# UNIVERSIDAD AUTÓNOMA DE ZACATECAS

## DISEÑO E IMPLEMENTACIÓN DE UN ANALIZADOR MULTICANAL PARA ESPECTROMETRÍA NUCLEAR CON ZYNQ UTILIZANDO VIVADO

**I.C. Ángel García Durán**

Tesis de Maestría

presentada a la Unidad Académica de Estudio Nucleares

de acuerdo a los requerimientos de la Universidad para obtener el Grado de

**MAESTRO EN CIENCIAS NUCLEARES**
**CON ORIENTACIÓN EN INSTRUMENTACIÓN NUCLEAR**

**Directores de tesis:**

**M. en C. Víctor Martin Hernández Dávila (UAEN-UAZ)**

**M. I. A. Oscar Osvaldo Ordaz García (UAIE-UAZ)**

**M. I. T. C. Salvador Ibarra Delgado(UAIE-UAZ)**

**Zacatecas., Zac., México. Agosto 2017**

# Resumen

Las diferentes aplicaciones de la radiación ionizante hace de esta, una herramienta muy significativa y útil, a su vez puede ser peligrosa para los seres vivos si son expuestos a dosis no controladas. Sin embargo, por sus características, no puede ser percibida por los cinco sentidos del ser humano, de tal manera que para conocer la presencia de esta se requieren de detectores de radiación y dispositivos adicionales que permitan cuantificarla y clasificarla. Este es el caso del analizador multicanal que se encarga de separar las diferentes alturas de pulso que se generan en los detectores, en un número determinado de canales; acorde al número de bits del convertidor análogo a digital. El desarrollo o acondicionamiento de tecnología nuclear ha aumentado considerablemente por la demanda de las aplicaciones, por consiguiente esto permite desarrollar sistemas que se adecuen a las necesidades del usuario, con características como reducción en el costo y volumen de los dispositivos. El objetivo del trabajo fue diseñar e implementar un núcleo de propiedad intelectual (IPcore) el cual funciona como un analizador multicanal para espectrometría nuclear. Los componentes del IPcore fueron creados en lenguaje de descripción de hardware VHDL y empaquetados en la suite de diseño Vivado, haciendo uso de los recursos como son el núcleo de procesamiento ARM que el chip Zynq contiene. Así mismo, para la primera fase de la implementación fue embebida en la FPGA la arquitectura hardware y programada en lenguaje C la aplicación para el procesador ARM. Para la segunda fase, el manejo, control y visualización de los resultados se desarrolló un instrumento virtual en la plataforma gráfica de programación LabVIEW. Los datos obtenidos como resultado del desarrollo e implementación del IPcore fueron observados gráficamente en un histograma que forma parte del instrumento virtual antes mencionado. Además los resultados obtenidos con el analizador multicanal embebido en la FPGA, tienen una gran semejanza con los resultados de analizadores multicanal comerciales.

# Abstract

The different applications of ionizing radiation has made this, a very significant and useful tool, in turn can be dangerous for living beings if exposed to uncontrolled doses. However, due to its characteristics, it can not be perceived by the five senses, so that to know the presence of this are required of detectors of radiation and additional devices that allow us to quantify and classify it. This is the case of the multichannel analyzer that is responsible for separating the different pulse heights that are generated in the detectors, in a determined number of channels; According to the number of bits of the analog-to-digital converter. The development or conditioning of nuclear technology has increased considerably by the demand of the applications, therefore this allows to develop systems that cover some commercial requirements cost and volume in relation to the needs of the user. The objective of the work was to design and implement an IP Core, which functions as a multichannel analyzer for nuclear spectrometry. For the IPcore design methodology, its components were created in hardware description language VHDL and packaged in the Vivado design suite, making use of resources such as the ARM processor cores that the Zynq chip contains. Also, for the first phase of the implementation, the hardware architecture was embedded in the FPGA and the application for the ARM processor was programmed in C language. For the second phase, the management, control and visualization of the results was developed a virtual instrument in the graphical platform of programming LabVIEW. The data obtained as a result of the development and implementation of IPcore were observed graphically in a histogram that forms part of the virtual instrument mentioned above.

# Agradecimientos

Principalmente agradezco al Consejo Nacional de Ciencia y Tecnología (CONACYT) por todo el apoyo económico brindado para el desarrollo de la investigación durante los 2 años que duro la maestría, a la Universidad de Autónoma de Zacatecas por darme la oportunidad de ser uno de sus estudiantes y a la Universidad de Alcalá por haberlo recibido en sus instalaciones durante la estancia donde se realizo gran parte del proyecto.

Me gustaría reconocer a la Unidad Académica de Estudios Nucleares y a todos sus docentes y trabajadores, por siempre dar lo mejor de si para que el estudiante pueda triunfar en sus metas.

Agradezco ampliamente en este trabajo a mis asesores: M. en C. Víctor Martin Hernández Dávila, M. I. A Oscar Osvaldo Ordaz García, M. I. T. C. Salvador Ibarra Delgado y Dr. Ignacio Bravo Muñoz, por su apoyo, confianza, enseñanza y por guiarme con respeto y con una infinita disposición para alcanzar cada una de las metas planteadas durante todo el transcurso del trabajo de tesis.

De igual manera deseo extender mis mas sinceros agradecimientos a los Drs. Héctor René Vega Carrillo y Carlos Ríos Martínez, por proporcionar de la manera mas cordial sus laboratorios para las pruebas y además brindar sus conocimientos y experiencias, necesarios para este trabajo.

Y sobre todo a mi familia y amigos por todo su apoyo y comprensión a través de los años, en especial a mis Padres Salvador y Crucita que con su ejemplo me guiaron y enseñaron que con el trabajo duro y la honradez alcanzaría mis sueños; a mi esposa Rosa Elena Garcia Saldivar que con su amor, siempre estuvo a mi lado y me apoyo incondicionalmente en las decisiones tomadas durante esta etapa de mi vida y por ultimo agradezco a la fuente de mi inspiración y mis ganas de seguir progresando en la vida a mi hija adorada **PAG$^2$**.

# Índice

# Índice de Figuras

# Índice de Tablas

# Capítulo 1. Introducción

La radiación es la emisión de partículas o fotones debido a la desintegración de su núcleo atómico, esto puede tener variación con respecto al número de nucleones y cambios energéticos, hasta conseguir un núcleo estable. A sí mismo, una parte muy importante en la medición de radiación es la espectrometría nuclear, que ha sido la columna vertebral de muchas aplicaciones, ya que permite, mediante el nivel de energía identificar la radiación y su radioisótopo, proporcionando información sistemática y detallada (Ibarra and Pabón, 2015, Adler et al., 2010).

Sin embargo, un estudio a fondo del riesgo que implica la radiación es sin duda, el mejor método para conocer la naturaleza de esta y permite conocer la tolerancia. Además, estos métodos poseen desventajas relativamente complejas y caras, asimismo de un tiempo muy lago para su implementación; también de personal especializado. Sin embargo se utiliza la instrumentación nuclear para identificar y cuantificar radionúclido alfa, beta, gamma, de neutrones, etc. (Oramas and Figueroa, 2014).

Además, si se considera como un ejemplo a la radiación gamma, se establece que ésta, es de origen electromagnético y nuclear, igualmente que tiene gran poder de penetración al interactuar con la materia, debido a su alta energía, ausencia de carga y masa; por lo que desplaza los electrones del átomo a través de tres efectos (fotoeléctrico, *Compton* y producción de pares). Con relación a lo anterior, la espectrometría gamma es contemplada como una de las más importantes, reflexionar así, que la radiación gama de baja actividad no es destructiva y es apropiada para el estudio y la inducción en la tecnología del analizador multicanal (Ibarra and Pabón, 2015).

Por otra parte, la espectrometría alfa permite la identificación y cuantificación de isótopos radioactivos emisores de este tipo. Estas partículas están constituidas por dos protones y dos neutrones, equivalente a un núcleo helio. Son emitidas por algunos elementos

radioactivos (inestables), con energías del orden de los MeV (Mega Electrón Volts). Además, son consideradas extremadamente peligrosas, por esta razón su identificación y cuantificación son de gran importancia, tanto en la seguridad radiológica, como en la caracterización de zonas ambientales. Las características generales de los diferentes tipos de radiación ionizante así como sus propiedades fisicoquímicas y la interacción de la radiación con la materia son importantes y elementales para identificar y cuantificar la radiación, esta información se utilizará para el desarrollo y aplicación de la espectrometría nuclear (García, 2006).

Así mismo, una rama de la instrumentación nuclear está enfocada en la espectrometría, un ejemplo de ello es el analizador multicanal (MCA: MultiChannel Analyzer), que tiene como finalidad almacenar la intensidad de la radiación detectada, mediante el valor de su energía. El mismo, está conformado por elementos esenciales como el subsistema discriminador que permite seleccionar la energía de la radiación cuya intensidad se va a medir, una memoria en la que se almacena la distribución de intensidades, y un subsistema para visualización del espectro. Es necesario conciderar que la señal proveniente del detector tiene que ser acondicionada para que pueda ser transmitida correctamente hacia el analizador y por consiguiente ser utilizada y tratada adecuadamente (Dambacher et al., 2011).

En aplicaciones de espectrometría de radiación, la altura del pulso que proporciona información de energía de la radiación, se complementa con la forma del pulso de la señal del detector, que de manera importante, proporciona varios aspectos de detección; tales como el tipo de la radiación, la posición de la interacción dentro de un detector, el tiempo de interacción, el tipo de material del detector, entre otros. Sin embargo, la forma del pulso se representa generalmente como el tiempo de subida, que corresponde a la diferencia de tiempo entre dos porcentajes específicos relativos a la altura máxima del pulso. Debido a la complejidad de la construcción de circuitos analógicos para la obtención de un tiempo de respuesta adecuado, la alternativa de esto se puede lograr fácilmente con procesamiento digital de señales, esto permite el avance en el desarrollo de las necesidades tecnológicas en la espectrometría nuclear y ofrece ventajas sobre las tecnologías anteriores como el aumento en el costo comercial de *MCA's* (Castro et al., 2003, Lee et al., 2013).

Por lo tanto, se consideran algunas otras características que pueden ser modificadas para optimizar los sistemas, de tal manera que se pretende aumentar la disponibilidad de dispositivos portátiles basados en un multicanal de alto volumen que facilita la adquisición de

señales y el análisis de sus oscilaciones instantáneas y las correlaciones temporales entre señales (Róbert et al., 2015).

Además si se emplea un lenguaje de descripción de hardware de alto nivel, que puede ser utilizado en placas de desarrollo basadas con dispositivos *FPGA´s* en las cuales se integra los *System on Chip* de la familia Zynq de Xilinx ; que combina en un mismo circuito integrado un microprocesador (parte software) y una zona de lógica programable (parte hardware). Con la facilidad de trabajar con Xilinx que proporciona herramientas de desarrollo agrupadas en ISE Design Suite. Por otro lado, Xilinx ofrece Vivado IDE como herramienta de síntesis de alto nivel que permite trabajar a un elevado nivel de abstracción (Lema, 2016, Ortega, 2014, Vázquez, 2014).

La radiación ionizante es peligrosa tanto para el ser humano como para el medio ambiente. El ser humano no puede sentir y apreciar por si solo la radiación, por consiguiente se han creado dispositivos que permiten cuantificar los distintos tipos de radiación, de tal manera que son de suma importancia para la seguridad y protección radiológica. Sin embargo existen una variedad de detectores y complementos electrónicos que facilitan el sistema de la espectrometría nuclear; el analizador multicanal que se conecta al detector de radiación y tiene como principio separar en distintos canales la energía de radiación que depende del radioisótopo a analizar, proporciona información significativa para su cuantificación y visualización, estos dispositivos hoy en día los ofrecen algunas compañías (ORTEC, CANBERRA, MIT ELECTRONICS, etc.), con algunas desventajas como el alto costo, el espacio físico que ocupa, además de que este dispositivo tiene que ser conectado a una fuente de alimentación de (NIM-BIN) o instalados en una computadora de escritorio antigua con una terminal adecuada las cuales están en desuso.

Basado en estas características y hechos se plantea el siguiente problema científico.

## 1.1. Problema científico

Un analizador multicanal es un equipo que examina las alturas de pulso provenientes de un detector de radiación, procesa la información y la almacena para su posterior visualización. El diseño e implementación de un sistema multicanal en este trabajo, se lleva a cabo mediante la creación de un IPcore (Núcleo de Propiedad Intelectual), utilizando el

lenguaje descriptor de hardware VHDL, utilizando como herramienta de desarrollo la suite de diseño Vivado para la implementación sobre una placa de desarrollo ZedBoard la cual contiene un chip Zynq que es un SoC que tiene integrado dos procesadores ARM-A9 y un espacio para lógico programable. Además, se emplea la plataforma de desarrollo de LabVIEW para crear el entorno gráfico ejecutable en la computadora para el control, manejo y visualización de los datos obtenidos. Con esto se puede reducir el costo, el volumen del dispositivo en relación con los analizadores multicanal comerciales y por consiguiente satisfacer los requerimientos para los laboratorios de la Unidad Académica de Estudios Nucleares. Además el sistema embebido cumple las especificaciones técnicas de peso y volumen para ser adaptado como carga útil en un nano satélite de construcción mexicana.

## 1.2. Objetivos

### 1.2.1. Objetivo general

Diseñar e implementar un sistema de análisis multicanal para espectrometría nuclear utilizando el SoC Zynq y la herramienta de desarrollo de sistemas embebidos Vivado.

### 1.2.2. Objetivos específicos

1.- Determinar las señales generadas por el detector de radiación mediante un conversor análogo digital.

2.- Diseñar e implementar un IPcore para un dispositivo reconfigurable (FPGA), que considere las características de las señales.

3.- Programar la aplicación para el procesador ARM.

4.- Mostrar la información obtenida por el detector en una computadora utilizando una interfaz diseñada en LabVIEW.

# Capítulo 2. Revisión de literatura

En este capítulo se comentan los aspectos necesarios a considerar para desarrollar el analizador multicanal, partiendo del conocimiento general de la radiación, los detectores de radiación, también se incluye las características fundamentales de la placa de desarrollo que contiene el SoC utilizado en el proyecto y las características del software de desarrollo grafico LabVIEW.

## 2.1. Radiación

El solo escuchar la palabra radiación es capaz de despertar diversas emociones: interés, al anunciarse como título de una conferencia o discusión pública; temor, ante el anuncio de la puesta en marcha de una planta nucleoeléctrica; admiración, al observarse una radiografía que muestra la fractura del dedo pulgar del pie de alguna persona; esperanza, ante un tratamiento de radioterapia que puede salvar la vida del mejor amigo. Además de estas emociones, la sociedad considera que el tema de radiación es un aspecto en el cual aún se desconoce una gran parte y por tanto, no entiende cómo en ocasiones puede resultar tan benéfica y en otras causar daño. La escasez de información respecto a la radiación y sus efectos no es satisfecha por los medios de comunicación, que la mayor parte de las veces, divulgan noticias y reportajes incompletos y superficiales. Esta situación origina que el ciudadano común haga conjeturas erróneas, basadas en la mala información y más que nada, en el temor ante algo que no entiende y que le parece imposible llegar a comprender.

El ser humano ha estado expuesto a las radiaciones ionizantes desde su aparición sobre la tierra, pero sólo fue capaz de identificarlas y usarlas desde 1895 cuando Wilhelm Konrad Roentgen descubrió los rayos X. Durante casi un siglo se ha trabajado para profundizar el conocimiento y ampliar las aplicaciones de la radiación y otras nuevas formas de la energía, y aumentar así el dominio sobre las fuerzas de la naturaleza.

El control en las normas que rigen el peligro de la radiación, ha permitido el progreso en diversas áreas del saber y específicamente en medicina, abriendo así nuevas posibilidades terapéuticas y de diagnóstico, ha contribuido a un mejor conocimiento de la fisiología humana, así como a identificar la causa de algunas enfermedades y por ende, a adecuar el

tratamiento, esto da como resultado que se prolongue y mejore la calidad de vida del ser humano.

La dualidad en los usos de la radiación, para fines benéficos o destructivos, fue imaginada desde el principio por sus descubridores. Cuando Pierre Curie en Estocolmo recibió con su esposa Marie el premio Nobel en 1903 señaló: *"Soy de aquellos que piensan que la humanidad obtendrá más beneficio que daño con estos nuevos descubrimientos."* En esta frase queda implícito que estaba consciente de que sus descubrimientos podrían dañar a la humanidad pero confiaba en que los beneficios serían mucho mayores. Casi noventa años después debemos aceptar que así ha sido (Brandan et al., 1998).

### 2.1.1. Interacción de la radiación con la materia

En la naturaleza existen elementos químicos que están constituidos por isótopos inestables. Estos isótopos experimentan una desintegración radioactiva. En la desintegración el núcleo puede emitir entre otras; partículas alfa (núcleos de helio), partículas beta (electrones) y radiación gamma (fotones).

Cada isótopo radiactivo tiene una probabilidad de desintegración por unidad de tiempo. Este valor es característico para cada isótopo. La velocidad de desintegración del radioisótopo en un tiempo dado ($t$) depende directamente del número de núcleos radiactivos ($N$) y de la constante de desintegración del radioisótopo ($\lambda$) (Júdez and López, 2013).

La expresión que rige la velocidad de desintegración es:

$$\boldsymbol{N = N_0 e^{-\lambda t}} \tag{2.1}$$

La naturaleza de la interacción de la radiación con la materia, es la ionización; es decir la formación de iones (átomos con carga, siendo estos neutros antes de la interacción) que son formados al sacar de sus orbitas a los electrones de los átomos de la materia. Es necesario el estudio de la interacción de la radiación ionizante con la materia, para entender los mecanismos o fenómenos por medio de los cuales ceden su energía al medio con el que interaccionan, para determinar cómo detectarlas, entendiendo el funcionamiento de los detectores para concebir los cambios que producen en los materiales (efectos biológicos) y como protegerse de ellas incluso calculando los espesores de blindajes adecuados.

La radiación posee energía, ya sea intrínseca como en el caso de la radiación electromagnética, o energía cinética en el caso de las radiaciones de partículas. La absorción de la radiación es el proceso por el cual se transfiere esta energía a los átomos del material absorbente. El hecho de decir que la radiación interacciona con la materia significa que ha sido absorbida o dispersada.

Los mecanismos de absorción de la radiación son de interés fundamental en el ejercicio de la seguridad radiológica, principalmente por las siguientes razones:

1. La absorción en los tejidos del cuerpo puede ocasionar un daño fisiológico.
2. La absorción es el principio en el cual se basa la detección.
3. El grado de absorción o tipo de interacción es un factor primario para determinar las necesidades del blindaje.

Las radiaciones pueden ser clasificadas de acuerdo con la naturaleza de su interacción con la materia de la siguiente manera: Partículas con carga eléctrica (Alfa, Beta, electrones rápidos y positrones) y Partículas sin carga eléctrica (Gamma, Rayos X, Neutrones, Neutrinos y Antineutrinos) (Huesca, 2015, Pedraza, 2013, Aramburu and Bisbal, 1996).

### 2.1.2. Partículas cargadas

**Las partículas cargadas pesadas (Alfas y protones).**

Interaccionan con la materia principalmente a través de fuerzas eléctricas de atracción entre su carga positiva y la carga negativa de los electrones orbitales del material absorbente. La energía transferida al electrón es a expensas de la energía de la partícula cargada incidente, por lo tanto la velocidad de esta decrece y se va frenando en cada interacción, ver Figura 2.1.

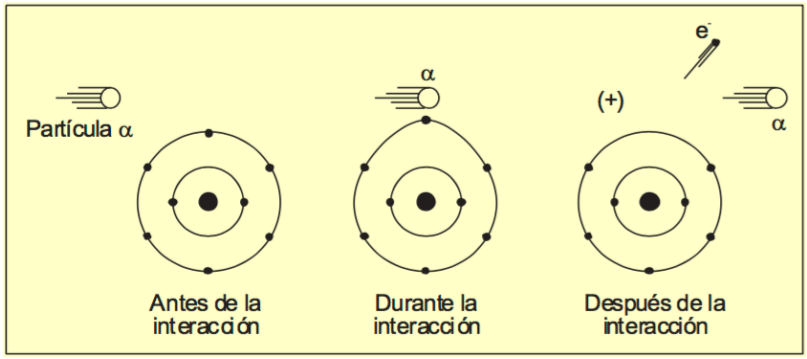


**Figura 2.1. Interacción de la partícula alfa con la materia (Huesca, 2015).**

El resultado de estas interacciones son átomos excitados y pares iónicos. Cada par iónico consta de un ion positivo y un correspondiente electrón libre. En algunas pocas interacciones el electrón libre resultante puede adquirir suficiente energía para crear más pares iónicos (secundarios) en el medio. Estos electrones energéticos son llamados electrones secundarios (Huesca, 2015, Glasstone and Sesonske, 1990).

**Las partículas cargadas ligeras (Betas – Electrones rápidos).**

Por ser los electrones rápidos, partículas con carga negativa, la interacción principal de estas energías, se lleva a cabo por repulsión con los campos eléctricos negativos de los electrones orbitales en el medio en que penetren sacándolos de sus orbitas, provocando así ionización y excitación en el material absorbente, observar Figura 2.2.

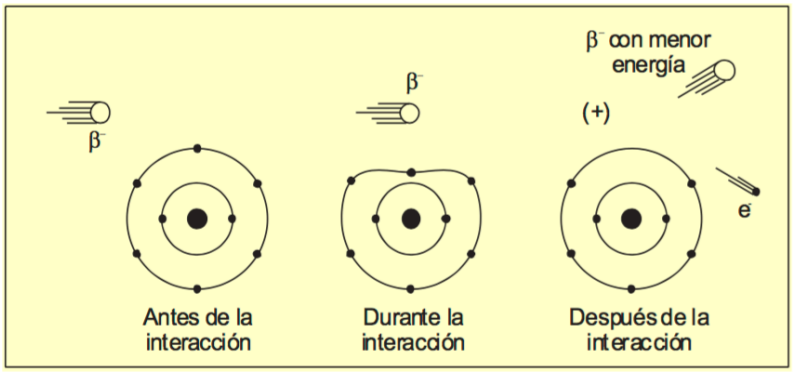


**Figura 2.2. Interacción de la partícula beta con la materia (Huesca, 2015).**

De acuerdo con la teoría electromagnética, cuando cualquier partícula cargada es acelerada, emite radiación electromagnética. Cuando electrones rápidos se aceleran por la interacción con los electrones orbitales se produce este efecto, obteniéndose rayos X de *Bremsstrahlung* (Huesca, 2015, Delgadillo, 2003).

La energía de un fotón de *Bremsstrahlung* corresponde a la energía cinética perdida por el electrón rápido cada vez que se produce el efecto, de esto se desprende que el espectro de energías es continuo (Figura 2.3). La producción de *Bremsstrahlung* puede ser apreciable para electrones rápidos, pero es totalmente despreciable para partículas pesadas.

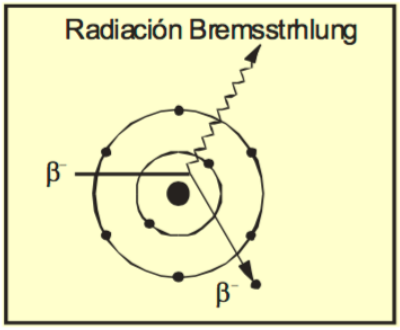


**Figura 2.3. Frenado de electrones - Radiación *Bremsstrahlung* (Huesca, 2015).**

La fracción de energía que se convierte en *Bremsstrahlung* aumenta con el incremento de la energía del electrón y con el incremento del número atómico del material absorbente. El *Bremsstrahlung* solo es significativo para materiales con alto número atómico. Ejemplo de esta radiación de frenado se tiene con los equipos generadores de rayos X, en donde el material con el que interaccionan los electrones generalmente es tungsteno, o una aleación de tungsteno (Huesca, 2015, Gyulassy and Wang, 1994).

### 2.1.3. **Fotones**

Al no poseer masa en reposo ni carga eléctrica la interacción con la materia no es por medio de atracción o repulsión eléctrica. Solo tres tipos de interacción de la radiación electromagnética con la materia tienen importancia en la detección de la radiación y en la protección radiológica, los cuales son: Efecto fotoeléctrico, Efecto Compton y Producción de pares.

**Efecto fotoeléctrico.**

El efecto fotoeléctrico, es la interacción de un fotón (gamma o X), que incide sobre los electrones de un átomo, el fotón desaparece totalmente y se desprende del átomo un electrón (partícula beta de la misma energía), es decir un fotoelectrón (Figura 2.4). El origen más probable del fotoelectrón son las capas electrónicas interiores, o sea K y L. El fotoelectrón resulta con una energía, Ec, dada por:

$$\boldsymbol{E_c = h\nu + E_e} \qquad (2.2)$$

Dónde: h = Constante de Planck ($4.1357 \times 10^{-15}$ *eV*.Seg), v = Frecuencia de la radiación incidente y *Ee* = Energía de enlace en la capa orbital o electrónica donde se origina el fotón gamma.

