *GUI de Radio FM (RFNoC FIR y C++).*

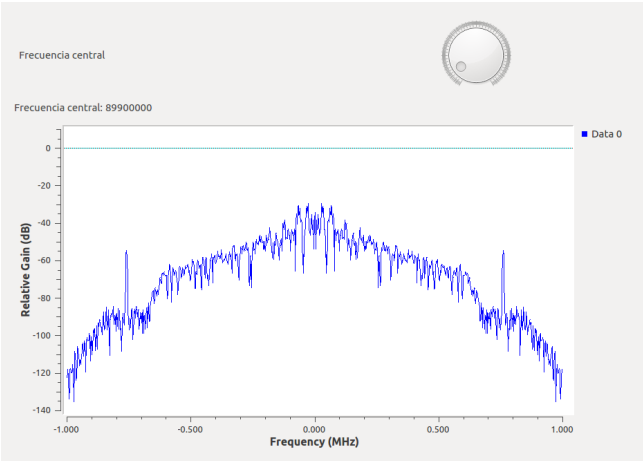

Figura 9: *GUI de Radio FM (Python).*

V. Resultados

Para medir el desempeño de los filtros FIR implementados en GNU Radio se determinó el número máximo de muestras procesadas por segundo por cada bloque individualmente, utilizando una función de monitor de procesos de GNU Radio que depende del kernel de Linux.

La herramienta *Performance Counters* de GNU Radio muestra la información específica del comportamiento en tiempo real de los recursos de memoria y procesamiento consumidos por cada bloque implementado. Esta es una manera viable que permite el monitoreo del comportamiento de un programa, brindando un beneficio técnico que permite a los usuarios tener información específica en un sistema de comunicación implementado en GNU Radio [4].

Los resultados obtenidos se muestran en la Tabla 2, como se esperaba, el filtro FIR implementado en hardware tiene una mayor capacidad de procesamiento de muestras por segundo en comparación con los implementados en software, aunque el resultado obtenido era el esperado, esta implementación permite cuantificar las ventajas que tiene el utilizar un bloque de procesamiento de hardware en una plataforma de software como GNU Radio. En cambio, aunque la implementación de un bloque de procesamiento de señales en Python puede ser sumamente sencillo en comparación con C y Verilog, presenta una gran desventaja en cuanto a procesamiento en tiempo real.

Tabla 2: *Número máximo muestras por segundo promedio.*

	Muestras por segundo
Python	25 131
C++	3 126 710
Verilog	5 240 000

VI. Conclusiones

Los resultados obtenidos en la implementación de los filtros FIR en diferentes lenguajes de software y en hardware, permitió determinar la factibilidad de desarrollar sistemas de radio definido por software en conjunto con FPGAs a nivel de prototipo. Así mismo se puede realizar una clasificación de bloques de procesamiento de señales, donde dependiendo de la capacidad de procesamiento estos sean realizados en hardware o software, e incluso determinar si este puede ser escrito en otro tipo de lenguaje de software de acuerdo a su importancia.

La desventaja principal en el desarrollo de bloques de procesamiento de radio definido por software en hardware, es el tiempo de desarrollo y el nivel de conocimiento tanto de diseño electrónico como de software de parte del desarrollador. Lenguajes de software como Python permiten la implementación de bloques de procesamiento en tiempos menores, lo que es útil para pruebas de concepto en el desarrollo de prototipos, una vez comprobado el correcto funcionamiento de este, se puede realizar la implementación en hardware en la plataforma de radio definido por software, lo que implicaría solamente sustituir un bloque de procesamiento por otro, sin modificar todo el sistema desarrollado en radio definido por software.

Agradecimientos

Gracias al programa de Apoyo a la Incorporación de NPTC en el proyecto denominado “Implementación de aceleradores de hardware a través de RFNoC en radio definido por software” por la beca de tesis.

Referencias

[1] Andrea Guerrieri. «Designing a RFNoC Block implementing a SISO Processor using High-Level Synthesis». En: *Proceedings of the GNU Radio Conference*. Vol. 2. 1. 2017, págs. 8-8.

[2] Martin Braun, Jonathan Pendlum y Matt Ettus. «RFNoC: RF network-on-chip». En: *Proceedings of the GNU Radio Conference*. Vol. 1. 1. 2016.

[3] Ettus Research. *USRP X300 High Performance Software Defined Radio Ettus Research*. URL: https://www.ettus.com/all-products/x300-kit/.

[4] Rafik Zitouni, H Bouaroua y B Senouci. «Hardware-Software Codesign for Software Defined Radio: IEEE 802.11p receiver case study». En: *RICESS 2017*. Ago. de 2017.