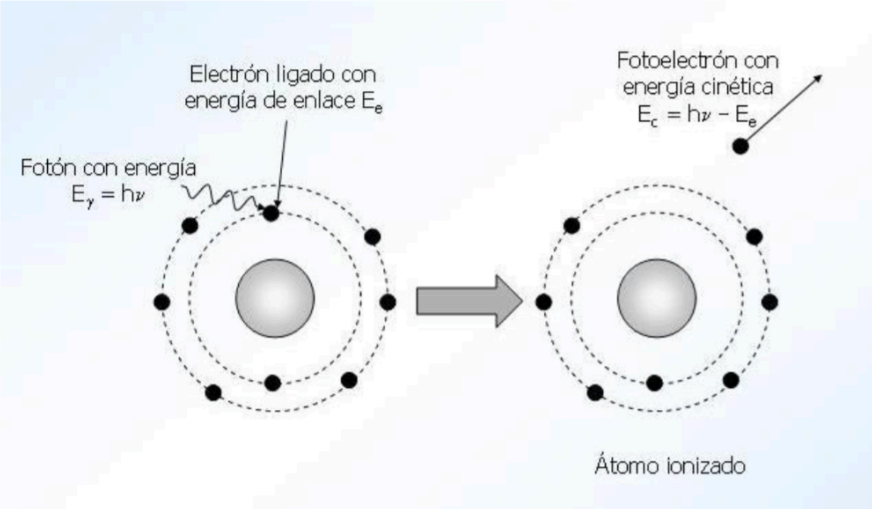


**Figura 2.4. Efecto fotoeléctrico (Huesca, 2015).**

A la salida del fotoelectrón, las capas electrónicas (K, L, M, N, O, P, Q del centro hacia afuera) del átomo que perdió el electrón se reacomodan. Sucede lo siguiente: Si un fotoelectrón se origina en la capa K, uno de la capa L bajará a la K emitiendo un fotón (Rayo X de fluorescencia) cuya energía es igual a la diferencia de la energía entre la capa L y la K, un electrón de la capa M bajará a la capa L emitiendo otro fotón de energía equivalente a la diferencia de la capa L y M, otro electrón bajará de la capa N a la M y así sucesivamente. El fotoelectrón cede su energía cinética al medio como cualquier electrón acelerado (Ver Figura 2.4). El proceso fotoeléctrico es el modo predominante de interacción para rayos gamma de baja energía y para materiales absorbentes de alto número atómico (Huesca, 2015, Tapia Júdez and Cuesta López, 2013).

**Efecto Compton.**

En el efecto *Compton* el rayo gamma o X interacciona con un electrón orbital cediendo solo parte de su energía a la vez que es desviado un ángulo $\theta$ (Ver Figura 2.5). Mientras mayor es el ángulo de desviación, mayor es la energía cedida al electrón. Esta energía cedida al electrón va desde casi cero a grandes fracciones de la energía del rayo gamma. La energía que no es cedida al electrón la conserva el rayo gamma desviado. La máxima transferencia de energía se produce cuando $\theta$ es igual a 180 grados.

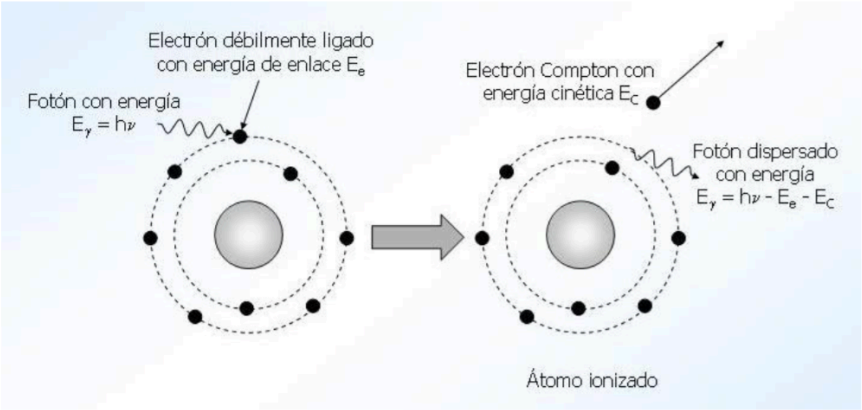


**Figura 2.5. Efecto Compton (Huesca, 2015).**

El efecto *Compton* es el mecanismo predominante de interacción en el rango de energías típicas que emiten los radioisótopos utilizados en reactores nucleares (Saglam et al., 2016, Del Monte et al., 2016, Huesca, 2015).

**Producción de pares.**

Si la energía del fotón (1.022 MeV) excede el doble, de la energía equivalente a la masa del electrón (0.511 MeV) entonces el proceso de producción de pares es energéticamente posible y por lo tanto este proceso es solo viable para rayos gamma o X de alta energía (Fotones que tengan una energía E ≥ 1.022 MeV). Ver Figura 2.6.

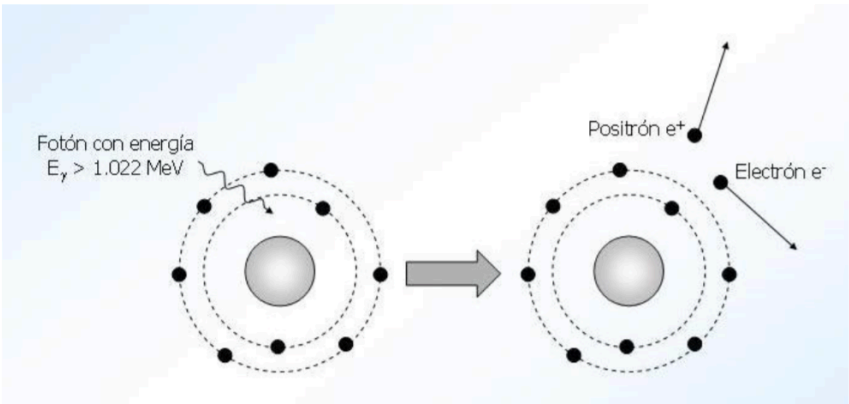


**Figura 2.6. Producción de pares (Huesca, 2015).**

En esta interacción, que se realiza con el campo eléctrico del núcleo del átomo, el fotón desaparece y da origen a un par (Electrón y positrón). El exceso de energía de 1.022 MeV, será repartido como energía cinética del electrón y el positrón. El electrón y el positrón ceden su energía a el medio, pero mientras el electrón se recombina con el medio, el positrón toma

un electrón del medio para producir una aniquilación de pares (Moreno et al., 2017, Huesca, 2015).

## 2.2. Detectores de radiación

La radiación interactúa con la materia, principalmente, mediante la ionización. De manera general, los detectores de radiación se aprovechan de este fenómeno para convertir esta radiación ionizante en señales eléctricas (Pellegrini et al., 2015).

### 2.2.1. Detectores de centelleo

La técnica de centelleo la utilizó Rutherford para el conteo de partículas alfa en sus experimentos de dispersión. Los principios básicos son los mismos que utilizaba Rutherford, aunque las técnicas han avanzado enormemente.

Una de las característica que tiene la radiación es la de emitir luz en ciertos materiales llamados centelladores, como el yoduro de sodio con algunas impurezas de talio, NaI(Tl). Es uno de los métodos más antiguos que permite realizar la detección de radiación al medir la luz generada en estos materiales. La luz generada es muy pequeña, de solo algunos fotones, por lo que la luz necesita convertirse en una señal eléctrica. Estos detectores por lo tanto, requieren de un dispositivo electrónico que amplifique la señal eléctrica, ya que también es muy pequeña; ese dispositivo se llama tubo fotomultiplicador que se lleva utilizando desde 1947 y desde entonces se ha venido mejorando y perfeccionando hasta ser capaces de detectar conteos muy elevados con tiempos muy pequeños, en la Figura 2.7 se muestra un tubo fotomultiplicador.

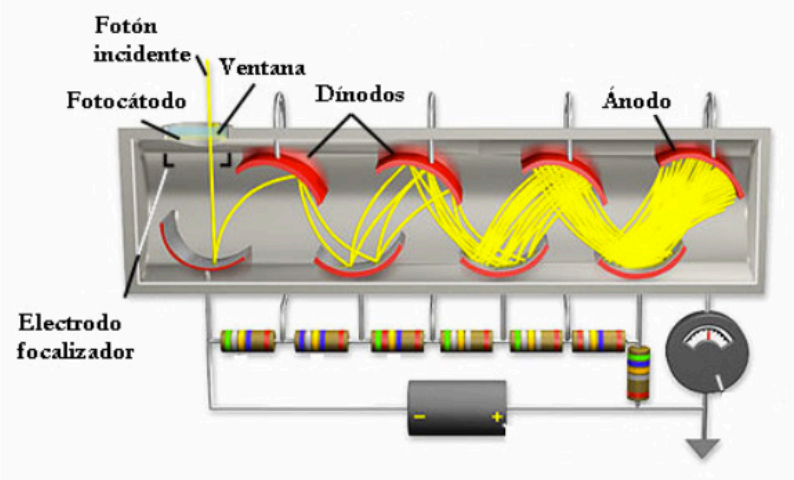


**Figura 2.7. Representación de un fotomultiplicador (Monedero and López, 2012).**

El tubo es un cilindro de vidrio al vacío que posee en uno de sus extremos un fotocátodo (placa sensible a la luz), generalmente formada por capas alternadas de antimonio y cesio, que emite electrones cuando recibe radiación luminosa (efecto fotoeléctrico). Mostrado en la Figura 2.7, los electrones desprendidos del cátodo se traslada hacia primer dinodo impulsados por la diferencia de voltaje aplicada entre este y el fotocátodo (-300 V); esto les da energía suficiente como para que al llegar al primer dinodo sean capaces de arrancar 2 o 3 electrones por cada electrón incidente. Ya que el siguiente dinodo tiene un potencial "más" positivo que el anterior, el proceso se repite produciendo una multiplicación de electrones, de manera que al ánodo llegan de $10^6$ a $10^8$ electrones por cada electrón inicial.

Esto da lugar a que la corriente formada por los electrones da una caída del voltaje del ánodo, lo que a su vez genera un pulso que pasa al preamplificador, al amplificador lineal y finalmente al escalador, esto para ser almacenado en la memoria de un analizador multicanal, para posteriormente reproducir el espectro.

Los detectores con un centellador de NaI(Tl) son empleados en las cámaras gamma para estudios de medicina nuclear. El germanato de bismuto, BGO, es empleado en nuevos detectores de centelleo como un importante aporte para la ciencia. Los detectores de centelleo se utilizan en aplicaciones donde se requiere tener una gran sensibilidad de detección. Se usan en monitores portátiles y en la calibración de dosis de materiales radiactivos para aplicación en pacientes de medicina nuclear. En la Tabla 2.1 se pueden observar algunos de los materiales centelladores y sus características (Ramirez, 2010, Hernández, 2015).

**Tabla 2.1. Principales sustancias centelladoras y sus características (Tsoulfanidis, 2013).**

| Material | Longitud de onda máxima de emisión (ns) | Eficiencia del centellador (%) | Constante de decaimiento (μs) | Densidad ($10^3$ kg/m$^3$) |
|---|---|---|---|---|
| NaI(Tl) | 410 | 100 | 0.23 | 3.67 |
| $CaF_2$(Eu) | 435 | 50 | 0.94 | 3.18 |
| CsI(Na) | 420 | 80 | 0.63 | 4.51 |
| CsI(Tl) | 565 | 45 | 1 | 4.51 |
| $Bi_4Ge_3O_{12}$ | 480 | 8 | 0.3 | 7.13 |
| $CdWO_4$ | 530 | 20 | 0.9 | 7.9 |

### 2.2.2. Detectores de semiconductor

En las últimas dos décadas los detectores de semiconductor o también conocidos de estado sólido se han utilizado ampliamente, permitiendo crear un número de portadores por pulso mucho mayor que con cualquier otro tipo de detectores (Venegas, 2015, Monedero and López, 2012, Molina et al., 2010).

Al depositar su energía una partícula en un detector semiconductor, se forman pares electrón-hueco en apenas picosegundos a lo largo de la trayectoria de la partícula. Debido a la presencia de un campo eléctrico, electrones y huecos se mueven en sentidos opuestos, lo que produce una corriente que permanece hasta que ambos portadores son recogidos a ambos lados del volumen activo (Monedero and López, 2012, Barrera et al., 2008).

Elementos como el Ge y el Si (semiconductores más ampliamente utilizados en la fabricación de detectores de estado sólido) que en su estado natural son considerados como aislantes eléctricos, tienen una estructura cristalina que puede ser descrita en términos de "bandas". Las energías de los electrones de valencia se encuentran tan cerca unos de otros que forman una banda (casi continua) de energías, conocida como banda de valencia. En el germanio puro (al igual que en el Si), existe una región de energías por encima de la banda de valencia, en la cual no hay niveles de energía permitidos. Esta zona se conoce como "gap" prohibido o banda prohibida. El tamaño del "gap" determina si se trata de un material conductor, semiconductor o aislante, en la Figura 2.8 se puede observar lo mencionado anteriormente (Venegas, 2015, Trujillo, 1998).

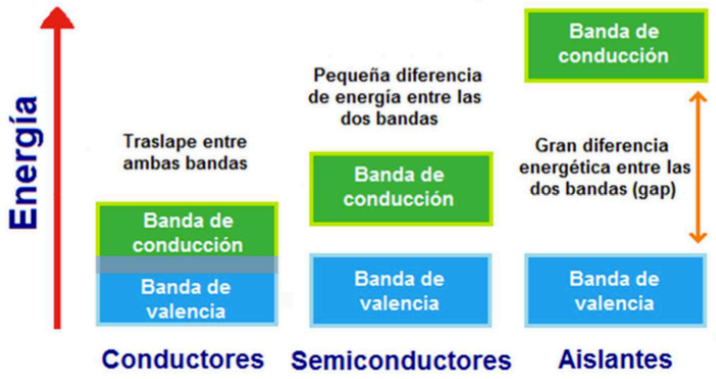


**Figura 2.8. Diagrama de bandas de energía (Venegas, 2015).**

El inconveniente de estos detectores es que algunos electrones de la banda de valencia pueden tener la suficiente energía térmica para excitarse y alcanzar a trasladarse a la banda de conducción. Ello obliga a operar a temperatura de licuefacción del nitrógeno para disminuir el número de estos electrones (González, 2014, Sánchez, 2013).

Los detectores de semiconductor tiene múltiples aplicaciones en la detección de la radiación y ventajas por encima de cualquier otro detector: el manejo es relativamente sencillo, presentan una buena uniformidad espacial, el área fotosensible tiene diferentes tamaños, formas y un reducido tamaño de su banda prohibida de energía o "gap".

El detector de centelleo es el principal "rival" de los detectores de semiconductor, sin embargo, para que los centelladores logren altas eficiencias de detección, requieren una gran energía (alrededor de 100 eV) para producir portadores de carga (pares electrón-hueco). Entonces, se producen menos portadores de carga para una energía de interacción dada, teniendo como resultando una resolución energética más pobre.

Por otro lado, los detectores de semiconductor requieren menos energía (cerca de 3 eV) para producir un portador de carga y por lo tanto, para una energía de interacción determinada se producirán más portadores de carga. Con un gran número de portadores de carga generados, la fluctuación estadística disminuye y la resolución en energía se mejora considerablemente (Venegas, 2015, Monedero and López, 2012, Molina et al., 2010).

## 2.3. Analizador multicanal

Los detectores de radiación producen señales o pulsos analógicos, estos pulsos son envidados a un preamplificador, pasando posteriormente por un amplificador y tratando estos impulsos eléctricos de tal manera que sean adecuados para el Analizador Multicanal o MCA por sus siglas en inglés (multichannel analyzer).

Estos pulsos tienen características especiales tales como: la duración del pulso, el valor máximo de éste, el tiempo promedio entre pulsos; siendo posible estudiar éstas señales mediante un Osciloscopio de alta velocidad. También es de interés el número de pulsos que se producen en un intervalo de tiempo; esto lo hace el MCA, el cual tiene el propósito de contar dichos pulsos, además de clasificarlos por amplitudes y almacenar las repeticiones de amplitudes en canales correspondientes (Martínez, 2015, Loza and Elias, 2010).

Por lo general el multicanal está compuesto por tres bloques principales que cumplen funciones específicas diferentes. En la Figura 2.9 se muestra un diagrama de los elementos principales de un analizador multicanal.

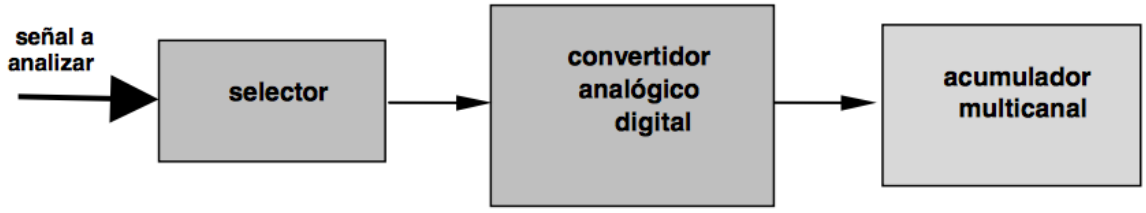


**Figura 2.9. Diagrama de bloques de Analizador multicanal (Sánchez, 2013).**

El primer bloque llamado *selector* permite el paso de señales que satisfacen algún criterio de aceptación, para su posterior análisis. Está formado por al menos un conjunto de analizadores monocanal de altura de pulsos de entrada y una compuerta lineal, cuya función es la de permitir el paso de sólo los pulsos de entrada cuya amplitud se encuentre dentro del rango de entrada del analizador.

El segundo elemento es el *convertidor analógico digital*, el cual clasifica las señales de entrada de acuerdo a su altura, generando un número proporcional a la misma. Esta información es guardada en el tercer de los bloques nombrado *acumulador multicanal* (memoria del multicanal) en forma de histograma. Esto constituye la salida del sistema y puede visualizarse en forma de espectro o mediante el trazado gráfico, vía algún otro

equipamiento externo. De este modo, a cada señal de voltaje de entrada se le asocia un canal correspondiente a su voltaje, que corresponde a la máxima altura de el pulso (Sánchez, 2013).

## 2.4. System on chip

Un sistema en chip o SoC por sus siglas en inglés (System on Chip) es un circuito integrado de silicio que implementa por completo la funcionalidad de un sistema, en lugar de utilizar varios chips físicos diferentes interconectados entre sí. Hablando de sistemas digitales, se puede combinar procesado, lógica de propósito específico, interfaces, memorias, decisiones, etc. En la Figura 2.10 se ejemplifica los componentes que podría tener un SoC (Álvarez, 2016).

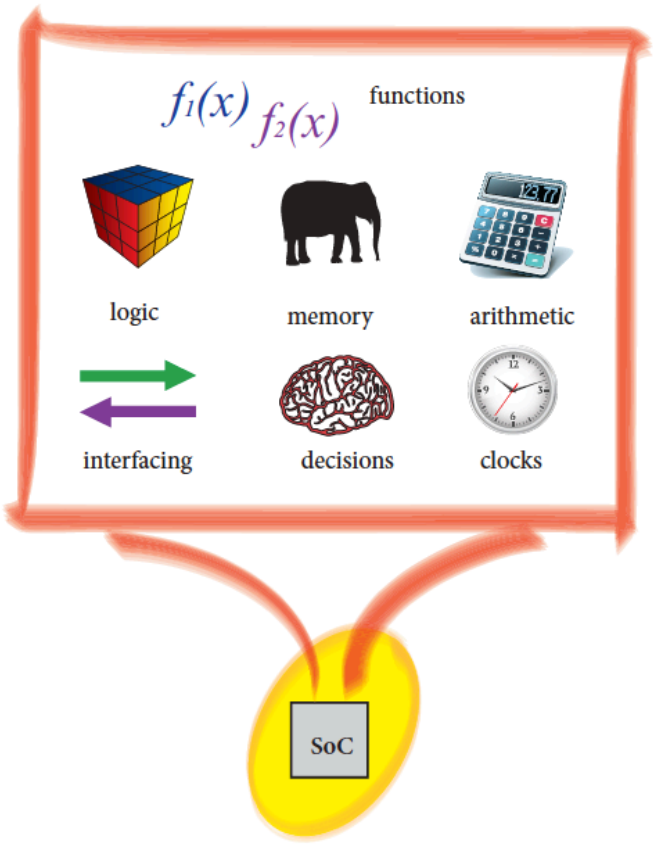


**Figura 2.10. Ejemplificación de System on chip (Crockett et al., 2014).**

Al disponer de una FPGA y un procesador en el mismo chip, que puede ser nombrado SoC basado en FPGA, ofrece la flexibilidad y escalabilidad de tener la lógica programable junto con el rendimiento y potencia de un procesador. Esto permite la implementación de

algoritmos en ambas partes, lo que es un uso muy común en el diseño de SoCs. Además, en comparación con los ASIC (Circuito Integrado para Aplicaciones Específicas), este tipo de SoC permite reducir considerablemente el tiempo de desarrollo y el costo (Pavón, 2015).

## 2.5. FPGA´s

La *FPGA* (*Field Programmable Gate Arrays*) es un dispositivo de hardware reconfigurable electrónicamente in situ (hacer código en lugar de electrónica) utilizando un lenguaje de programación especializado para la descripción de hardware, que su arquitectura se basa en arreglos de bloques de lógica digital cuya interconexión y funcionalidad puede ser programada, estos dispositivos constituyen la plataforma hardware ideal para explotar al máximo los diferentes niveles de paralelismo. Permitiendo describir y modelar hardware sobre su arquitectura. El tipo de hardware que se puede describir y modelar en un FPGA va desde compuertas lógicas, procesamiento de señales, procesamiento de imágenes, criptografía, hasta complejos circuitos que pueden ser reutilizados, entre otras.

En la Figura 2.11 se pueden observar los elementos básico en una *FPGA*; el bloque lógico configurable (*CLB*), las interconexiones, multiplexores, el manejador digital de reloj (DCM), boques de memoria y los bloques de entrada/salida (I/O) (Arce et al., 2016, Reyes, 2015, Ramirez et al., 2015, Colodro et al., 2012).

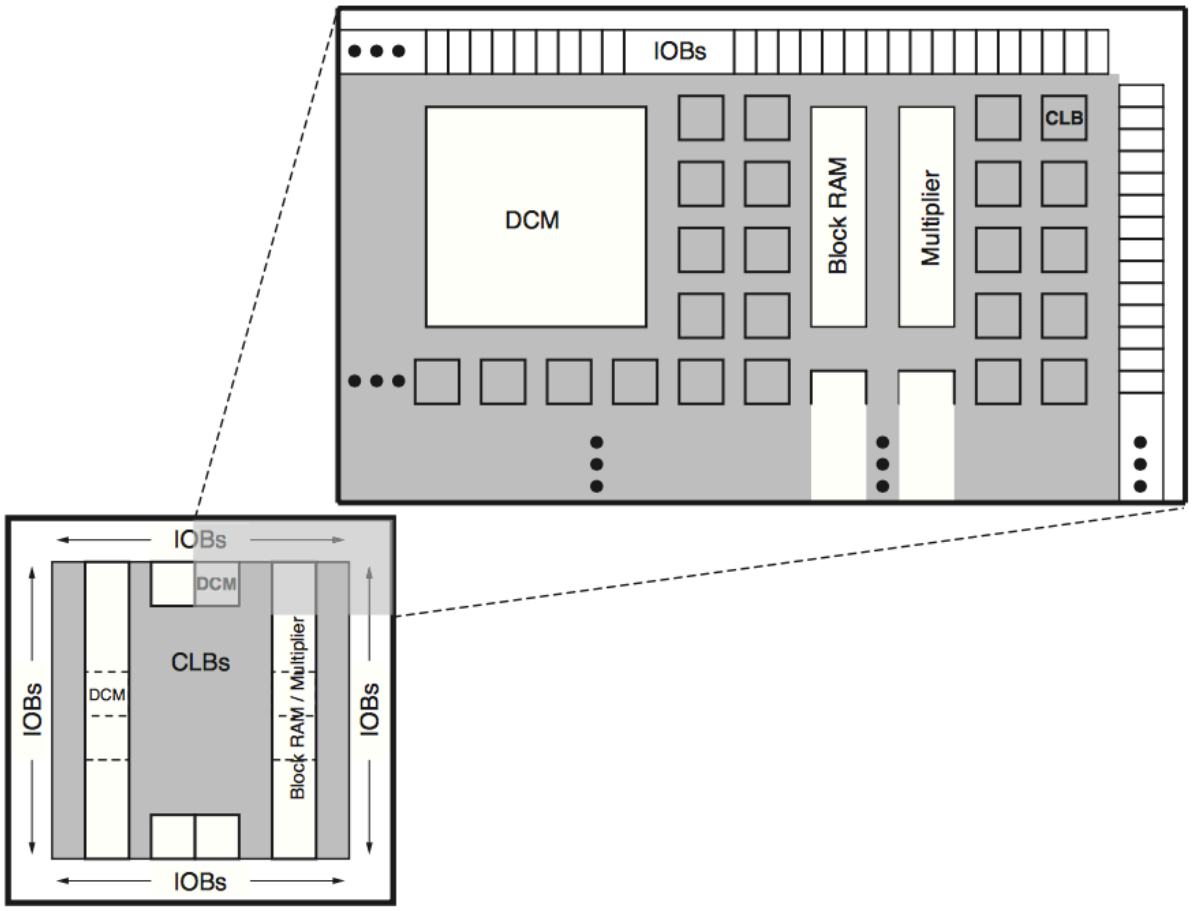


**Figura 2.11. Estructura de los bloques de una FPGA (Xilinx, 2011).**

La unidad lógica básica en un *FPGA* es el *CLB*, su número y características varía dependiendo el dispositivo. En general los CLB's consisten en cuatro segmentos, cada segmento está conformado por dos *Look Up Table (LUT)* para la implementación de lógica y dos más dedicadas al almacenamiento, que pueden ser usadas como *flip-flops* (dispositivos de dos estados, que sirven como memoria básica para las operaciones de lógica secuencial), cada una de las *LUT's* cuenta con cuatro entradas y una salida; también los *CLB´s* cuentan con multiplexores y componentes lógicos de aritmética y acarreo. Cada *CLB* puede configurarse para implementar lógica combinacional y registros de almacenamiento. Estos *CLB's* se unen para crear bloques lógicos más complejos (Arce et al., 2016, Reyes, 2015).

La tecnología ha tenido una evolución tal, que en un futuro las *FPGA's* podrían llegar a tener millones de *LUT's* dando la posibilidad de diseñar arquitecturas con muchos procesadores trabajando en paralelo, generando más lógica de circuitos, y a medida que la

complejidad incrementa, la lógica de circuitos entre los procesadores hace que los diseños *RTL* (Register File Level) tradicionales sean ineficientes (Arce et al., 2016).

### 2.5.1. Placa de desarrollo ZedBoard

La placa de desarrollo *ZedBoard*, proveniente de las siglas *Zynq Evaluation & Development Board*, es una placa de desarrollo de la familia de dispositivos *Zynq-7000 All Programmable SoC*. La *ZedBoard* es fruto de una asociación entre *Xilinx*, *Avnet* (el distribuidor) y *Digilent* (el fabricante de la placa). La placa y gran parte de sus distintos componentes se puede verse en la Figura 2.12 (Álvarez, 2016, Ortega, 2014).

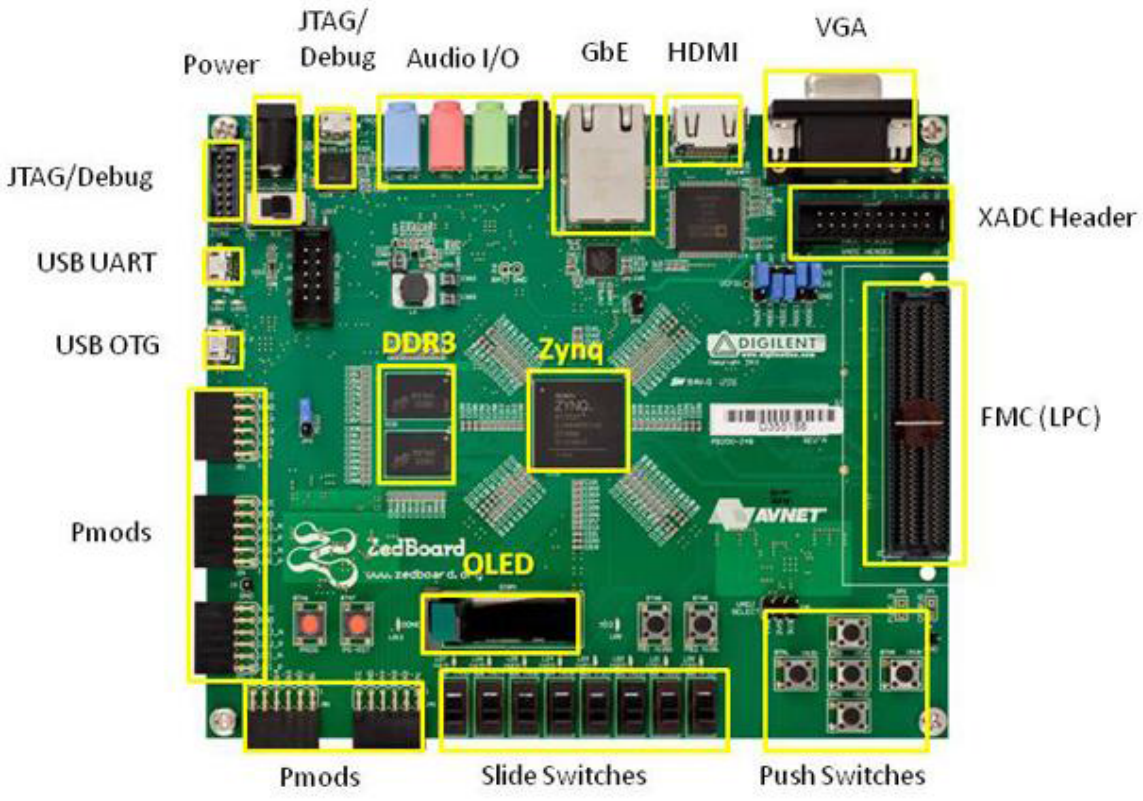


**Figura 2.12. Placa de desarrollo ZedBoard (Ortega, 2014).**

La ZedBoard cuenta con un gran número de interfaces y características que se enlistan a continuación:

- SoC: Xilinx Zynq XC7Z020-1CLG484C Zynq-7000 AP SoC
  - Configuración primaria = QSPI Flash

- Configuración auxiliares
  JTAG
  SD Card

- Memoria:
  - 512 MB DDR3 (128 x 32)
  - 256 Mb QSPI Flash
- Osciladores:
  - PS - 33.333 MHz
  - PL - 100 MHz
- Periféricos:
  - GPIO: 9 LEDs, 8 Interruptores, 7 botones
  - 10/100/1G Ethernet
  - USB-OTG 2.0
  - USB-JTAG
  - USB-UART 2.0 FS
  - Ranura para tarjeta SD
  - 5 Conectores PMOD para tarjetas de expansión (1 PS, 4 PL)
  - Conector XADC
  - Conector FMC
  - Botones de Reset para PS y PL
- Dysplay/ Audio
  - Salida HDMI
  - VGA (12-bit Color)
  - Display OLED 128x32
  - Audio entradas/salidas: Line In, Line Out, Micrófono y Salida de Auriculares
- Power
  - Switch encendido/apagado
  - 12V @ 5A AC/DC regulador (ZedBoard, 2014).

Las características mencionadas se ilustran en la Figura 2.13 donde se muestran las partes de la Zynq y que componentes pueden ser utilizados por la parte PL y PS. En la Figura

2.14 se observa los bancos de energía que proporciona la placa de desarrollo y la conexión de los componentes.

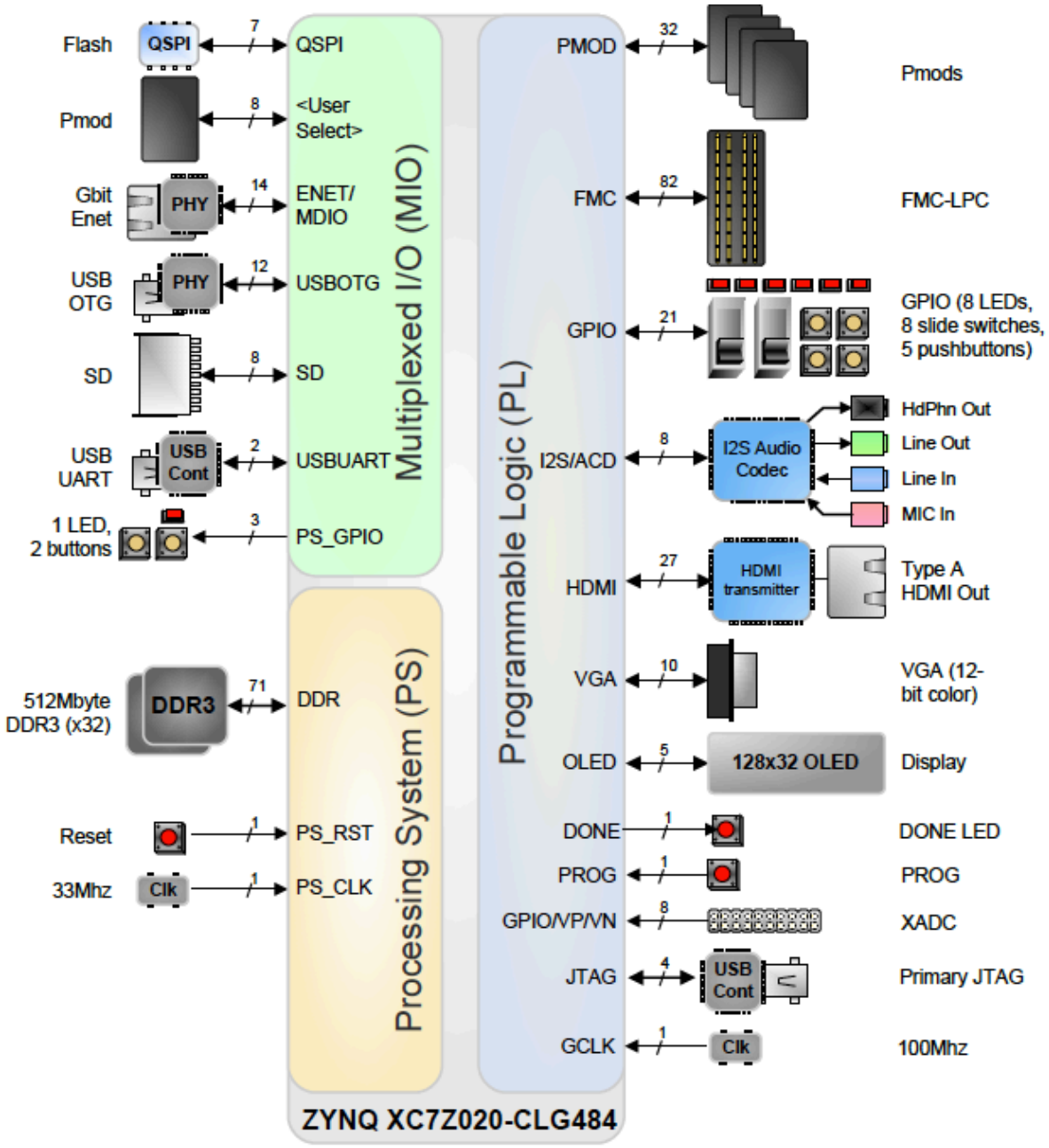


**Figura 2.13. Diagrama de bloques de ZedBoard (ZedBoard, 2014).**

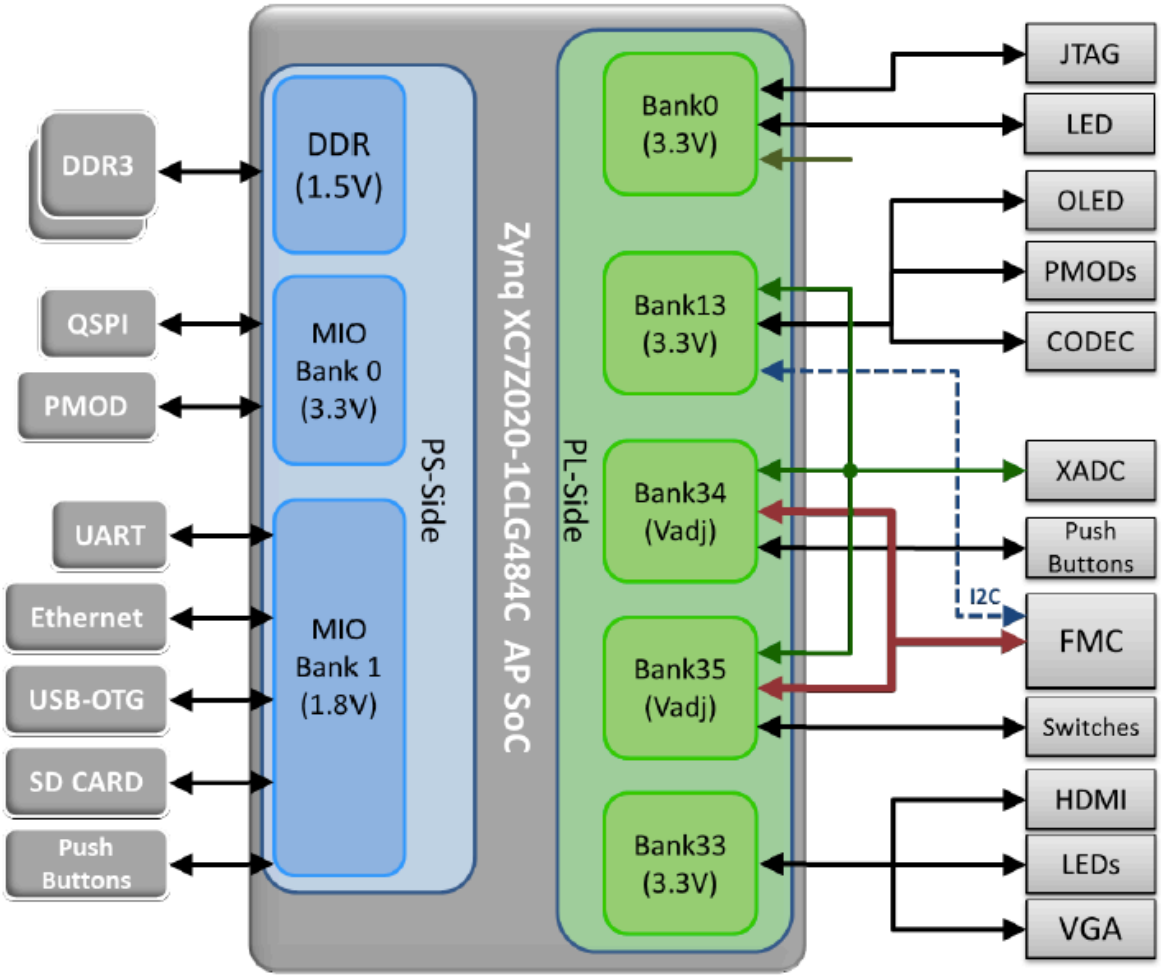


**Figura 2.14. Asignación de bancos de energía ZedBoard (Apu, 2016).**

### 2.5.2. Zynq-7000

El dispositivo Zynq-7000 All Programmable SoC de Xilinx, lleva el nombre de Zynq por el elemento químico zinc (Zn), ya que representa un elemento de procesamiento que se puede aplicar y utilizar a cualquier cosa, tanto para describir hardware como para programar software. Zynq es la plataforma ideal para infundir inteligencia a los sistemas embebidos de hoy en día, están diseñados para ser flexibles y formar un entorno convincente para una amplia variedad de aplicaciones, de la misma manera que el elemento químico zinc se puede mezclar con otros metales para formar aleaciones deseables con diferentes propiedades.

La placa de desarrollo Zynq está comprendida por dos partes principales integradas en un único chip: un Sistema de procesamiento (PS) basado en un microprocesador Dual-Core

ARM Cortex A9, y la Lógica Programable (PL) basada en la arquitectura de la FPGA de la serie 7 de Xilinx. La parte de la PL es ideal para implementar lógica de alta velocidad (para acelerar tareas con una alta carga computacional, como sistemas aritméticos, procesamiento de video en tiempo real, etc.) y el PS soporta sistemas operativos completos y rutinas software, de manera que la funcionalidad total de cualquier sistema puede ser particionada adecuadamente entre hardware y software. En la Figura 2.15 se muestra una representación de una Zynq con sus partes PS y PL (Álvarez, 2016, Ortega, 2014, Crockett et al., 2014).

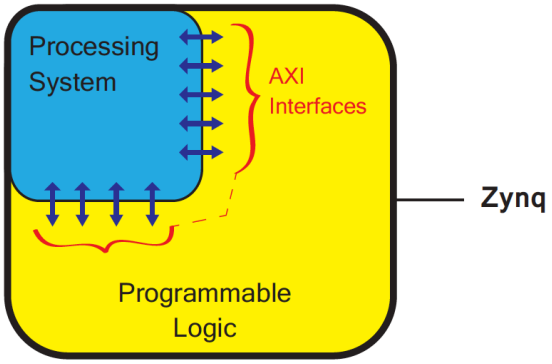


**Figura 2.15. Modelo simplificado de un Zynq (Crockett et al., 2014).**

Existe un sinfín de aplicaciones para la Zynq entre las que se incluyen las comunicaciones cableadas y wireless, automoción, procesamiento de imagen y video, computación de altas prestaciones, medicina, control industrial, procesamiento de datos, decisiones en tiempo real, etc. Una de las características de Zynq es que se puede cortar la energía a la PL para reducir el consumo de potencia, y por otra parte, se puede reducir la frecuencia de los relojes de la PS o incluso desactivarlos para reducir el consumo. La arquitectura puede ser empleada para satisfacer las demandas de aplicaciones con requerimientos tanto de computación paralela de alta velocidad como de funcionalidad secuencial con uso intensivo del procesador *hardCore* que logra un rendimiento excelente a diferencia de los procesadores softCore (microBlaze, niosll). El All Programmable system on chip Zynq está compuesto por varios elementos de alto rendimiento como se muestra en la Figura 2.16 (Álvarez, 2016, Pavón, 2015, Galindo, 2015).

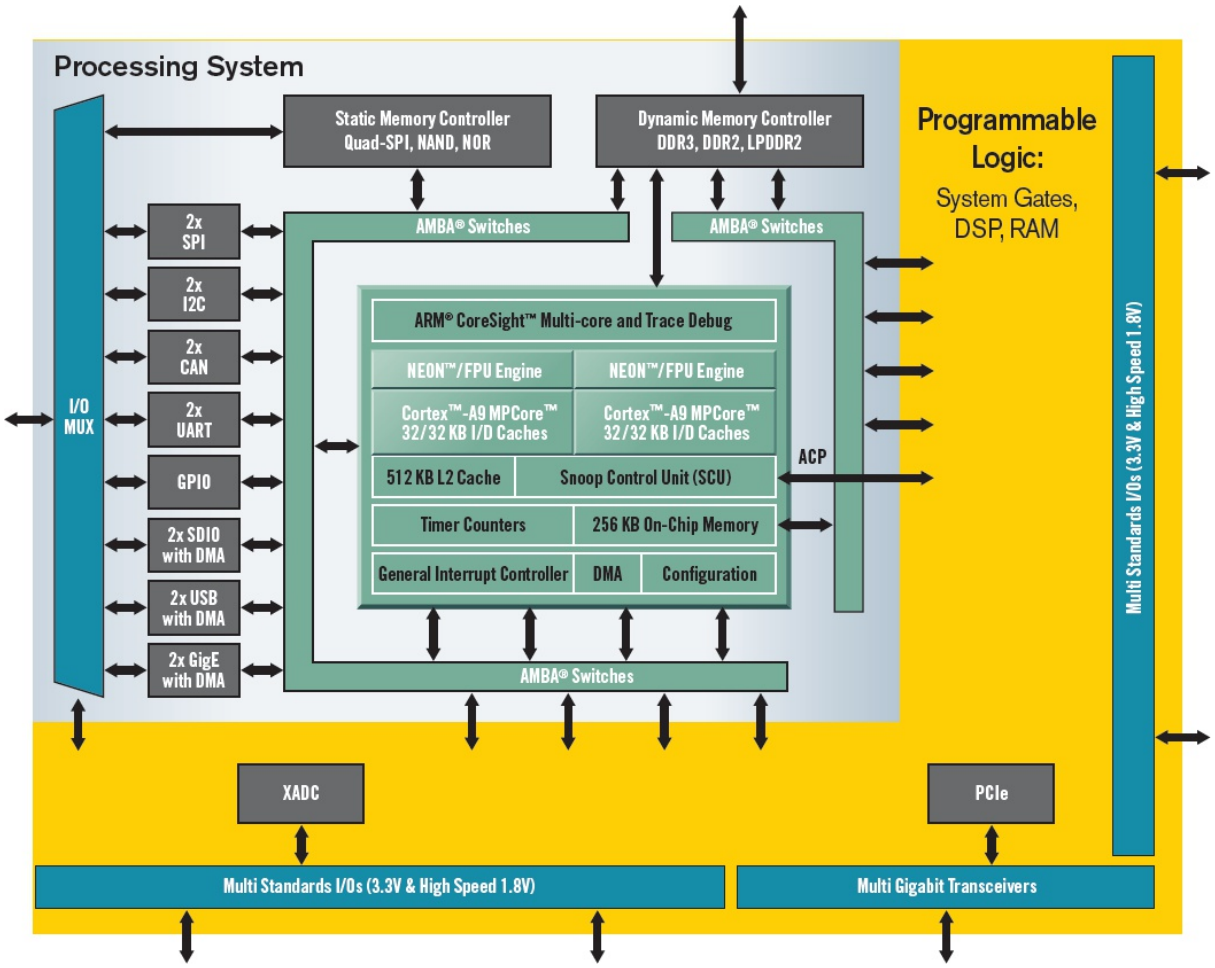


**Figura 2.16. Arquitectura del Zynq-7000 (Getman, 2011).**

### 2.5.2.1. **Sistema de procesamiento (PS)**

El PS es la parte que controla el sistema de procesamiento de todos los dispositivos Zynq, es un microprocesador Dual-Core ARM Cortex A9 que puede operar a una frecuencia de reloj de hasta 1 GHz, esto dependiendo del modelo de la Zynq. Es un procesador *"hard"*, es decir, existe dentro del *SoC* como un elemento de silicio dedicado y optimizado (en contraposición a los procesadores *"soft"* que se implementan utilizando la lógica programable de la *FPGA*, como el Xilinx MicroBlaze).

La Application Processing Unit (APU) está formada por los dos núcleos en el procesador ARM, cada uno con una serie de elementos adicionales: un coprocesador NEON y una unidad de punto flotante (FPU), una MMU (Memory Management Unit) y memoria cache L1 y L2.

Además de la APU, el sistema de procesamiento incluye interfaces para periféricos de I/O (USB, GigE, UART, I2C, SPI...), memoria cache, interfaces de memoria, hardware de interconexión y circuitos para generar frecuencias de procesamiento. En la Figura 2.17 se observa la organización de la APU (Álvarez, 2016, Sánchez, 2014).

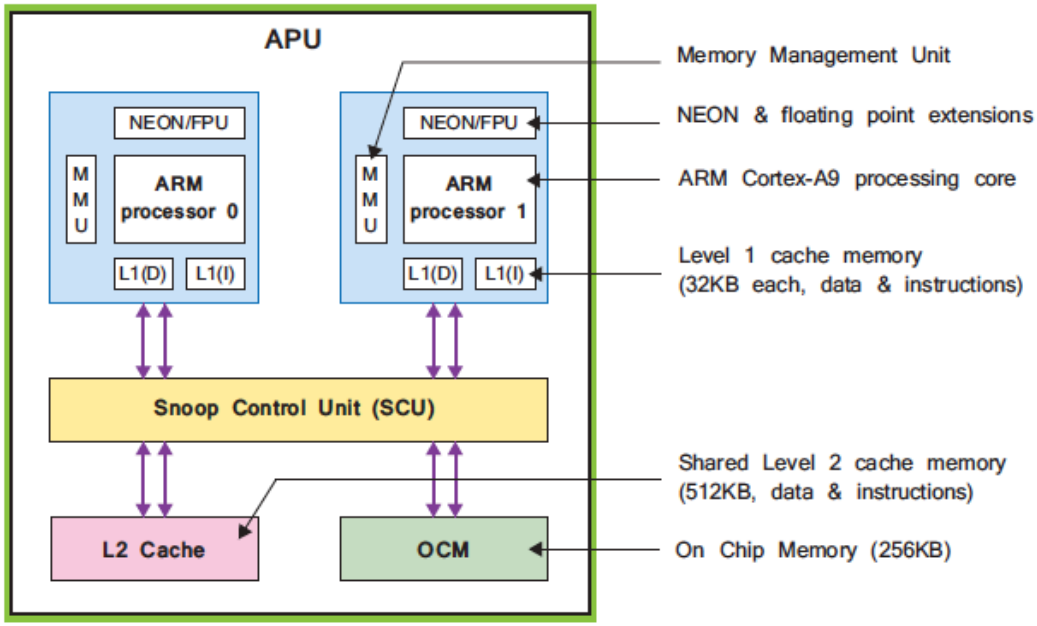


**Figura 2.17. Diagrama de bloques de APU (Crockett et al., 2014).**

### 2.5.2.2. **Lógica programable (PL)**

La lógica programable de la Zynq no puede iniciar antes que el sistema de procesamiento, la lógica está basada en la arquitectura de la FPGA Artix-7 y Kintex-7 de Xilinx. Esta predominantemente compuesta de *CLB's* (*Configurable Logic Blocks*), que contienen dos *slices* cada uno. Los *slices* están formados por 4 LUT's, y 8 Flip-Flops adicionados de lógica. Para proporcionar una interfaz entre la lógica y los pines físicos del dispositivo, existen IOB's (Input/Output Blocks) alrededor del perímetro de la PL.

Adicionalmente a la lógica general, hay dos tipos de recursos de propósito especial: memorias Block RAM dedicadas de 36 kB o 18 kB (capaces de implementar RAM's, ROM's y bufferes FIFO) y DSP's (Digital Signal Processors) de tipo DSP48E1 para aritmética de alta velocidad.

La PL recibe cuatro entradas separadas de reloj desde el PS y, adicionalmente, tiene la posibilidad de generar y distribuir sus propias señales de reloj independientemente del PS (Álvarez, 2016, Dobai and Sekanina, 2013).

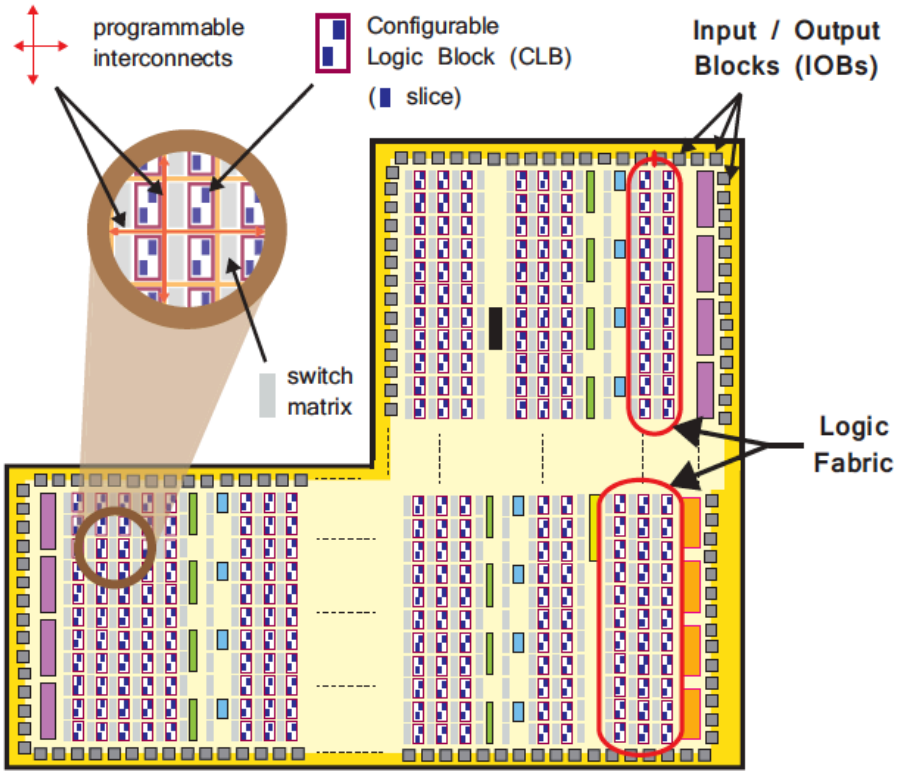


**Figura 2.18. Constitución de la parte lógica programable (Crockett et al., 2014).**

### 2.5.2.3. **Interfaces de comunicación PL-PS**

Las interfaces de comunicación entre la PS y la PL están basadas en buses de comunicaciones AXI (Advanced eXtensible Interface), y permiten alcanzar velocidades de transmisión de hasta 100 Gbps a una baja latencia. Una de las ventajas que ofrece la interfaz AXI es la posibilidad de realizar accesos rápidos a la memoria RAM DDR3 de la placa desde la PL. Este protocolo facilita y mejora la productividad en el proceso de creación de una arquitectura SoC, ya que existe una gran cantidad de IP Cores disponibles que la utilizan. Un IP Core es un bloque que constituye un subsistema y proporciona cierta funcionalidad (Álvarez, 2016, Agudelo and Valderrama, 2016, Pavón, 2015, Sosa, 2014).

Estas conexiones de gran ancho de banda son buses de comunicaciones diseñado por ARM, que constituye la cuarta generación de la especificación AMBA (Advanced

Microcontroller Bus Architecture), cuya objetivo es la interconexión de bloques funcionales en un SoC.

Las herramientas de desarrollo de Xilinx ha adoptado este protocolo para facilitar la creación de diseños SoC. El uso de estos buses proporciona varios beneficios, entre ellos una mejora en la productividad del diseño.

Existen tres tipos de interfaces AXI:

- AXI4 o AXI4-Full: Para realizar transferencias mapeadas (métodos de instanciar entradas con salidas entre los periféricos) en memoria con un rendimiento alto, permitiendo enviar ráfagas de datos en una misma transferencia.
- AXI4-Lite: Se trata de una simplificación del AXI4 para transferencias mapeadas en memoria, donde tan solo se pueden realizar transferencias simples, es decir, no permite el envío de ráfagas.
- AXI4-Stream: Para transferencias tipo streaming (retrasmisión en difusión continua ) de alta velocidad donde un maestro transfiere datos a un esclavo (Pavón, 2015, Sosa, 2014).

Una representación de todos los componentes de la Zynq, como son la parte lógica programable (PL), el sistema de procesamiento (PS), la unidad se procesamiento de aplicaciones (APU) y la conexión AXI se puede observar en Figura 2.19.

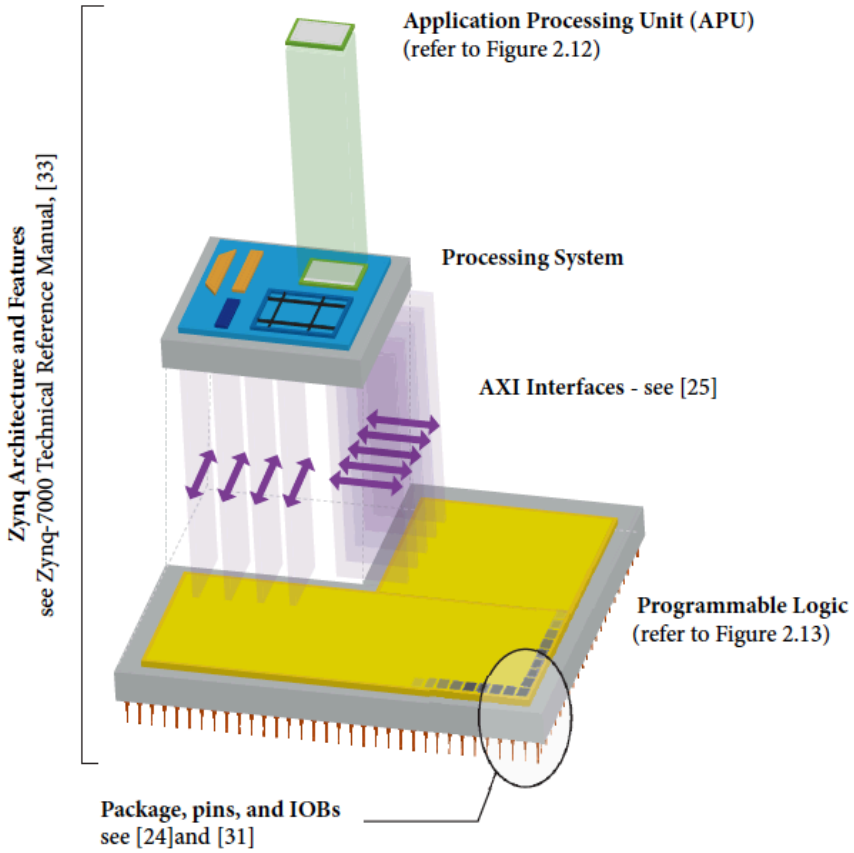


**Figura 2.19. Arquitectura general del Zynq (Crockett et al., 2014).**

### 2.5.3. Lenguaje de descripción de hardware VHDL

El lenguaje para descripción de hardware VHDL (*VHSIC* (*Very High Speed Integrated Circuit*) *Hardware Description Lenguaje*), describe circuitos lógico para la implementación en *SIC's* (*Specific Integrated Circuits*), *FPGA's* y circuitos digitales convencionales, de acuerdo a su función, al comportamiento del flujo en sus datos o bien a su estructura. Aunque VHDL no fue diseñado para su implementación en lenguajes de propósito general, es posible diseñar algoritmos con este lenguaje. La mayoría de su sintaxis deriva del lenguaje Ada (Ashenden, 2010, Bhasker, 1999).

VHDL fue creado para satisfacer numerosas necesidades en el proceso de diseño. Es posible realizar la especificación de las funciones para los diseños mediante el uso de un lenguaje de programación pero sobre todo posee la capacidad de simular el diseño antes de su fabricación, dando así la posibilidad de comparar alternativas y realizar pruebas correctivas

sin el retraso y el costo que implica la construcción de prototipos de hardware. Existen una gran cantidad de plataformas de software que permiten describir hardware, entre estas se encuentra *ACTIVE HDL*, el cual es un software que maneja lenguaje *VHDL* y *Verilog* que permite realizar simulaciones de diversos sistemas y es un ambiente completo y totalmente integrado para el diseño y la verificación de proyectos digitales (Roth Jr and John, 2016, Hernández et al., 2015).

## 2.6. Vivado desing suite

Xilinx ha desarrollado una suite llamada Vivado Design Suite para diseñar y trabajar con dispositivos tales como FPGA y SoC's para las familias de la serie 7 de Xilinx: Ultrascale, Virtex-7, Kintex-7, Artix-7, y Zynq -7000. Además permite implementarlos mediante la interconexión de IP Cores. Esta suite que sustituye a las herramientas de la suite Xilinx ISE (ChipScope, XST, Coregen, PlanAhead, Simulator, etc), las incorpora en una misma interfaz gráfica, permitiendo mejorar la productividad en los diseños, ya que el cambio de una herramienta a otra se realiza rápidamente y dentro de la misma plataforma de desarrollo, además Vivado proporciona una interfaz gráfica que permite la creación de diseños complejos de forma visual. Se pueden crear arquitecturas SoC con IP Integrator mediante la configuración e interconexión de distintos bloques funcionales (IP's), tanto los creados por Xilinx, como los diseñados por el usuario (Álvarez, 2016, Pavón, 2015).

En esta herramienta se puede apreciar el desarrollo de la plataforma, porque el sistema está constituido por bloques para ver las conexiones entre los mismos y detectar errores. En la Figura 2.20 se puede visualizar el entorno de desarrollo de Vivado así como la interconexión de los bloques.

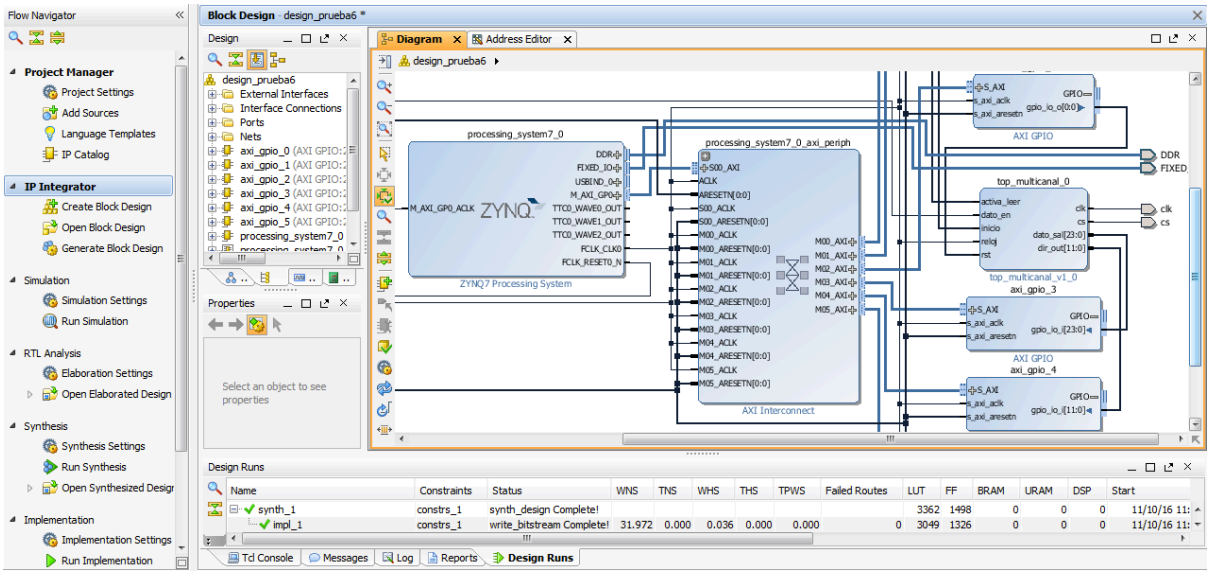


**Figura 2.20. Entorno de desarrollo de Vivado.**

Dicha herramienta permite emplear lenguajes de descripción hardware (HDL) para realizar diferentes diseños mediante la integración de bloques de propiedad intelectual (IP), así como también funciones para la descripción de bloques de hardware mediante código C o C++, Síntesis de Alto Nivel (HLS) y simulación (González, 2016, García, 2016).

La plataforma de desarrollo Vivado ofrece los siguientes componentes:

- Vivado: es un IDE (Entorno de Desarrollo Integrado) para diseñar y sintetizar el hardware que se implementa en la lógica programable de las FPGA's o del SoC Zynq. Además de trabajar con diseños RTL (VHDL/Verilog), cuenta con la posibilidad de desarrollar a alto nivel, formando e integrando bloques IPcore, lo que aumenta la posibilidad de reutilizar diseños antes creados.
- Xilinx SDK (Software Development Kit): es un IDE para el diseño de software basado en Eclipse que soporta procesadores embebidos en los dispositivos de Xilinx, que cuenta con compilador de GNU (gcc y g++) para desarrollar en C y C++ para procesadores ARM, además tiene herramientas para la depuración (Debug).
- Vivado HLS (High Level Synthesis): es un IDE para acelerar la creación de bloques IP hardware a partir de especiaciones de alto nivel en C, C++ o System C sin necesidad de describir el diseño RTL (VHDL/Verilog), siendo este generado automáticamente por la herramienta (síntesis de alto nivel) (Álvarez, 2016).

## 2.7. LabVIEW

LabVIEW (Laboratory Virtual Instrument Engineering Workbench) es un ambiente de desarrollo gráfico, diseñado por *National InstrumentTM Corporation*, con funciones integradas para realizar adquisición de datos, control de instrumentos, análisis de mediciones, presentaciones de resultados y su principal característica es la de crear aplicaciones complejas. LabVIEW da la flexibilidad de un poderoso ambiente de programación, que es difícil con lenguajes de programación de alto nivel tradicionales (Saltos et al., 2014, Jimenez et al., 2014).

LabVIEW permite diseñar interfaces de usuario mediante una consola interactiva basada en software. También se puede diseñar especificando las funciones del sistema, su diagrama de bloques o una notación de diseño de ingeniería. LabVIEW es compatible con herramientas de desarrollo similares y puede trabajar con programas de otra área de aplicación (Matlab, Wolfram Mathematica, entre otros).

Tiene la ventaja de una fácil integración con hardware, específicamente con tarjetas de medición, adquisición y procesamiento de datos (incluyendo adquisición de imágenes). Además LabVIEW trabaja con más de 1000 librerías de instrumentos de cientos de fabricantes.

Los programas creados en LabVIEW se guardan como instrumentos virtuales en archivos con la extensión VI. También relacionado con este concepto se da nombre a las dos ventanas principales: un instrumento real tiene un panel frontal donde están los botones, pantallas, etc. y otra donde se encuentra la circuitería interna. En LabVIEW estas dos partes reciben el nombre de panel frontal y diagrama de bloques respectivamente (Tobías et al., 2016, Saltos et al., 2014).

El panel frontal, es la parte con la que interactúa usuario (suele tener fondo gris), en contra parte el diagrama de bloques, es donde se realiza la programación (suele tener fondo blanco), estas dos ventanas principales se ilustran en la Figura 2.21.

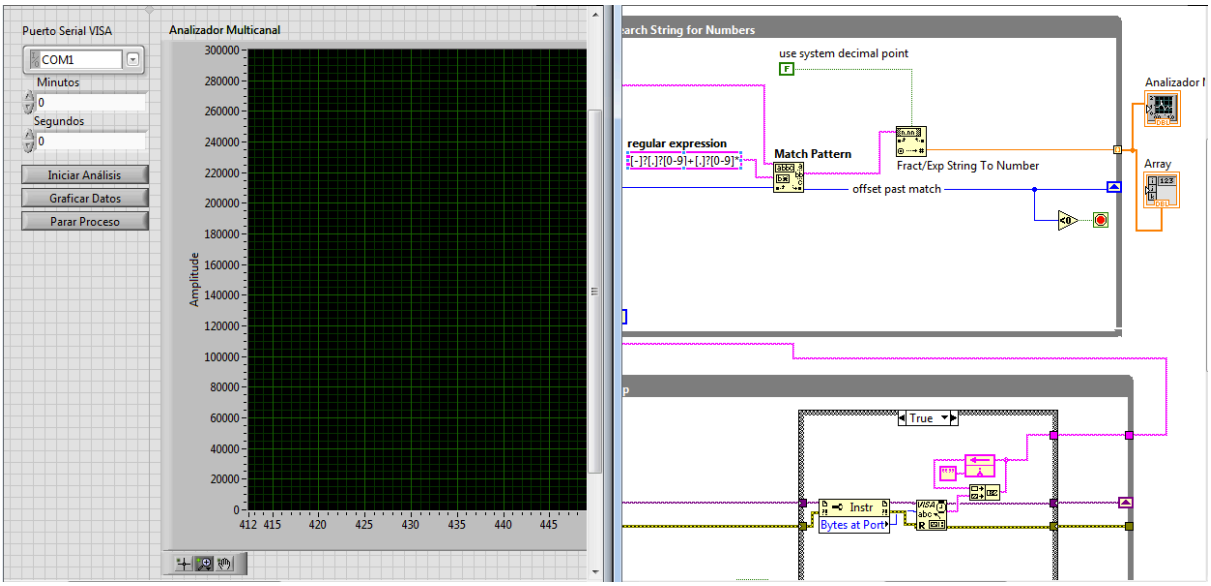


**Figura 2.21. Panel frontal y diagrama de bloques de LabVIEW.**

Esta plataforma ofrece la opción de seleccionar mecanismos diferentes para la transferencia de datos, entre ellos los incluidos en el protocolo de transferencia VISA, así como los procesos que se comunican a través de objetivos (Tobías et al., 2016).

### 2.7.1. Protocolo de comunicación NI-VISA

El software NI-VISA de la compañía National Instruments permite la conexión entre el instrumento virtual que se diseñe en LabVIEW y diferentes dispositivos que comprenden interfaces *GPIB*, *VXI*, *PXI*, serial (RS232/RS485), *Ethernet/LXI*, interfaces *USB*, *IEEE* 1394 y transductores, incluyendo acelerómetros, sensores de temperatura y sensores ultrasónicos de distancia. Proporciona la interfaz de programación entre el hardware y los entornos de desarrollo de aplicaciones como NI-LabVIEW, LabWindowsTM/CVI y Measurement Studio para Microsoft Visual Studio (Ceballos et al., 2015, Goñi Esparza, 2015, Julián Moreno et al., 2014, Mora Miranda, 2016).

# Capítulo 3. Materiales y métodos

En este capítulo se abordara tres secciones la primera es la descripción de hardware del dispositivo del analizador multicanal mediante un proyecto creado en Vivado, detallando la arquitectura de cada uno de los componentes descritos en VHDL que se proponen para el analizador, después se explica la sección de software que es una aplicación programada en lenguaje C y por último se detallan los componentes del instrumento virtual para el manejo, control y visualización de los datos, estas tres secciones se pueden ver ilustradas en la Figura 3.1.

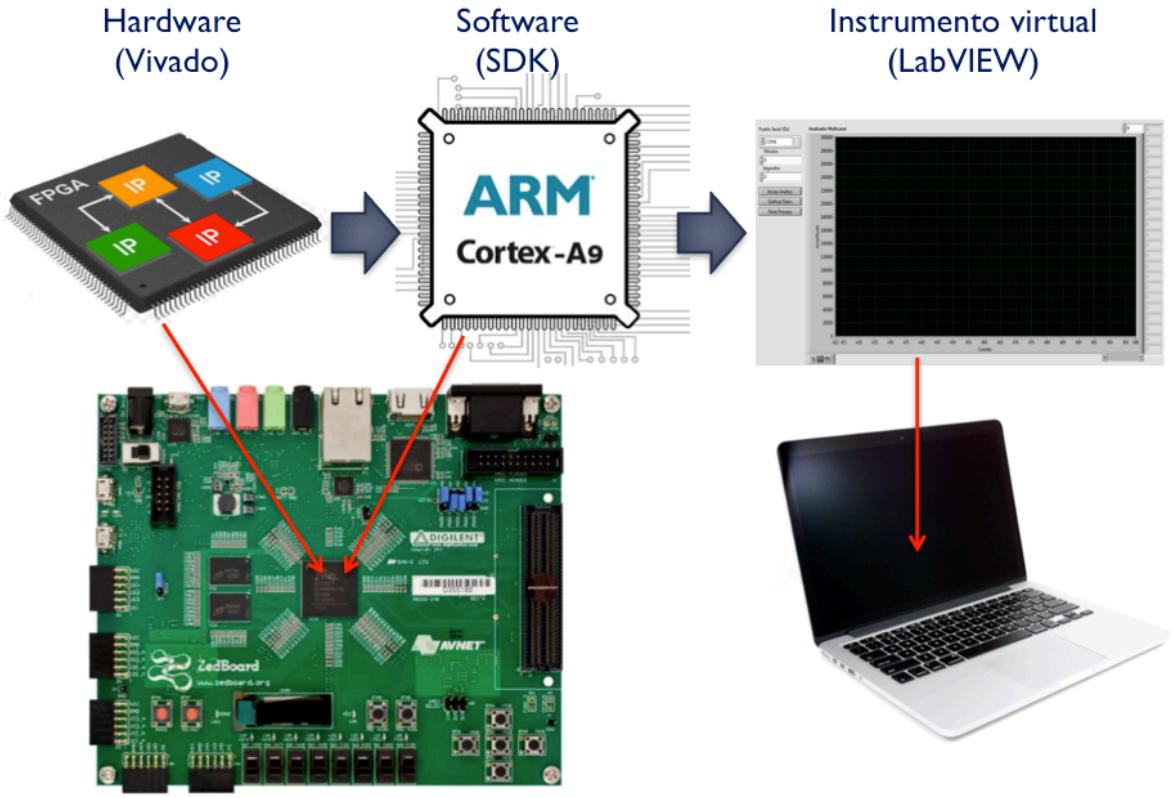


**Figura 3.1 Diagrama de flujo del analizador multicanal.**

## 3.1. Análisis y adecuación de las señales del detector

Para el análisis y adecuación de las señales provenientes del detector se utilizó un conversor analógico - digital de la compañía *Analog Devices,* este acoplado sobre un Pmod (Modulo periférico) desarrollado por *Digilent* ver Figura 3.2*,* Con una frecuencia de muestreo

de hasta un millón de muestras por segundo, tiene dos canales con muestreo paralelo con una resolución de 12 bits cada uno, con un intervalo de operación variable que tiene un rango de 0 Volts al voltaje de alimentación (VCC), el cual puede estar entre 2.35 y 5.25 Volts, que en este caso será de 0 a 3.3 Volts, este conversor se observa en la (Ada Pmod Xilinx et al., 2011). En la Figura 3.3 se muestra el circuito de interconexión del conversor analógico digital (Digilent, 2016).

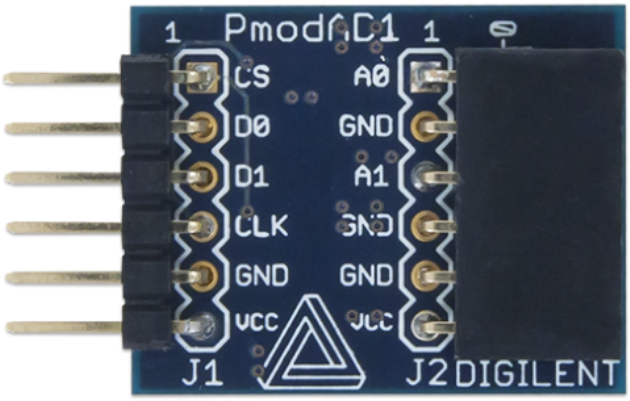


**Figura 3.2 Pmod del ADC (Digilent, 2016).**

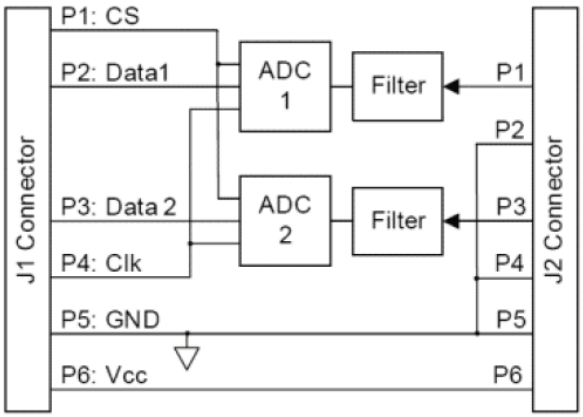


**Figura 3.3 Circuito del ADC (Digilent, 2016).**

El conversor tiene dos terminales *J1* y *J2* cada una con 6 puertos, *J1* conformada por los puestos *CS*, *D0*, *D1*, *CLK*, *GND* y *VCC* y *J2* constituida por *A0*, *A1*, *VCC* y 3 *GND*. En la Tabla 3.1 se explica cada uno de los puertos del conversor analógico digital.

**Tabla 3.1 Funcionamiento de los puertos del Pmod (Digilent, 2016).**

| Descripción de puertos | | | | | |
|---|---|---|---|---|---|
| Terminal J1 | | | Terminal J2 | | |
| **Nombre** | **Sentido** | **Función** | **Nombre** | **Sentido** | **Función** |
| CS | Entrada | Activa ADC | A0 | Entrada | Señal a convertir canal 1 |
| D0 | Salida | Canal convertido 1 | GND | Salida | Tierra asociada a canal 1 |
| D1 | Salida | Canal convertido 2 | A1 | Entrada | Señal a convertir canal 2 |
| CLK | Entrada | Reloj | GND | Salida | Tierra asociada a canal 2 |
| GND | Entrada | Tierra | GND | Salida | Tierra |
| VCC | Entrada | Alimentación (2.35 a 5.25 Volts) | VCC | Salida | Auxiliar de voltaje |

Las señales provenientes del detector de radiación son acopladas en la terminal *J2* en los puertos *A0* y *GND* que corresponden al canal 1, para que dichas señales sean convertidas de un estado analógico a un estado digital y la terminal *J1* se conecta a la placa de desarrollo para de ella recibir la señal de activar el *ADC* por el puerto *CS* además, por el puerto D0 puede enviar los datos ya convertidos del canal 1 asimismo, por el puerto *CLK* recibe el reloj a la frecuencia indicada que en este caso seria de 20 MHz, el puerto GND se conecta a la tierra de la placa de desarrollo y por último el Pmod se alimenta con un voltaje de 3.3 Volts por el puerto VCC. Las especificaciones técnicas de funcionamiento del conversor analógico digital se muestran en la Figura 3.4.

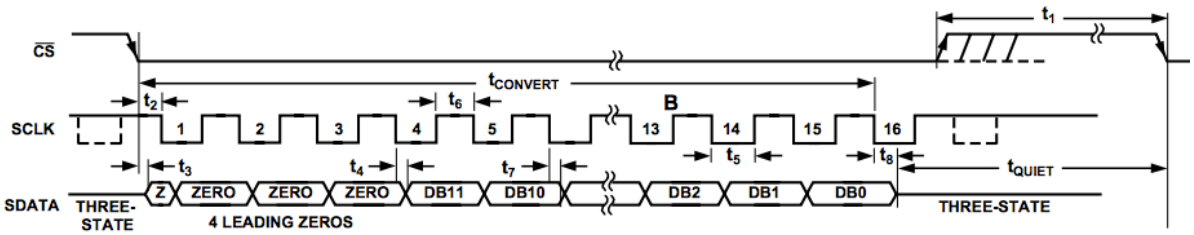


**Figura 3.4. Interface serial del conversor análogo digital (Ada Pmod Xilinx et al., 2011).**

## 3.2. Diseño y descripción del hardware para el analizador multicanal

El diseño para el analizador multicanal fue creado en el software de desarrollo Vivado (2015.4), para ser implementado sobre una placa de desarrollo ZedBoard que contiene un SoC de Xilinx (Zynq XC7Z020-1CLG484C). El proyecto total consta de 10 entidades: uno, identificado como *rst_processing_system7_0* el siguiente, nombrado *processing_system7_0*

también, un bloque llamado *processing_system7_0_axi_periph* así mismo, una entidad referida como *top_multicanal_0* y seis bloques *axi_gpio.* Estos seis boques son (General Purpose Input/Output, Entrada/Salida de Propósito General). Todos los bloques o entidades mencionadas anteriormente se observan en la Figura 3.5 además de las líneas de interconexión. La entidad *top_milticana_0* fue creada mediante la descripción de hardware en lenguaje VHDL, los demás bloques fueron proporcionadas por el entrono de desarrollo, en la Tabla 3.2 se mencionan las entidades que conformaron el analizador multicanal.

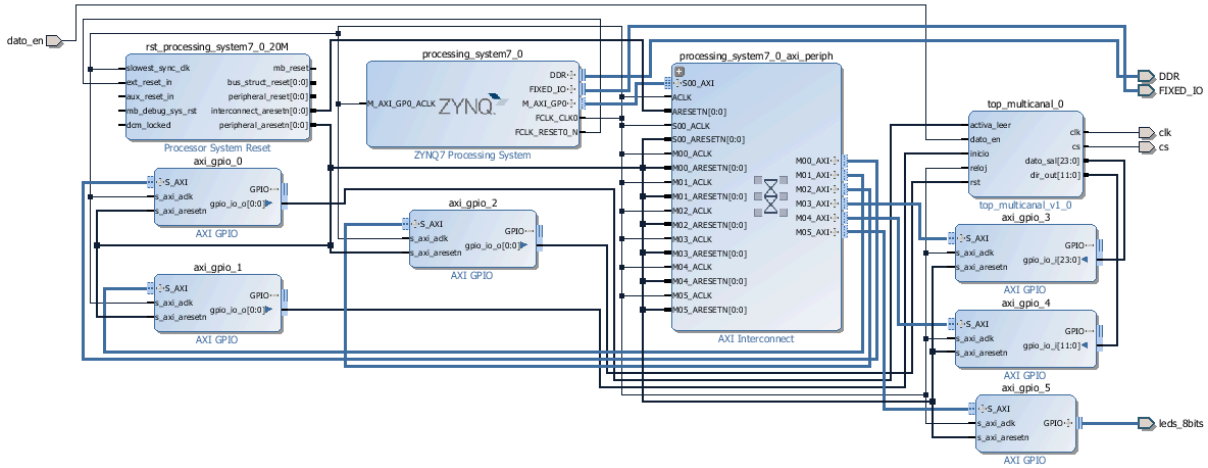


**Figura 3.5 Diagrama a bloques del analizador multicanal en Vivado.**

**Tabla 3.2 Descripción de bloques del analizador multicanal.**

| Nombre del bloque | Descripción |
|---|---|
| top_multicanal_0 | IPcore del analizador multicanal |
| rst_processing_system7_0 | Restablece los valores por default |
| processing_system7_0 | Conexión lógica entre el PS y el PL |
| processing_system7_0_axi_periph | Comunica los periféricos con el PS mediante el protocolo AXI |
| axi_gpio_0 | Salida para el inicio de lectura |
| axi_gpio_1 | Salida para comenzar el proceso de muestreo |
| axi_gpio_2 | Salida para restablecer síncronamente todo el módulo top_multicanal_0 |
| axi_gpio_3 | Entrada del dato proveniente de la memoria |
| axi_gpio_4 | Entrada de la dirección proveniente de la memoria |
| axi_gpio_5 | Salida para enciende los led's indicadores de inicio y fin de proceso |

### 3.2.1. Descripción del IPcore para el analizador multicanal

Se diseño y describió la entidad *top_multicanal_0* cuyo objetivo fue, realizar el proceso del analizador multicanal, desde la conversión de los pulsos a valores digitales hasta el almacenamiento de los datos necesarios en memoria. Se constituyó por 10 entidades: *start_0*, *ADC_0*, *discrimina_0*, *compara_0*, *start_1*, *pulso_0*, *pulso_1*, *leer_0*, *selec_0* y *memoria_0*, dichos componentes son enlistados y descritos la Tabla 3.3.

También, tiene 5 entradas (*activa*, *rst*, *reloj*, *inicio* y *dato_en*) y 4 salidas (*direccionout(11:0)*, *dato_sal(23:0)*, *pmod_clk* y *cs*), las entidades e interconexiones entre ellas se pueden observar en la Figura 3.6. El funcionamiento en general de este IPcore es, en primer lugar convertir las señales analógicos a datos digitales, después encontrar el puntos mas alto que representa la altura máxima de un pulso radioactivo, el valor máximo encontrado sirve como apuntador a una memoria en la cual a dicha localidad se ingrementa su valor en uno y por último poder leer todos las localidades de la memoria. Las entidades de este IPcore fueron elaborados en el lenguaje de descripción de hardware VHDL, posteriormente importados a la plataforma de desarrollo Vivado donde fueron interconectados para entonces poder empaquetarlos en un IPcore que funge como el analizador multicanal.

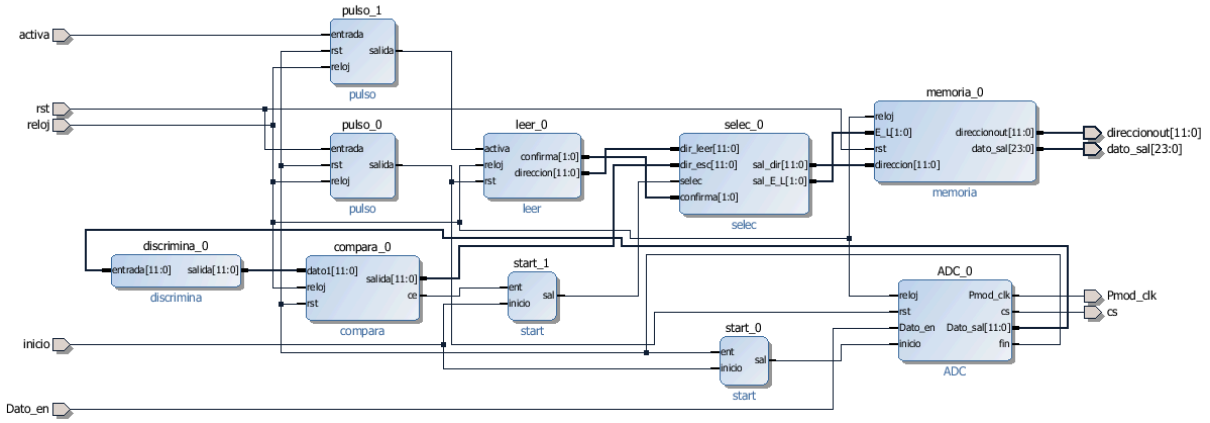


**Figura 3.6 Diagrama bloques del IPcore del analizador multicanal.**

**Tabla 3.3 Descripción de bloque del IPcore diseñado.**

| Nombre del bloque | Descripción |
|---|---|
| start_0 | Habilita al controlador del conversor análogo digital |
| ADC_0 | Controla el conversor análogo digital |
| discrimina_0 | Desecha datos insignificantes provocados por ruido |
| compara_0 | Obtiene los puntos más altos de los picos |
| start_1 | Activa la escritura o la lectura |
| pulso_0 | Restablece todas las entidades |
| pulso_1 | Activa el módulo leer_0 |
| leer_0 | Recupera los 4096 datos de la memoria uno a uno |
| selec_0 | Selecciona la dirección de lectura o escritura |
| memoria_0 | Almacena los puntos más altos de los picos |

#### 3.2.1.1. **Diseño del activador de inicio y fin de conversión**

La entidad *start_0* tiene como finalidad activar el proceso de recolección de datos durante el tiempo de muestreo establecido, dicha entidad tiene 2 entradas y una salida, las estradas se nombraron *ent* e *inicio* y la salida se identificó como *sal*. Por consiguiente, este tiene la finalidad de iniciar y terminar el proceso de conversión, cuando la entrada *inicio* está en estado bajo, esto indica que el conversor está apagado de lo contrario cuando *inicio* está en un estado alto, indica que el conversor esta encendido y lo que hay en la entrada *ent* se asigna a la salida *sal,* el valor de la entrada *ent* indica con un valor alto, que el proceso de conversión a finalizado y se puede comenzar con otra conversión.

#### 3.2.1.2. **Diseño del controlador del conversor análogo digital**

Se diseño y describió la entidad llamada *ADC_0* cuyo objetivo fue controlar el conversor análogo digital AD7476A de Digilent. La entidad está conformada por 4 entradas (reloj, rst, dato_en e inicio) y 4 salidas (pmod_clk, cs, dato_sal(11:0) y fin). La entidad se encarga de proporcionar el reloj al pin *CLK* por la salida *pmod_clk*, además envía la señal de inicio y fin de conversión mediante la salida *cs* y también se encarga de recibir por la entrada *dato_en* los 12 bits del dato convertido proveniente del conversor, los 12 bit se reciben en serie para

posteriormente poder almacenarlos en paralelo por la salida *dato_sal,* la entrada *reloj* recibe la señal de reloj de 20 MHz, la entrada *rst* se utiliza para restablecer la entidad, la entrada *inicio* y la salida *fin* se conectan a la entidad start_0 descrita en el apartado anterior.

La entidad se describió como una máquina de estados finitos (*FSM*), de 3 estados, con 2 entradas (*reloj*, *inicio* y *d*), 3 salidas (*sal*, *cs* y *fin*) y 2 señales auxiliares (*conta* y *t*) como se puede observar en la Figura 3.7. La máquina inicia en el *estado 0* inicializando todas las señales de salida y auxiliares necesarias para la *FSM,* para hacer una transición entre el *estado 0* y el *estado 1* es necesario de un ciclo de reloj y además que las señal de entrada *inicio* se active con 1, una vez en el *estado 1* se cambia la salida *cs* de un valor en alto a uno en bajo esto para iniciar el proceso de conversión e iniciar la recepción de los datos en serie por la entrada *d* y siendo almacenados en la señal auxiliar *t,* para que sea posible una transición entre el *estado 1* y el *estado 2* es necesario de tener un ciclo de reloj y además que se cumpla la condición de que la señal auxiliar *conta* sea igual a 16 que es el número de ciclos de reloj que tarda el proceso de cada conversión, en el estado dos se asignan los valores ya convertidos a la salida *sal* para ser utilizados posteriormente y las salidas *cs* y *fin* se colocan en un valor alto para indicar que el proceso de conversión a finalizado, para cumplir las especificaciones del conversor analógico a digital, mostradas en la Figura 3.4.

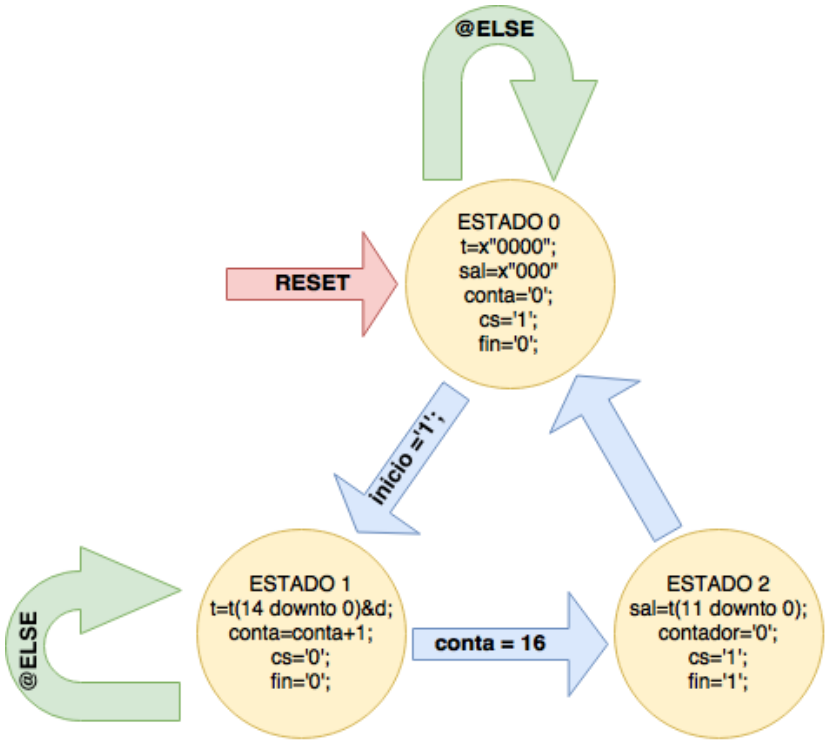


**Figura 3.7. Máquina de estados de la entidad ADC.**

### 3.2.1.3. **Discriminador**

Con la finalidad de descartar el ruido de fondo, se describió la entidad nombrado "discrimina_0", la cual se conecta entre la salida *dato_sal* de la entidad *ADC_0* y la entrada *dato1* de la entidad *comparador.* Tiene como objetivo descartar los primeros 100 canales del conversor análogo digital para eliminar el ruido cuando no haya pulsos detectados. Esto lo realiza comparando el dato de salida del *ADC_0* con el valor numérico 100, si el dato es mayor es colocado en la salida del discriminador para ser utilizado por la entidad a la cual esta conectado de lo contrario lo desecha.

Después de la simulación y prueba del funcionamiento de las entidades mencionada en los apartados anteriores se prosiguió a describir la entidad *compara*.

### 3.2.1.4. **Comparador de altura de pulsos**

La entidad *compara_0* tiene como finalidad comparar todos los valores provenientes de la entidad llamada *discriminador_0* para poder localizar el valor más alto de cada uno de los

pulsos provenientes del ADC. El proceso es comparar el dato anterior con el dato nuevo y conservando el mayor de los dos si el dato nuevo es mayor que el antiguo, esto indica que el flanco del pulso es de subida posteriormente se compara el dato conservado con el dato nuevo y esto sucesivamente hasta que suceda que el dato nuevo sea menor que el dato antiguo esto indica que el flanco es de bajada y por consiguiente se a encontrado uno de los picos que es el punto más alto de cada pulso de voltaje, esto es necesario para poder desechar los datos no útiles y poder utilizar el valor más alto que son los necesarios para el funcionamiento de MCA. Al igual que en la entidad *ADC_0,* también esta cuanta con una entrada *reloj* y una *rst,* que funcionan de la misma manera que en la antes mencionada, el bloque *compara* cuenta con dos salidas una de ellas llamada *ce* indica con un valor en alto que a encontrado uno de los picos y la otra llamada *salida* es la que contiene la dirección a la cual apunta el pico encontrado.

#### 3.2.1.5. **Activador de escritura o lectura**

La entidad *start_1* tiene como objetivo activar la escritura o lectura de la memoria. Este entidad tiene 2 entradas y una salida, la entrada *inicio* al estar en un estado bajo indica que a la salida *sal* se le asignara un cero, de lo contrario si *inicio* es uno, el dato de la entrada *ent* proveniente del comparador (indica si se a encontrado un pico), se pasa a la salida *sal*, tanto el *start_0* como el *start_1* comparten la misma entrada de *inicio*, por lo que las dos se activa a la par.

#### 3.2.1.6. **Acoplador de instrucciones para restablecer las entidades**

La entidad llamada *pulso_0* fue creada para recibir las instrucciones de restablecimiento de la parte software y acoplarlos a la parte hardware, esto debido a que un ciclo de reloj en hardware no dura lo mismo que una instrucción de código en software. Esta entidad que es una máquina de estados finitos, en la Figura 3.8 se pueden observar los 4 estados y sus conexione. La máquina de estados, inicia en el *estado 0* al inicializar la salida y allí permanece hasta que por la entrada *entrada* llegue un valor de 1, para posteriormente pasar

del *estado 0* al *estado 1*, estando en el *estado 1* y para hacer una transición al *estado 2* es necesario que *entrada* cambie a un valore de 0, después en el *estado 2* la salida se coloca en un valor alto y al pasar un ciclo de reloj automáticamente cambia al *estado 3* donde la salida la adquiere un valor en bajo, para así acoplar una señal de entrada de un tiempo indefinido a una señal de salida de un ciclo de reloj.

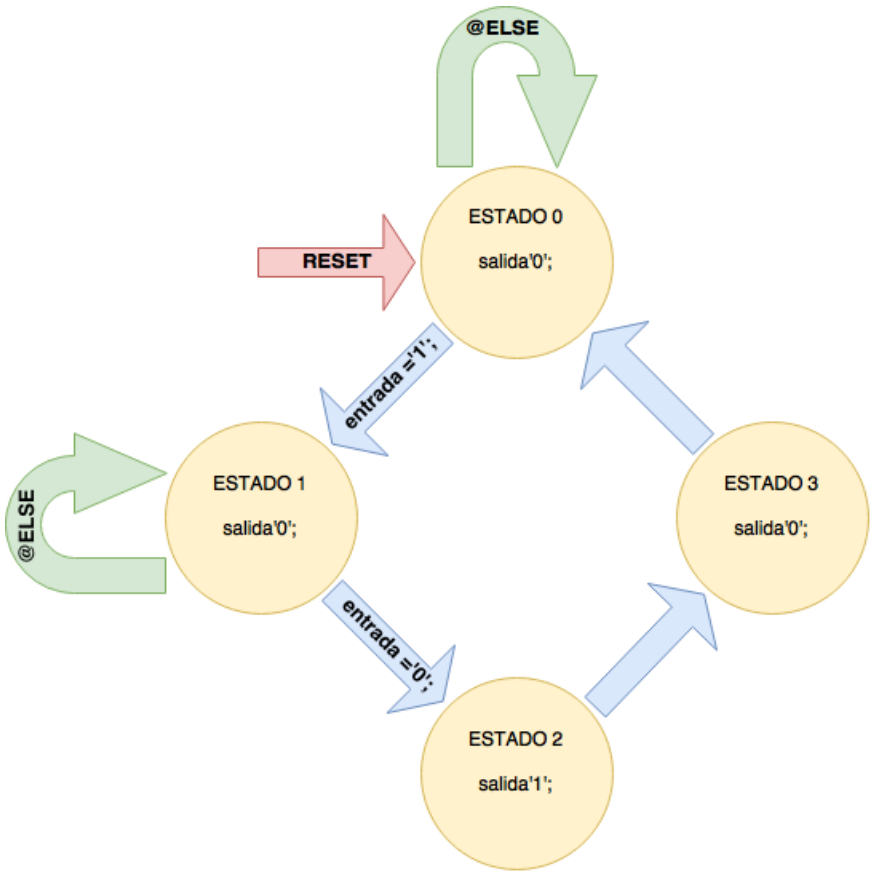


**Figura 3.8. Máquina de estados del bloque pulso.**

### 3.2.1.7. Acoplador de instrucciones para enviar direcciones de lectura

La entidad llamada *pulso_1,* tiene como finalidad acoplar las instrucciones de inicio de lectura provenientes de la parte de software a la parte de hardware, esta entidad tiene 3 entradas (entrada, clk y rst) y una salida (*salida*), dicha entidad al recibir un puso en alto por

la entrada *entrada* espera los ciclos de reloj necesarios hasta que la entrada cambie a valor en bajo para posteriormente por la salida enviar solo un pulso en alto durante un ciclo de reloj.

#### 3.2.1.8. **Lectura y restablecer de memoria**

La entidad que lleva por nombre *leer_0*, tiene como finalidad que cuando el modo de lectura o el modo de restablecer está activado, proporciona a la memoria la instrucción de lectura por medio de la salida *confirma* y todas las direcciones que van desde 0 hasta 4095 por la salida *dirección*, una por una cada ciclo de reloj son enviadas, para así en el caso de la lectura tener acceso a los datos de cada una de las localidades de memoria para ser procesados y en el caso de el restablecimiento tener acceso a todas las localidades para inicializarlas con un valor numérico de 0.

#### 3.2.1.9. **Seleccionador de modo lectura o escritura**

La entidad llamado *selec* tiene como objetivo proporcionar a la memoria las instrucciones de escritura o lectura por la salida *sal_E_L* así como también las direcciones por la salida *sal_dir* para escribir o leer según corresponda. Tiene 4 entradas (*dir_leer*, *dir_esc*, *selec* y *confirma* ) y 2 salidas (*sal_dir* y *sal_E_L*) al activar la escritura esta entidad recibe por la entrada *dir_leer* las direcciones provenientes del comparador que corresponden a los picos encontrados y las envía a la memoria y al activar la lectura esta entidad recibe las direcciones por la entrada *dir_esc* provenientes de la entidad *leer_0* y así tener acceso a todas las localidades de memoria.

#### 3.2.1.10. **Memoria de almacenamiento de picos**

Por último se describió la entidad llamado *memoria_0* tiene como objetivo almacenar las cuentas representativas a los pulsos, dicha memoria tiene 4 entradas (*reloj*, *E_L*, *rst* y *dirección*) y 2 salidas (*direccionout* y *dato_sal* ), al estar activa la escritura se le proporciona una dirección por la entrada *dirección* y esta toma el dato que tiene almacenado en esa

dirección posteriormente a ese dato le suma el valor numérico “1” y finalmente lo vuelve a almacenar en la misma dirección de memoria. Este proceso es el almacenamiento de todos los picos que proporciona la entidad *compara* mientras mas picos se encuentren con la misma altura, mayor será el numero almacenado en su canal o mejor dicho en su dirección de memoria. Esta memoria también tiene un puerto de salida llamado *dato_sal,* cuando se trata de lectura de la memoria, esta entidad recibe por la entrada *dirección* la localidad de memoria a la cual tiene que apuntar para ser leída y posteriormente asignar el contenido de dicha dirección a la salida *dato_sal,* para poder recolectar todos los datos de la meoria. También se tiene una entrada llamada *rst* que cuando está activa a la dirección de memoria a la que se esté apuntando le asigna un valor numérico ‘0’ tanto la entidad *leer_0* como la entrada *rst* tienen que estar activas durante 4096 pulsos de activación para poder restablecer toda la memoria a ceros.

### 3.2.2. Configuración del bloque processing_system7_0

El bloque *processing_system7_0* es una IPcore que contiene la plataforma de desarrollo Vivado, este bloque actúa como una conexión lógica entre el PS y el PL, el cual es configurable. La Figura 3.9 muestra la ventana de configuración, en la cual se visualizan todos los elementos que se pueden adecuar según las necesidades. Para este proyecto solo se habilitaron la *UART1* (comunicación serial, para la comunicación entre la placa de desarrollo y la computadora) y los *GPIO* (puertos genéricos de entrada y salida).

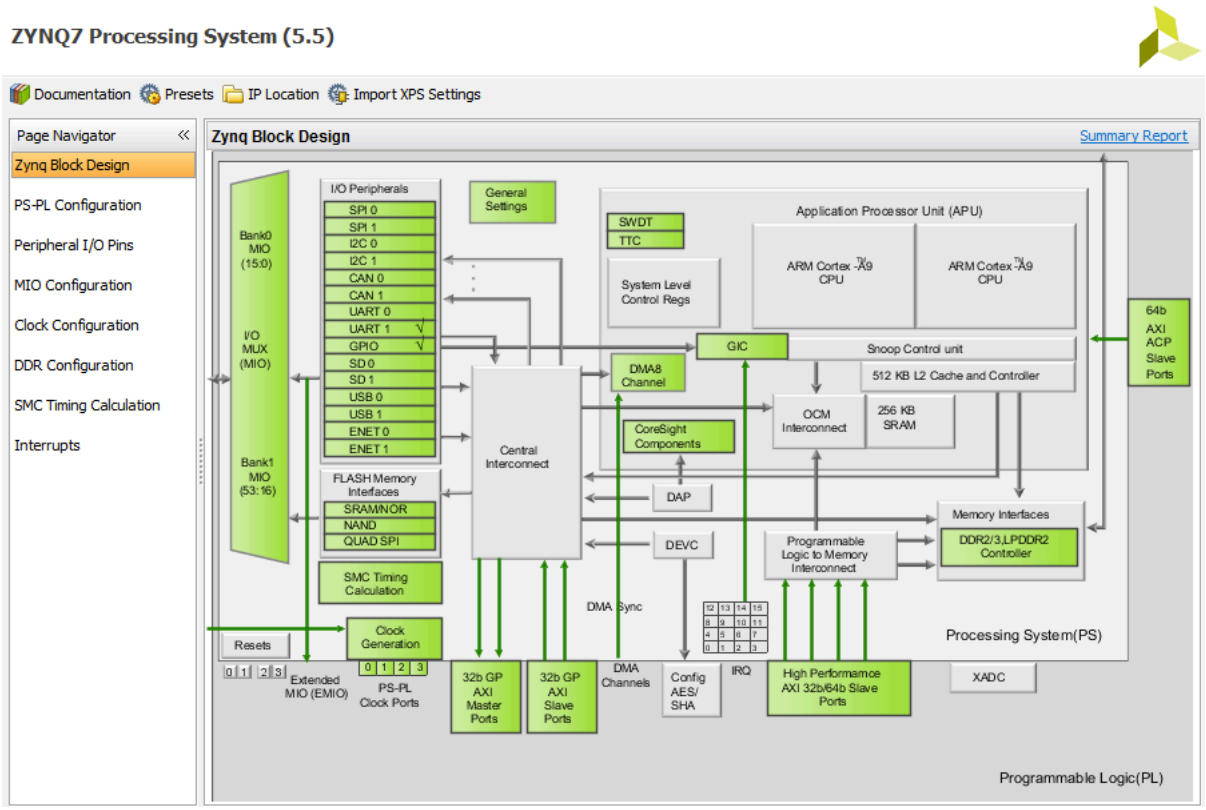


**Figura 3.9. Interface de configuración de processing system.**

Dentro del bloque *processing_system7_0* se pueden configurar para la parte lógica 4 relojes distintos, en este proyecto solo se activa uno como se puede observar en la Figura 3.10, con una frecuencia de reloj de 20 MHz, para la operación todo el sistema. Esto debido a que el conversor análogo digital que se utilizo en este proyecto especifica 20 MHz como máxima frecuencia. Para optimizar el proyecto se activan solo las características necesarias.

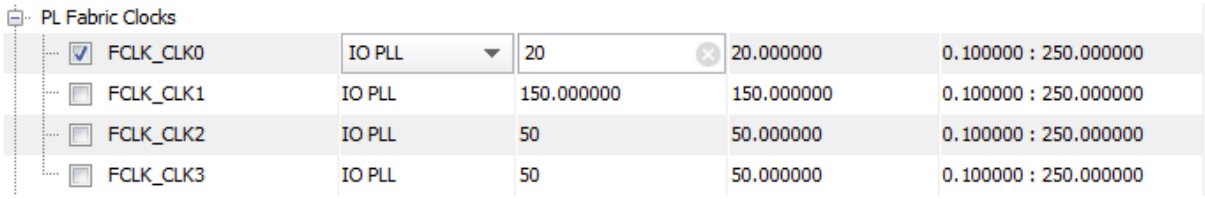


**Figura 3.10. Configuración de frecuencia de reloj de 20 MHz.**

Una ves hechas todas las configuraciones e interconexiones se puede decir que la sección de hardware está completa como se puede observar en la Figura 3.5 y se prosigue a validar el diseño, después este hardware se exporta, para posteriormente ser utilizado en la programación de la FPGA, una vez exportado el hardware del proyecto se prosigue a desarrollar el código para la parte de software, este paso se describe en la sección siguiente.

## 3.3. Programación de la aplicación para el procesador

La aplicación para la parte del software fue creada en lenguaje de programación *C*, con la herramienta de desarrollo *Software Development Kit* (SDK). Esta estructurada de la siguiente manera: definición de librerías, definición de los gpio's que fueron colocados en la parte de hardware y se encuentra dentro del menú principal, declaración de variables, declaración de la función para utilizar el puerto serie *UART*. Después inicia un ciclo *while* infinito, donde en cada ciclo se lee una variable de tipo *char* la cual controla una estructura *switch,* que contiene 4 posibles casos.

El primer caso tiene como finalidad inicial el proceso de muestreo, para esto es necesario recibir la letra *i,* la aplicación envía a los gpio's las siguientes instrucciones: envía el valor numérico '1' al *axi_gpio_1* que está conectado a la entrada *inicio* de la entidad multicanal_0 y con esto inicia el proceso de muestreo, también se envía el numero hexadecimal *0F* (00001111) al *axi_gpio_5* que está conectado a los 8 led's de la placa de desarrollo, esto para encender los 4 led's del lado derecho de la ZedBoard para indicar que el proceso de muestreo está activo.

En el segundo caso la finalidad es recolectar todos los datos almacenados en la memoria y esto corresponde a recibir la letra *t,* para esto, se encuentra un ciclo *for,* el cual se repite durante 4096 veces, en donde se envían y reciben a y de los gpio's las siguientes señales: se envía el valor '1' e inmediatamente después se envía el numero '0' hacia el *axi_gpio_0* que está conectado a la entidad *multicanal_0* en la entrada *activa,* esto para que la entidad *leer_0* coloque en su salida *direccion* las direcciones, una a una en cada ciclo, además por cada ciclo de instrucción se lee de los gpio's *axi_gpio_3* y *axi_gpio_4* la dirección de salida y el dato que contiene la dirección respectivamente y los envía a el instrumento virtual que va a graficar el espectro.

En el tercer caso el obejtibo es finalizar el proceso de muestreo y esto se hace al recibir la letra *f*, que indica que el proceso de muestreo tiene que ser detenido, para lo cual se envía el dato '0' hacia el *axi_gpio_1* que es el mismo que inicia el muestro y se envía el valor hexadecimal *F0* (11110000) al *axi_gpio_5* que es al que están conectados los led's para apagar los cuatro led's izquierdos y activa los derechos.

Para el último caso se recibe el carácter *r* el cual indica que se restablece la entidad *multicanal_0* y para esto, se programó otro ciclo que envía el valor '1' al *axi_gpio_2* que es la entrada del *rst,* esto para indicar el restablecimiento de las entidades, después envía el valor '1' e inmediatamente después envía el numero '0' hacia el *axi_gpio_0,* para nuevamente enviar a la memoria todas las direcciones y para que en cada una de ellas se almacene 0 y finalmente al concluir el ciclo se envía el valor '0' al *axi_gpio_2* para termina el restablecimiento.

## 3.4. Diseño del instrumento virtual para el analizador multicanal

Mediante la plataforma de programación gráfica LabVIEW se creó el instrumento virtual del analizador multicanal el cual aplica como interface para el usuario.

### 3.4.1. Descripción del panel frontal

El panel frontal es visible para el usuario, consta de una gráfica donde se representaran los datos provenientes de la memoria descrita en el IPcore así mismo, representan el espectro de los datos muestreados. Cuenta con un selector para el puerto serial, el cual es conectado a la placa de desarrollo, donde se recibieron y enviaron los datos, también posee un medidor de tiempo, dividido en minutos y segundos donde es especificado el intervalo de tiempo para el periodo de recolección de datos y cuenta con tres botones *iniciar análisis*, *graficar datos* y *parar proceso*. El primero envía las instrucciones para iniciar el proceso de muestreo, la recolección de datos, el tiempo de duración de encendido del proceso, al terminar ese tiempo envía la instrucción de paro de proceso, este botón también activa los led's de la placa de desarrollo para visualizar el comienzo y fin del proceso, una vez terminada la acción de muestreo, se oprimir el botón de *graficar datos* para visualizar los datos en la gráfica y por ultimo se activa el botón *parar proceso* para concluirlo y enviar la instrucción de restablecer las entidades y limpiar la memoria, para continuar los siguientes muestreos, el panel frontal del instrumento virtual se visualiza gráficamente en la Figura 3.11.

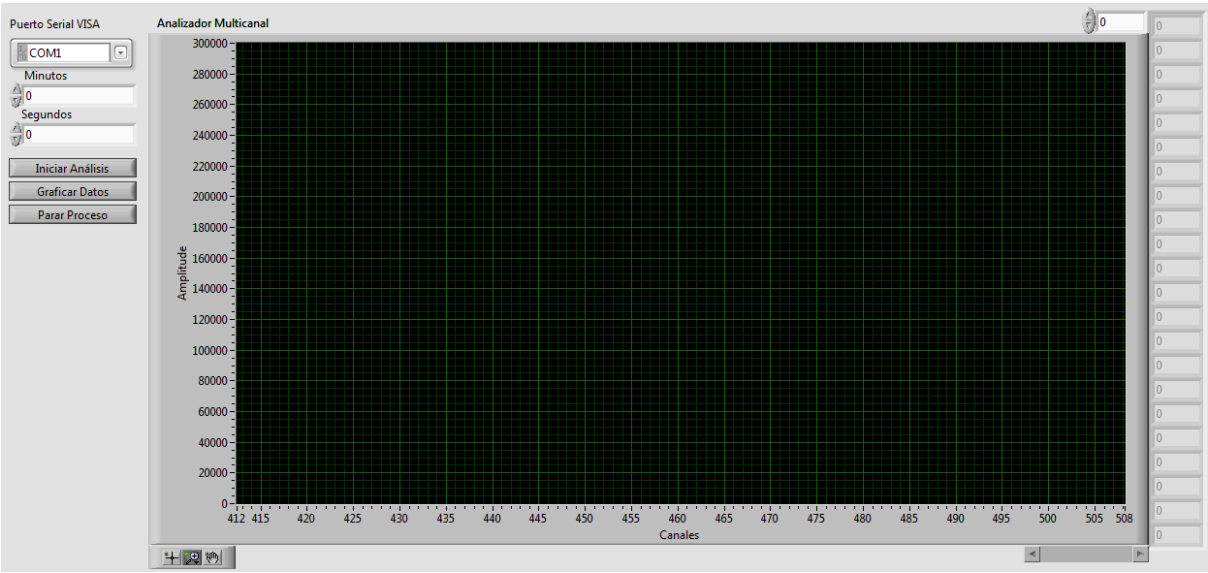


**Figura 3.11. Panel frontal para la interface de usuario del analizador multicanal.**

## 3.4.2. Descripción del panel de programación

En el panel de programación se asignan bloques a los elementos del panel frontal que interactúan con la terminal del instrumento virtual donde se indican todas las instrucciones y los componentes. En la Figura 3.12 se representa el diagrama de bloques principal del programa para la interacción con el analizador multicanal.

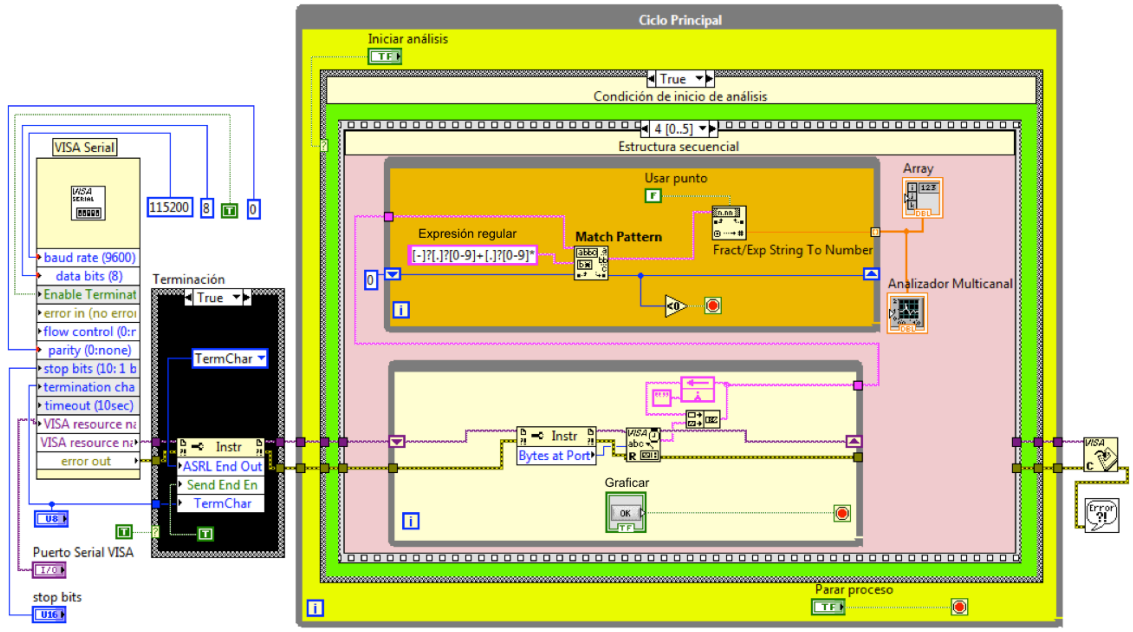


**Figura 3.12. Panel de programación para el instrumento virtual.**

### 3.4.2.1. **Configuración de VISA serial y terminación**

Los bloques llamados *Visa serial* y *terminación* se representan en la Figura 3.12, siendo el primero, donde se configura la comunicación serial entre la placa de desarrollo y el instrumento virtual. Además la conexión física se realizó en un puerto USB de la computadora mediante un adaptador serial, los parámetros de configuración para la comunicación son: (la frecuencia de comunicación (baud rate, número de unidades de señal por segundo) para este trabajo se configuro a 115200 debido a que la interface serial (UART) de la ZedBoard funciona a ese frecuencia, otro parámetro es el nombre del recurso VISA donde se indica el puerto al cual conectarse, este objeto se despliega en el panel frontal con el nombre *puerto serial VISA.* También se configura el número de bits de cada dato a transferir, para este proyecto se configuró con 8 bits, que es el valor máximo. Otras configuraciones de este bloque son la paridad igual a cero, tiempo de espera de 10 segundos, entre otras).

El bloque llamado *terminación* está compuesto por un nodo de configuración de propiedades, este se encarga de reconocer el carácter final de las líneas de datos para la lectura de información proveniente de la aplicación de software. Cada dato enviado por el software contiene al final, un salto de línea (\n), por tal motivo se configuró con *0xA* siendo este el equivalente hexadecimal de un carácter de salto de línea.

### 3.4.2.2. **Descripción de ciclo principal**

El bloque llamado *ciclo principal*, es un ciclo *while* que su única condición para finalizar es ejecutar el botón *parar proceso,* el cual es mostrado en el panel frontal, el contenido de este ciclo es una estructura de condición y un botón que la controla.

### 3.4.2.3. **Funcionamiento de condición de inicio de análisis**

Dentro del *ciclo principal* mencionado en el apartado anterior se encuentra una decisión llamada *condición de inicio de análisis* que tiene dos posibles casos controlados por el botón *iniciar análisis.* El estado inicial del botón es *false,* por lo cual no se ejecutara ninguna operación como se muestra en la Figura 3.13. El inicio del proceso de muestreo se realiza

cuando el valor del botón *iniciar análisis* cambia a *true,* como se muestra en la Figura 3.14. Esto se programo mediante una estructura secuencial.

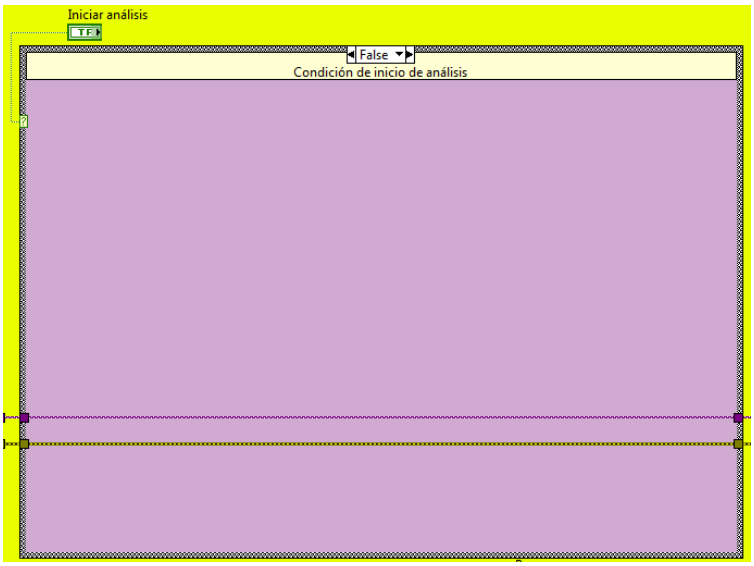


**Figura 3.13. Condición de inicio de análisis en estado false.**

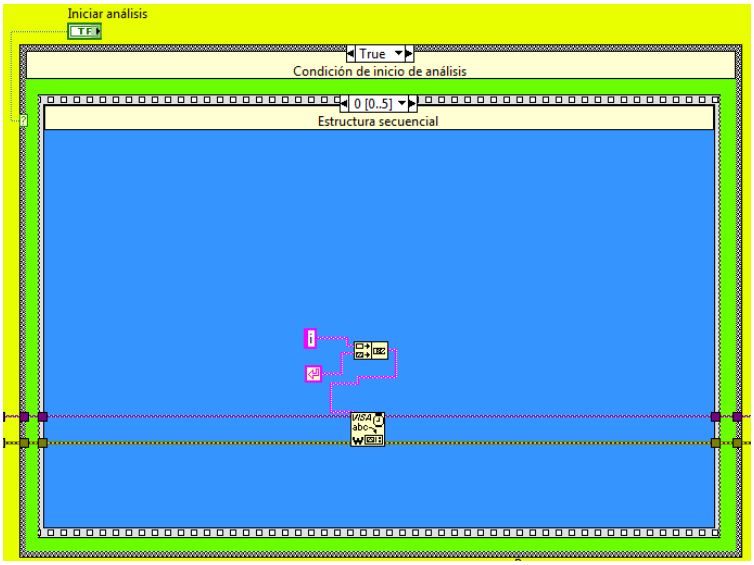


**Figura 3.14. Condición de inicio de análisis en estado true.**

### 3.4.2.4. **Seguimiento de estructura secuencial**

La estructura secuencial tiene como finalidad pasar y realizar el contenido de cada uno de sus bloques, iniciando con el primero y con la condición que no puede continuar con el segundo hasta que haya concluido con el primero y así secuencialmente hasta el ultimo. Esta estructura secuencial cuenta con 6 bloques numerados desde el numero 0 hasta el número 5.

Al iniciar el análisis la condición entra al bloque de true y por consiguiente entra al bloque de la estructura 0 como se puede observar en la Figura 3.15. Se ejecuta el contenido enviando mediante el componente escritura VISA la letra *i,* seguida de un salto de línea o mejor conocido como *enter,* rumbo a la aplicación de software para que inicie el muestreo.

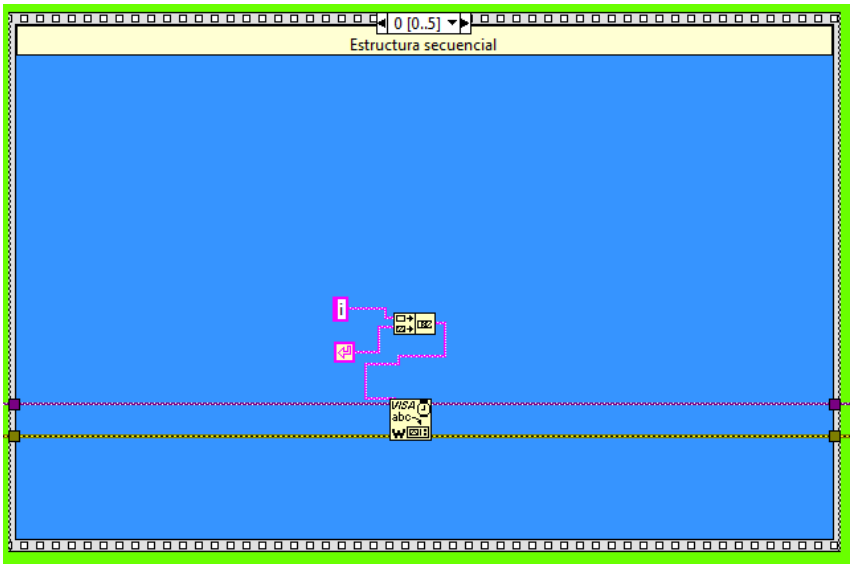


**Figura 3.15. Estructura secuencial número 0.**

Concluido el bloque anterior la secuencia continua ejecutando el segundo bloque identificado con el valor 1, en este se controla el tiempo de muestreo del analizador multicanal mediante dos cuadros de texto numéricos en donde se indican el numero de minutos y segundos posteriormente se convierten a milisegundos y el valor resultante es el numero que se le indicara al reloj, como se observa en la Figura 3.16. Por consiguiente el resultado de estas conversiones se suman y se establece como el tiempo determinado en el que el sistema multicanal estará muestreando.

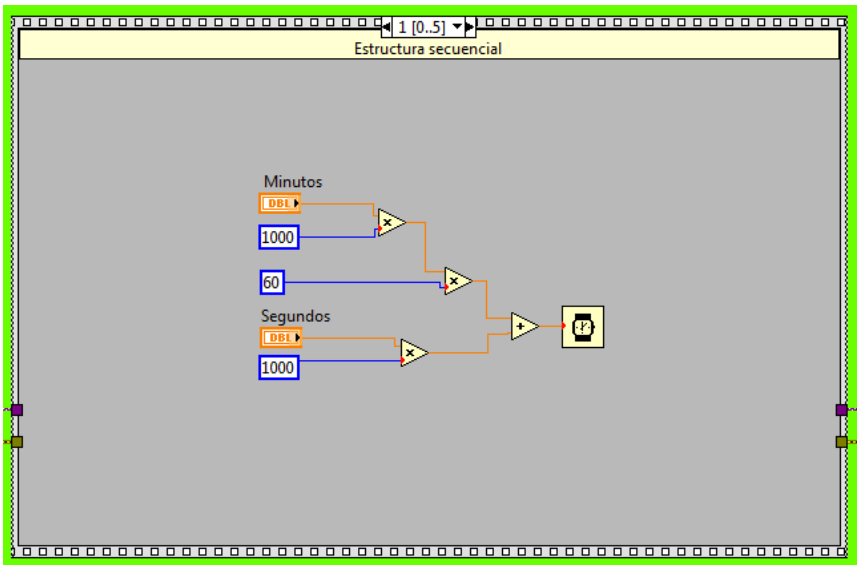


**Figura 3.16. Estructura secuencial número 1.**

Seguido a lo anterior la secuencia continuara la estructura número 2 como lo muestra la Figura 3.17. Similar al funcionamiento de la estructura 0, la instrucción se envía de la misma manera pero con el carácter *f* seguido de un salto de línea, así el procesador envié las instrucciones correspondientes para concluir el proceso de muestreo.

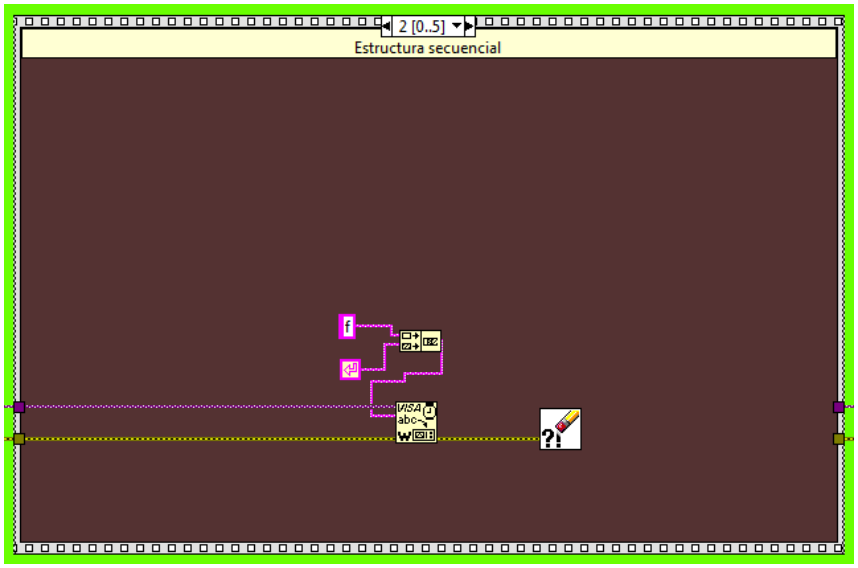


**Figura 3.17. Estructura secuencial número 2.**

La estructura numero 3, es la que inicia el proceso de recolección de los datos que tiene almacenada la memoria de la placa de desarrollo y esto lo hace enviando el carácter *t* seguido de un salto de línea, en la Figura 3.18 se puede visualizar la estructura numero 3.

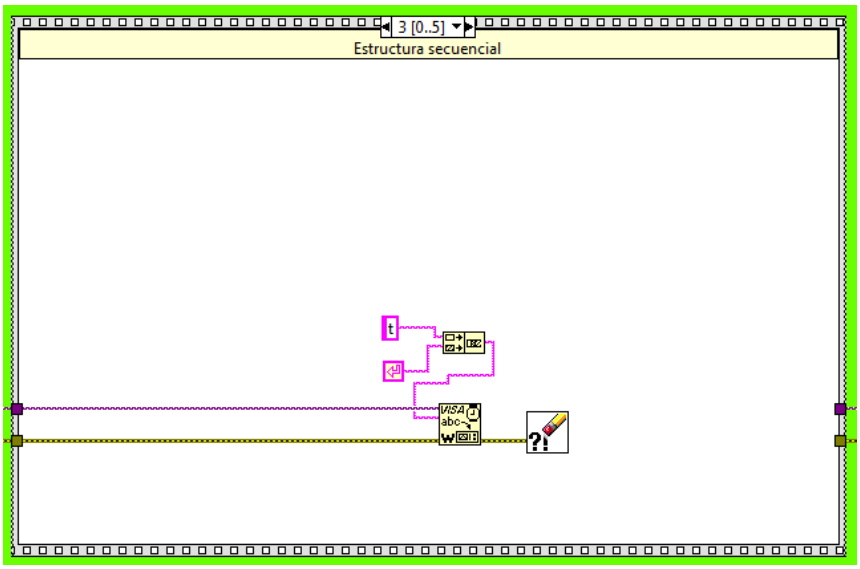


**Figura 3.18. Estructura secuencial número 3.**

Una vez iniciado el proceso de extracción de los datos de la memoria la secuencia sigue en la estructura numero 4 mostrada en la Figura 3.19, en esta estructura se encuentran dos ciclos *while*, el primero tiene el objetivo que mediante una función de lectura de VISA, lee todos y cada uno de los datos provenientes de la memoria y los va almacenando en un túnel a la salida de este ciclo, una vez recolectados todos los datos para poder salir del ciclo se tiene que presionar el botón *graficar* y continuar con el segundo *while*. Todos los datos almacenados en el tunes son de tipo *char* o carácter, por lo cual no se puede hacer una gráfica con ellos para lo cual se necesita el segundo ciclo *while*, el cual convierte todos los datos almacenados a formato numérico, una ves convertido el ultimo valor ese ciclo llega a su fin y los datos son almacenados en un arreglo y graficados.

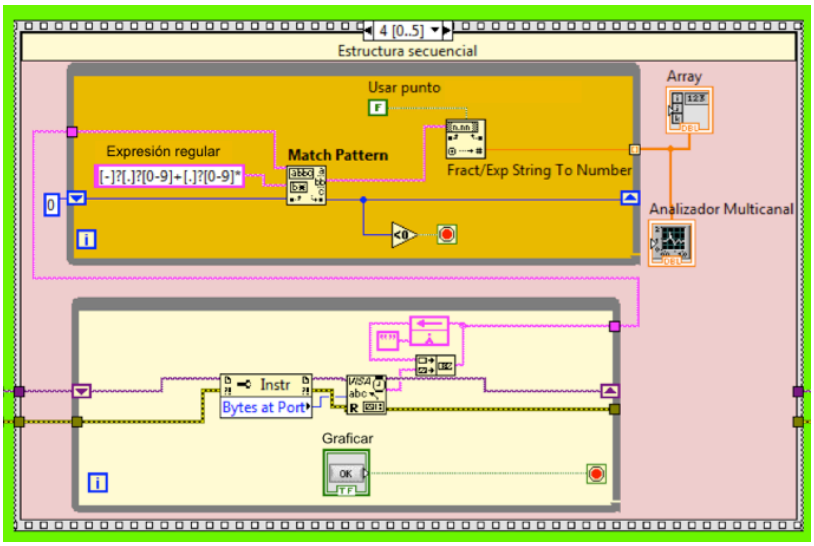


**Figura 3.19. Estructura secuencial número 4.**

Y por último en la estructura número 5 se envía la letra *r* seguida de un salto de línea al procesador para iniciar el proceso de restablecer todas las entidades, esto con la finalidad de poder continuar haciendo otras mediciones en el analizador multicanal.

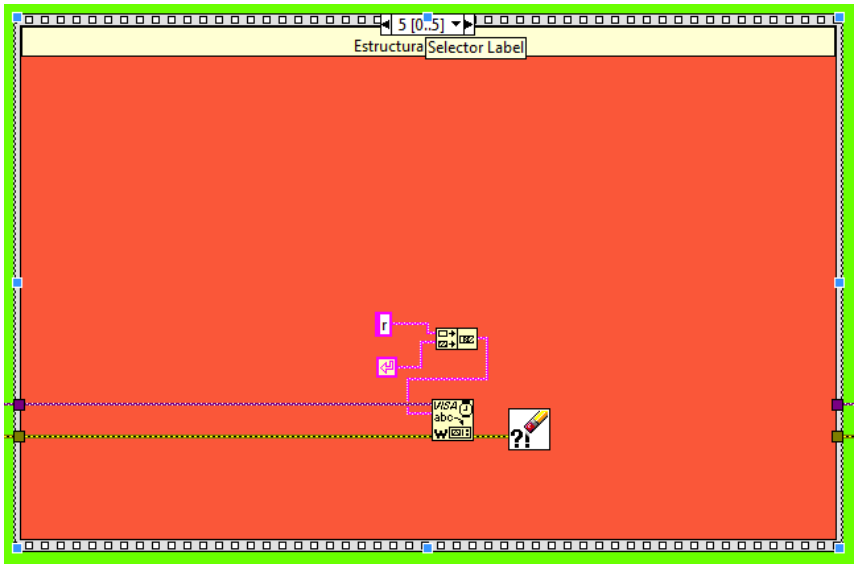


**Figura 3.20. Estructura secuencial número 5.**

## 3.4.2.5. **Cerrar sesión de VISA y mostrar errores**

Según las indicaciones del desarrollador de VISA, este protocolo tiene que ser cerrado cuando se termine de utilizar. Y es opcional colocar una instrucción para poder visualizar los

errores que se hayan producido en las entidades de VISA, conectada a la instrucción de cerrar VISA como se muestra en la Figura 3.12 al lado derecho del ciclo principal.

# Capítulo 4. Resultados y discusión

En este capítulo se presentan los resultados obtenidos de la caracterización y de las mediciones realizadas con el analizador multicanal embebido en un FPGA, los cuales son visualizados en el instrumento virtual diseñado para su control.

## 4.1. Caracterización del conversor analógico digital

El diseño del controlador del ADC fue caracterizado con un generador de funciones con el objetivo de establecer que el sistema funcione con una repuesta de tipo lineal, corroborando así para la prueba experimental un coeficiente de correlación R2 igual a 0.99999912. La Figura 4.1 muestra la relación entre los datos calculados y los datos obtenidos, donde es perceptible la semejanza entre dichos valores.

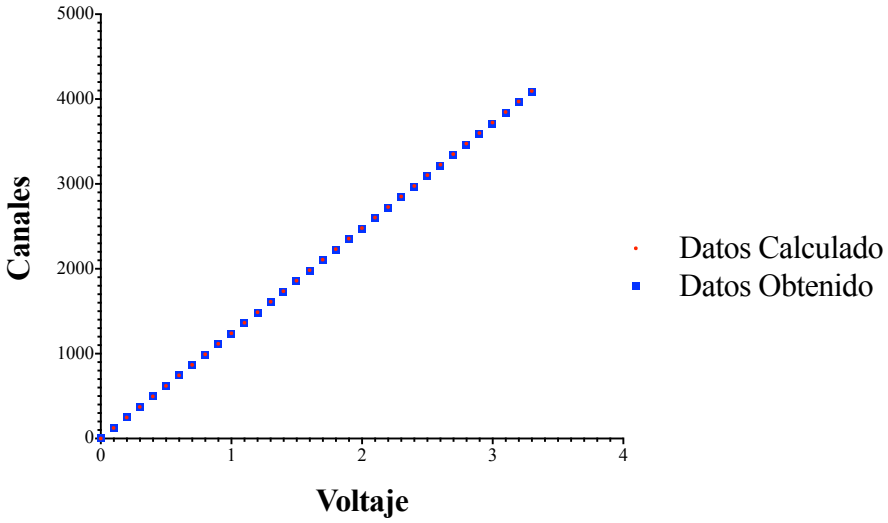


**Figura 4.1. Grafica de linealidad del analizador multicanal.**

## 4.2. Funcionamiento de los componentes del IPcore

Los resultados obtenidos del núcleo de propiedad intelectual se visualizaron mediante simulaciones de las entidades que lo conforman, a continuación se muestran y detallan algunas de esas simulaciones.

### 4.2.1. Simulación de la entidad start

La simulación mostrada en la Figura 4.2, se encargó de probar el funcionamiento de la entidad *start*. Dicha simulación comienza en 0 ns. Entre el periodo de 50 a 200 ns la entrada *ent* es activada con un valor en alto pero la salida no es modificada y continua en un estado bajo, inmediatamente después la señal *inicio* cambia su valor a 1 y una vez hecho esto, en los periodos de 400 y 800 nanosegundos la entrada cambia de estado bajo a un estado alto y por consiguiente a la salida *sal* se le asigna el mismo valor de la entrada cuando la señal *inicio* se encuentra activa.

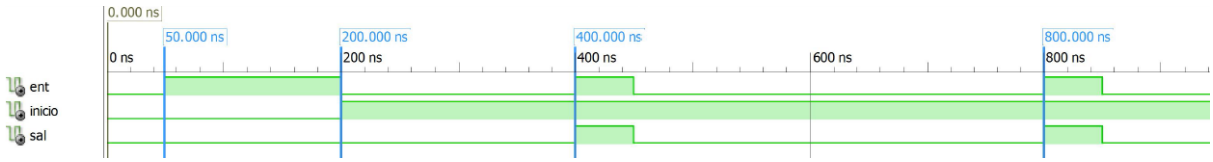


**Figura 4.2. Simulación de la entidad start.**

### 4.2.2. Simulación del controlador para el ADC

Se realizó una simulación mostrada en la Figura 4.3 para corroborar el correcto funcionamiento de la entidad descrita para controlar el conversor analógico digital, como lo señala la Figura 3.4, en donde se visualizan las especificaciones técnicas. Se puede observar que la simulación comienza en 0 ns, transcurridos 25 ns la señal de entrada *inicio* se activa con 1 y por consiguiente la señal *cs* que enciende con un valor en bajo el conversor analógico digital se coloca en 0, seguido a esto por la entrada *dato_en* se reciben 4 ceros que corresponde al periodo de 50 a 250 ns, después por la misma entrada y en el periodo de 250 a 825 ns se reciben los 12 bits que junto a los 4 ceros se van almacenando en una señal temporal llamada *temp1* y por ultimo en el nano segundo 875 se colocan en la salida los 12 bits del dato convertido, para ser utilizados, la señal fin se activa en alto para iniciar que el proceso de conversión a finalizado y se puede comenzar otro proceso de conversión. El proceso de conversión tarda 850 nanosegundos desde que se enciende el ADC y hasta que se obtienen los 12 bits del dato convertido puede ser utilizado.

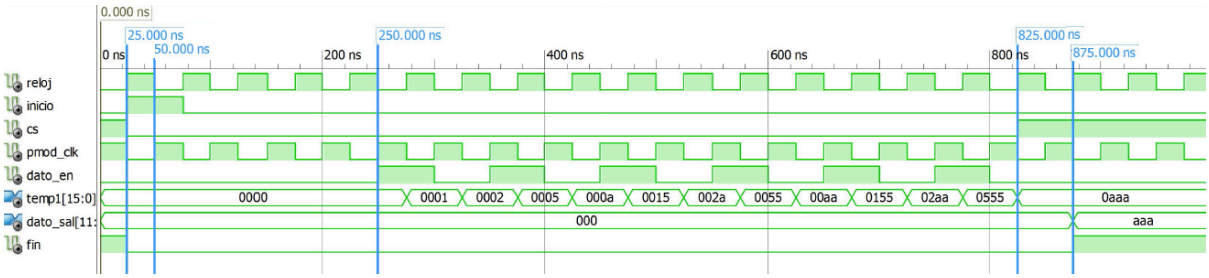


**Figura 4.3. Simulación de la entidad que controla el ADC.**

### 4.2.3. Simulación del discriminador

Para cerciorarse del que la entidad *discrimina* funciona adecuadamente se realizó una simulación la cual se muestra en la Figura 4.4, como se puede observar solo los datos entrantes mayores a 100 son colocados en la salida de lo contrario son desechados. La simulación comienza en 0 μs y en la entrada se encuentra el dato con valor numérico 1 el cual es desechado, en 0.85 μs de igual manera el dato 17 es descartado, al transcurrir 1.7 μs se coloca en la entrada el dato 273 el cual es asignado a la salida, de igual manera que el dato 1541 que ingresa al pasar 3.4 μs del inicio de la simulación.

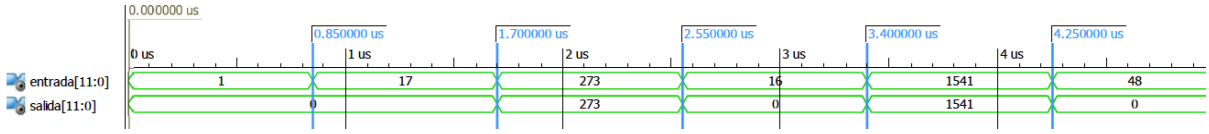


**Figura 4.4. Simulación de la entidad discrimina.**

### 4.2.4. Simulación de la entidad pulso

Se confirmó el funcionamiento mediante la simulación de la entidad *pulso* mostrada en la Figura 4.5, esta simulación inicia en 0 ns, tanto la entrada como la salida se encuentran en 0 al transcurrir 75 ns la entrada cambia a un valor 1 y pasados 725 ns la entrada vuelve a un valor de 0, después de un ciclo de reloj que equivale a 50 ns a la salida se le asigna el valor de 1 durante 50 ns y vuelve a 0, la entidad pulso es un acoplador que al entrar un pulso espera el tiempo necesario para que este caiga a 0 y así por la salida sacar un pulso con duración de un ciclo de reloj.

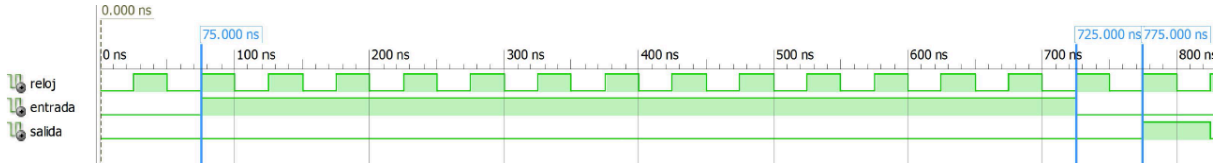


**Figura 4.5. Simulación de la entidad pulso.**

### 4.2.5. Simulación de la entidad leer

La entidad *leer* se simuló y de ella se obtuvo la Figura 4.6 en la cual se pudo garantizar que dicha entidad cumplía con las características para lo cual fue creada. De igual manera que las simulaciones anteriores esta comienza en 0 ns, al estar en estado bajo la entrada *activa* la salida *dirección* permanece inactiva como se ve reflejado de el nanosegundo 25 al 275, inmediatamente después se cambia la señal *activa* a un valor 1, para después por la salida enviar las 4096 direcciones, como se observa en el rango de 275 a 825 ns donde se envían las primeras 11 direcciones y en el rango de 204525 a 205075 ns se visualizan las ultimas 11.

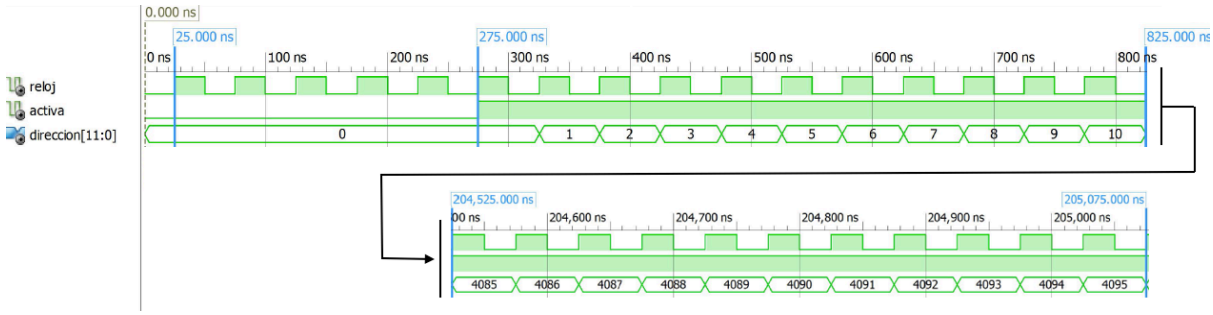


**Figura 4.6. Simulación de la entidad leer.**

### 4.2.6. Simulación de la memoria

La simulación de la entidad memoria comienza en 0 ns como se puede observar en la Figura 4.7, inicia en modo de escritura con un valor de *01* en la entrada *e_l* y apuntando a la dirección 0 donde tiene almacenado un 0, al transcurrir 125 ns cambia a modo de escritura y la dirección cambia a 10 donde se mantiene por 5 pulsos de reloj, después en el nanosegundo 375 vuelve a cambiar a modo de lectura sin modificar la dirección, esto da como resultado que a la salida *dato_sal* se le asigne el valor 5 que son los 5 pulsos que estuvo activa la escritura, en el periodo de 425 y 575 nanosegundo se activa la escritura y se apunta a la

dirección 75 donde permanece activa por 3 siclos de reloj, por ultimo se le dicha dirección y se muestran los 3 pulsos almacenados.

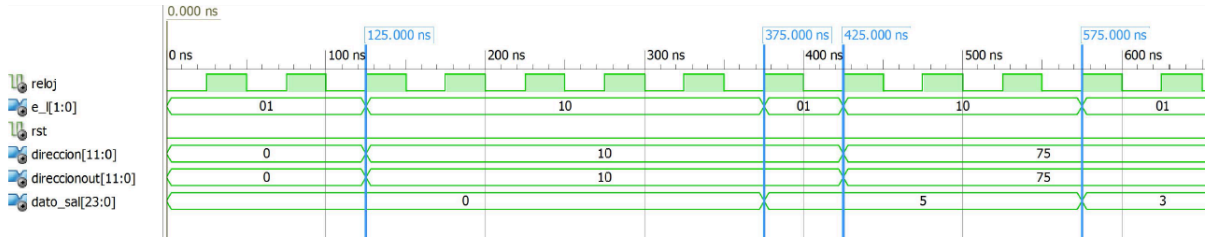


**Figura 4.7. Simulación de la memoria.**

## 4.3. Prueba de conectividad del Zynq con la PC

Para las pruebas de la parte de software se hizo uso de una consola instalada en la computadora que fungió como el instrumento virtual. Dicha consola se muestra en la Figura 4.8, se observa indicado con el número (1) el manejador de las sesiones que es donde se selecciona y configura el puerto por el cual se transmitirán y recibirán los datos de la placa de desarrollo. Una vez configurado y conectada la sesión con el puerto *serial com3,* se carga e inicia el sistema, en la consola se despliega el texto *Escribe i para comenzar,* después aparece el texto *Escribe f para terminar,* el tiempo entre el inicio y fin del proceso no se puede indicar, realizado esto se escribe la letra f y aparece un texto el cual pide escribir la letra t para iniciar con la recolección de los datos de la memoria y ser visualizados. En la sección marcada con el numero (2) se pueden observar las instrucciones así como también las primeras 10 direcciones de memoria cada una con el dato que contiene, en la sección (3) se muestran los datos desde la dirección 989 hasta la dirección 999 estos datos representan cuentas reales de una prueba de Cs137. Por último en la sección (4) se muestran las 10 direcciones finales de memoria y sus datos correspondientes.

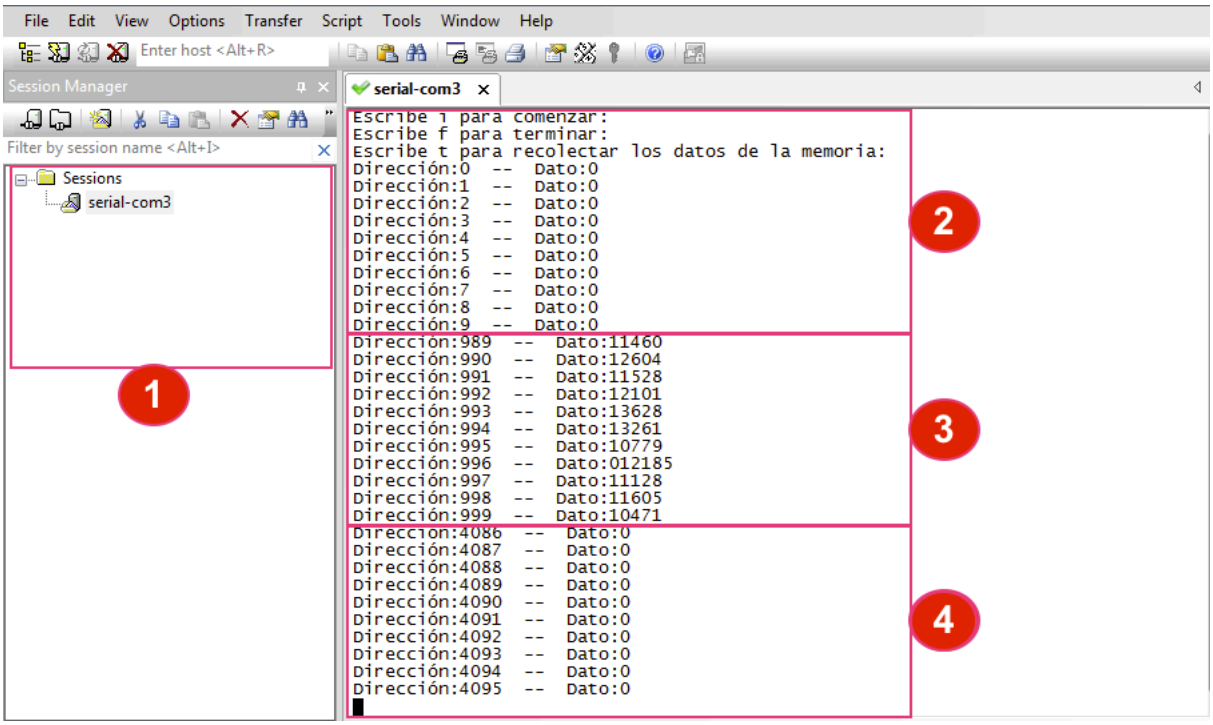


**Figura 4.8. Resultados mostrados en consola.**

## 4.4. Muestra de resultados en el instrumento virtual

Se realizaron pruebas con detectores de centelleo y detectores de semiconductor, utilizando fuentes radioactivas para así poder obtener espectros que se pueden visualizar en el instrumento virtual.

### 4.4.1. Comparación con analizador multicanal ORTEC

Una vez caracterizado el analizador multicanal, se realizó el proceso de pruebas con un sistema de espectrometría de rayos gamma mostrado en la Figura 4.9, utilizando un detector de centelleo de Nal(TI) cilíndrico que tiene una dimensión de 3 pulgadas de diámetro por 3 pulgadas de altura (1), dicho sistema consta de un gabinete o NIMbin (2), una fuente de alto voltaje (3) y un amplificador espectroscópico (4). Posteriormente se compararon los resultados con un analizador multicanal de 2048 canales de la marca Ortec montado sobre una computadora (5). En la Figura 4.10 se muestra un espectro de una fuente sellada de cesio 137 tomado con el analizador multicanal embebido en la FPGA, en contraste en la Figura

4.11 se muestra el mismo espectro adquirido con el analizador multicanal de Ortec. Claramente se observa que los dos espectros son muy similares en cuanto a la forma del espectro en el cual se puede ver el fotopico característico del Cs-137 que es de 662 keV.

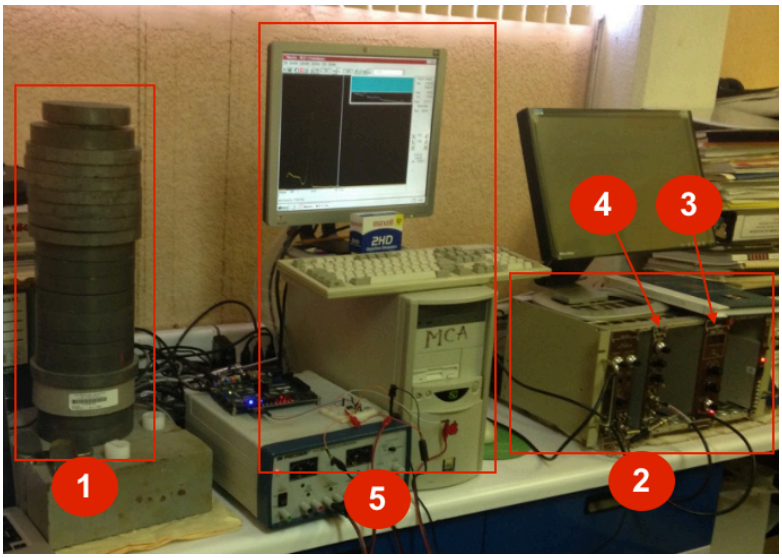


**Figura 4.9. Sistema espectroscópico de rayos gamma.**

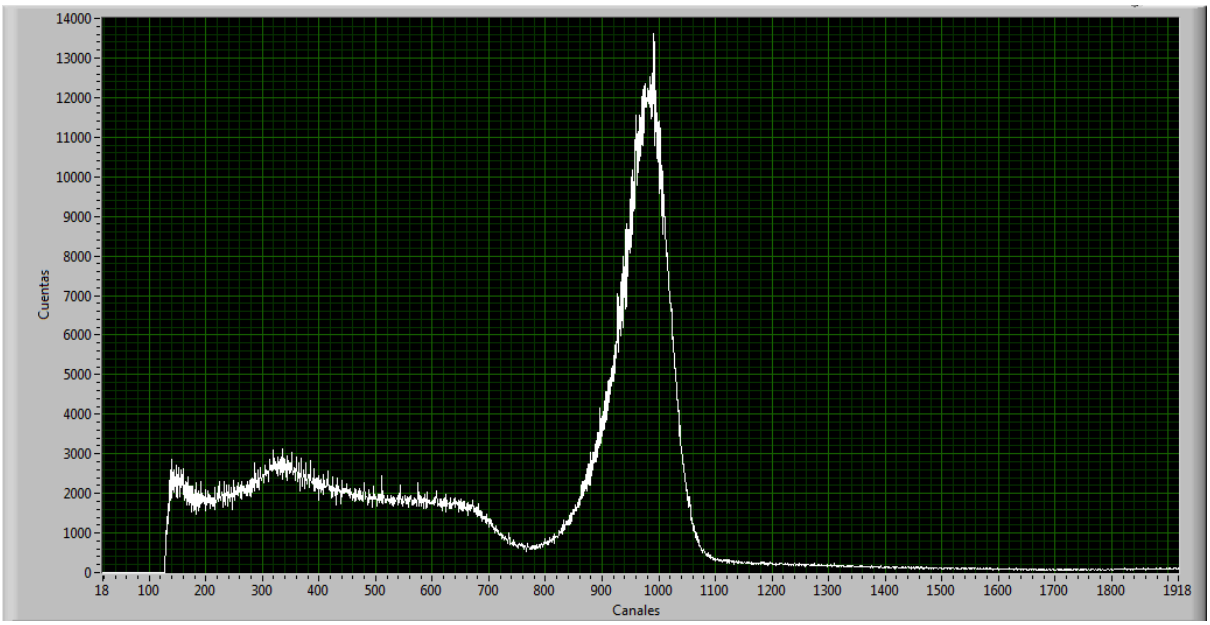


**Figura 4.10. Espectro de Cs 137 MCA con Zynq.**

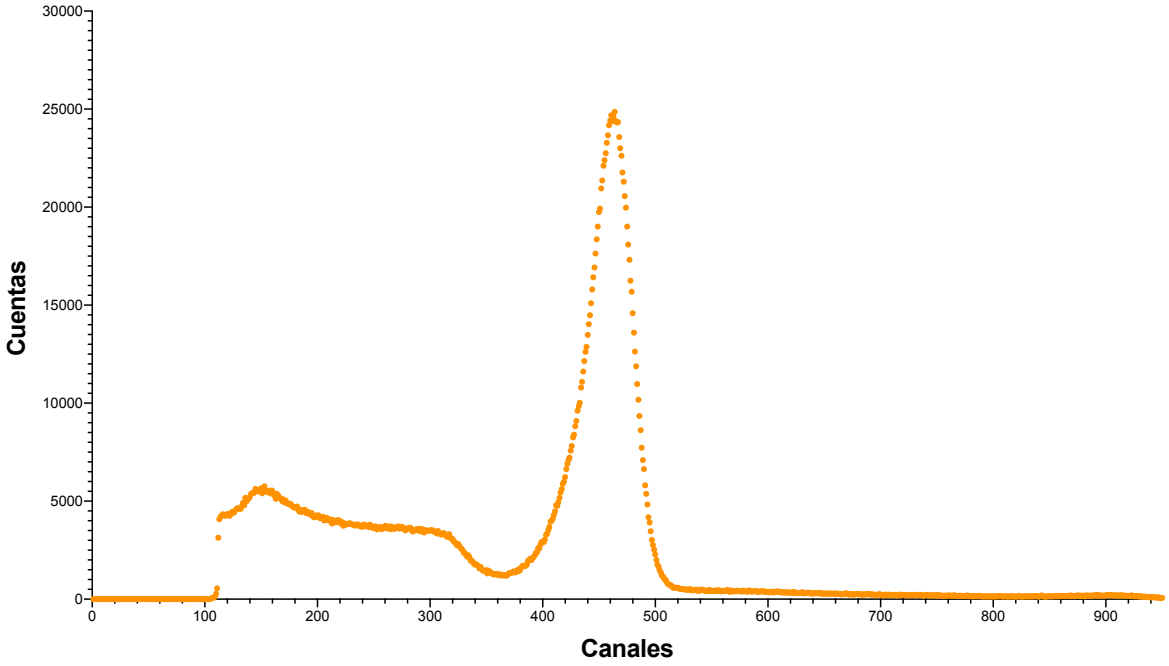


**Figura 4.11. Espectro de Cs 137 MCA Ortec.**

En la Figura 4.12 y la Figura 4.13 también se hace una comparación de espectros entre el analizador multicanal diseñado e implementado en este trabajo y el analizador comercial de Ortec, pero en esta ocasión muestreando la fuente sellada del cobalto 60; de igual manera se pueden identificar perfectamente los dos fotopicos característicos del cobalto 60 que son de 1.17 MeV y 1.33 MeV.

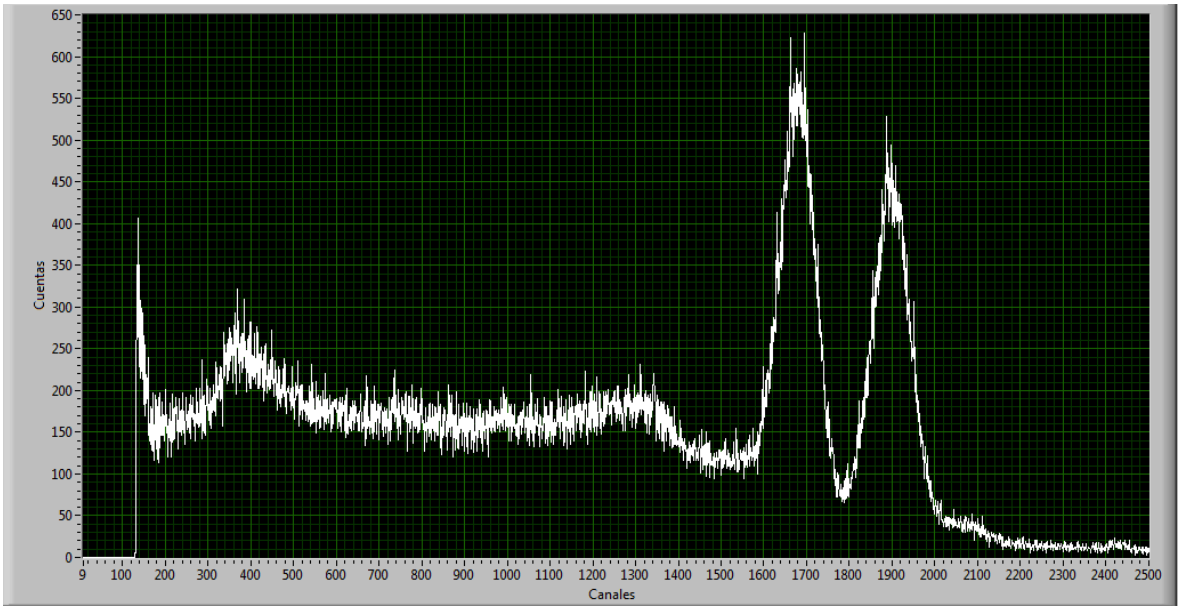


**Figura 4.12. Espectro de Co 60 MCA con Zynq.**

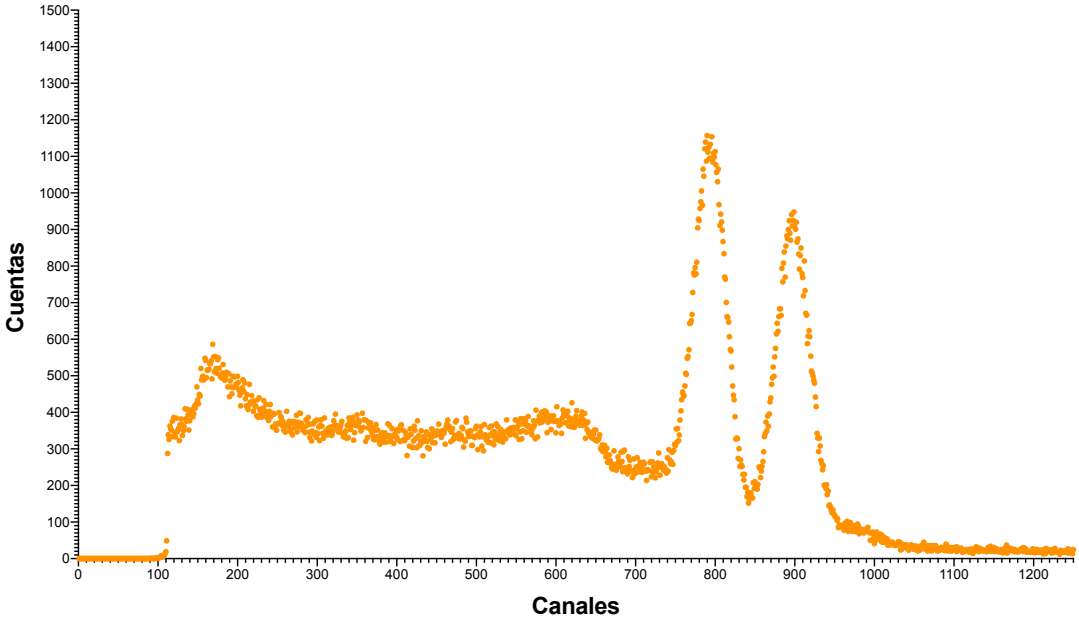


**Figura 4.13. Espectro de Co 60 MCA Ortec.**

Posteriormente se colocaron las dos fuentes selladas (Cs137 y Co60), se tomaron los espectro simultáneamente con el mismo tiempo de muestreo en los dos multicanales (MCA embebido en el FPGA y MCA Ortec), dando como resultado los espectros mostrados en la Figura 4.14 y la Figura 4.16 respectivamente. Debido a la gran diferencia entre las actividades del Cs 137 y el Co 60, la escala de el eje *Y* que corresponde a las cuentas, se coloco en escala logarítmica para poder visualizar los fotopicos característicos de estos radioisótopos.

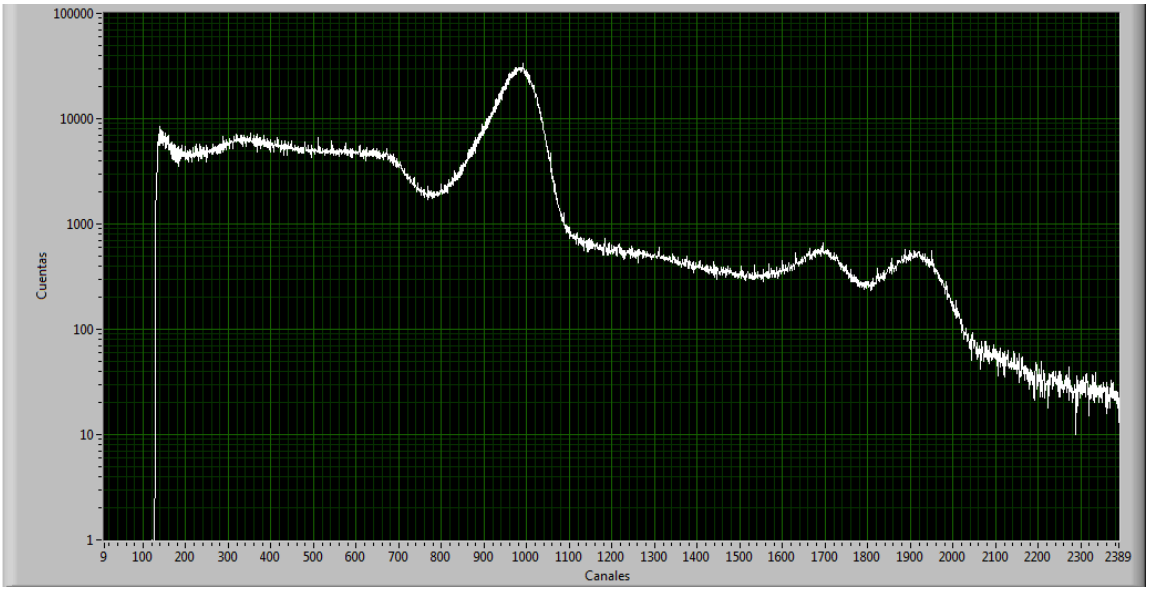


**Figura 4.14. Espectro de Cs 137 y Co 60 MCA Zync.**

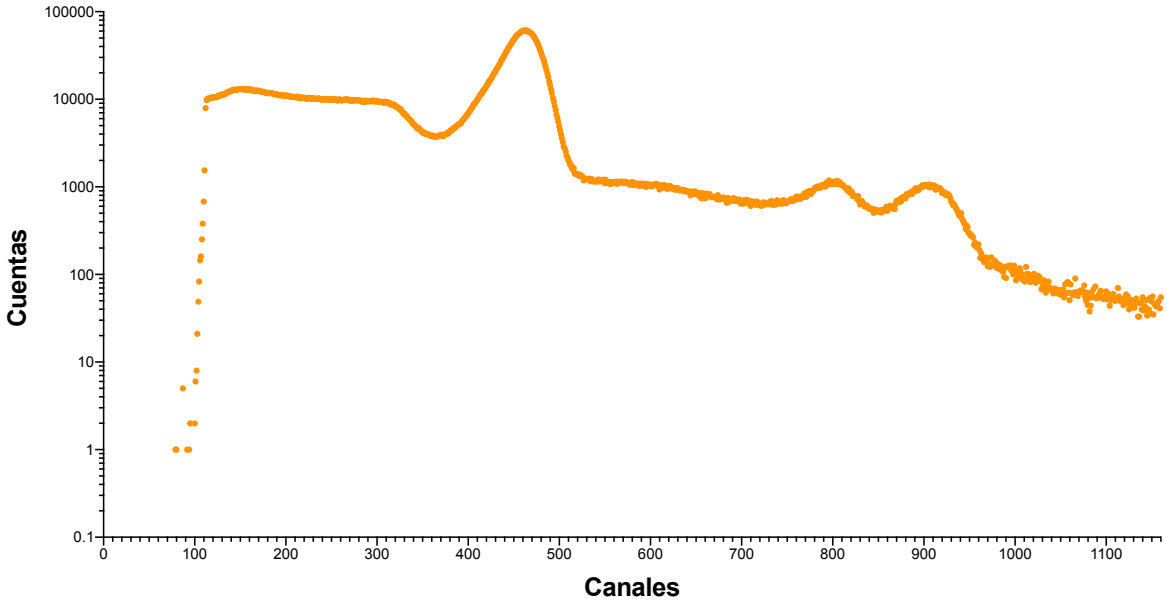


**Figura 4.15. Espectro de Cs 137 y Co 60 MCA Ortec.**

Conociendo los 3 canales en los cuales se almacenaron la mayor cantidad de cuentas en el muestreo y la energía representativa de cada fotopico, en relación a la Tabla 4.1, se prosiguió a hacer la calibración mediante regresión lineal, la cual es mostrada en la Figura 4.16, el coeficiente de correlación $R^2$ es igual a 0.9993 y la ecuación que representa a la línea es *Y=0.0007391*X-0.07333.*

**Tabla 4.1 Canales y energías de Cs 137 y Co 60.**

| Canal (#) | Energía (MeV) |
|---|---|
| 992 | 0.662 |
| 1696 | 1.17 |
| 1888 | 1.33 |

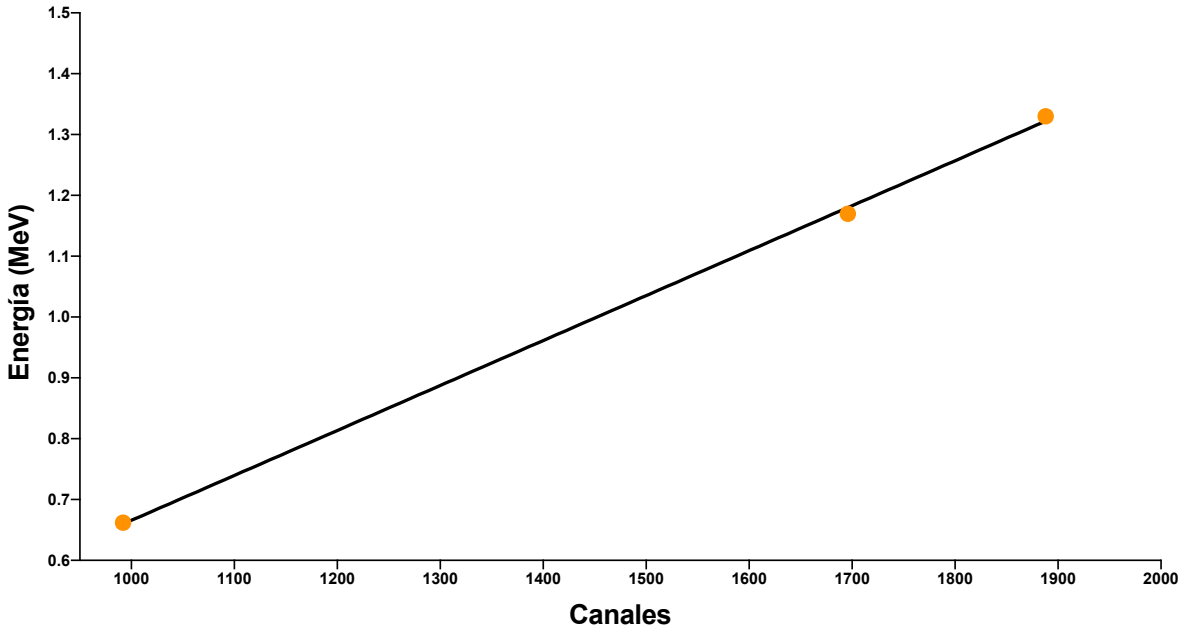


**Figura 4.16. Calibración con Cs 137 y Co 60.**

### 4.4.2. Comparación con analizador multicanal Canberra

Se realizaron pruebas con un sistema espectroscópico de radiación gamma, el cual está constituido de un detector de Ge el cual se observa en la Figura 4.17, un contenedor con nitrógeno líquido, un gabinete o NIMbin , una fuente de alto voltaje, un amplificador espectroscópico y un analizador multicanal de la marca Canberra con una resolución de 8192 canales. Se muestreo una fuente sellada de Europio 152 se tomaron dos espectros, uno tomado con el analizador multicanal embebido en el FPGA mostrado en la Figura 4.18 y el otro con el analizador multicanal de la marca Canberra se puede visualizar en la Figura 4.19. En base a la Tabla 4.2 se realizó la calibración según las energías características del Europio 152, dicha calibración se ver reflejada en la Figura 4.20, calculando la regresión lineal, con un coeficiente de correlación igual a 0.9993, se obtuvo la ecuación representativa de la recta que es igual a *Y=0.0004752*X-0.004023.*

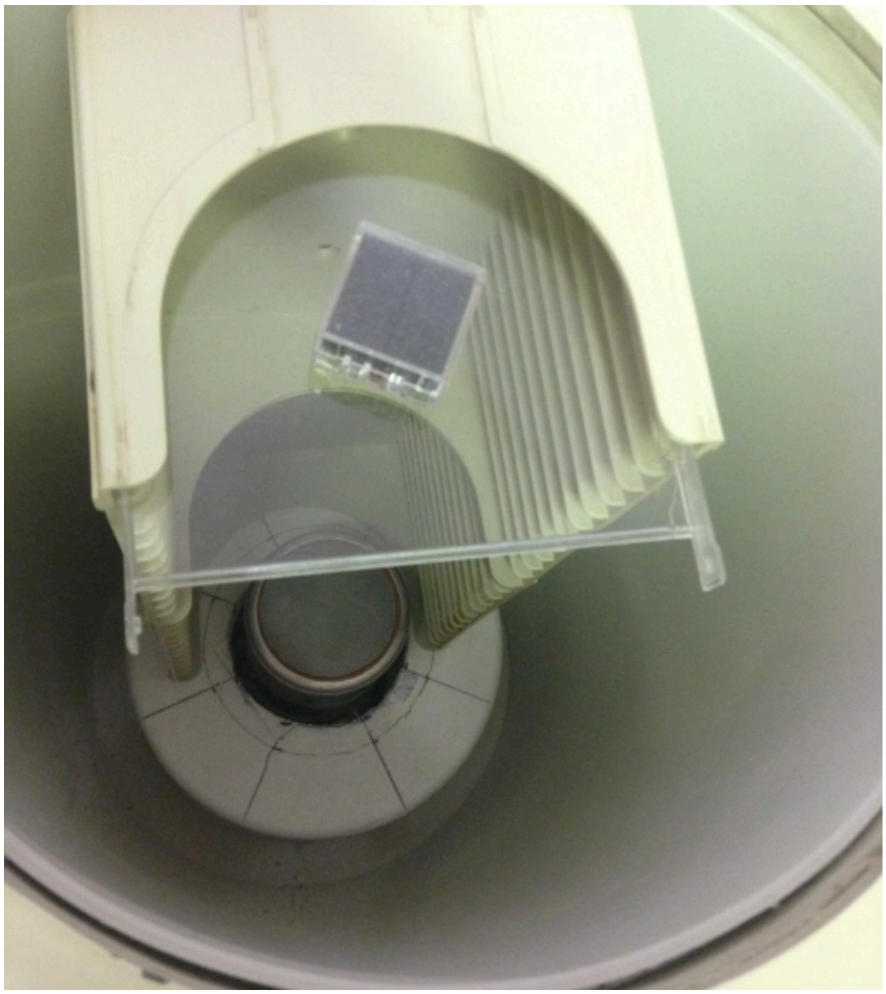

**Figura 4.17. Detector de Ge**

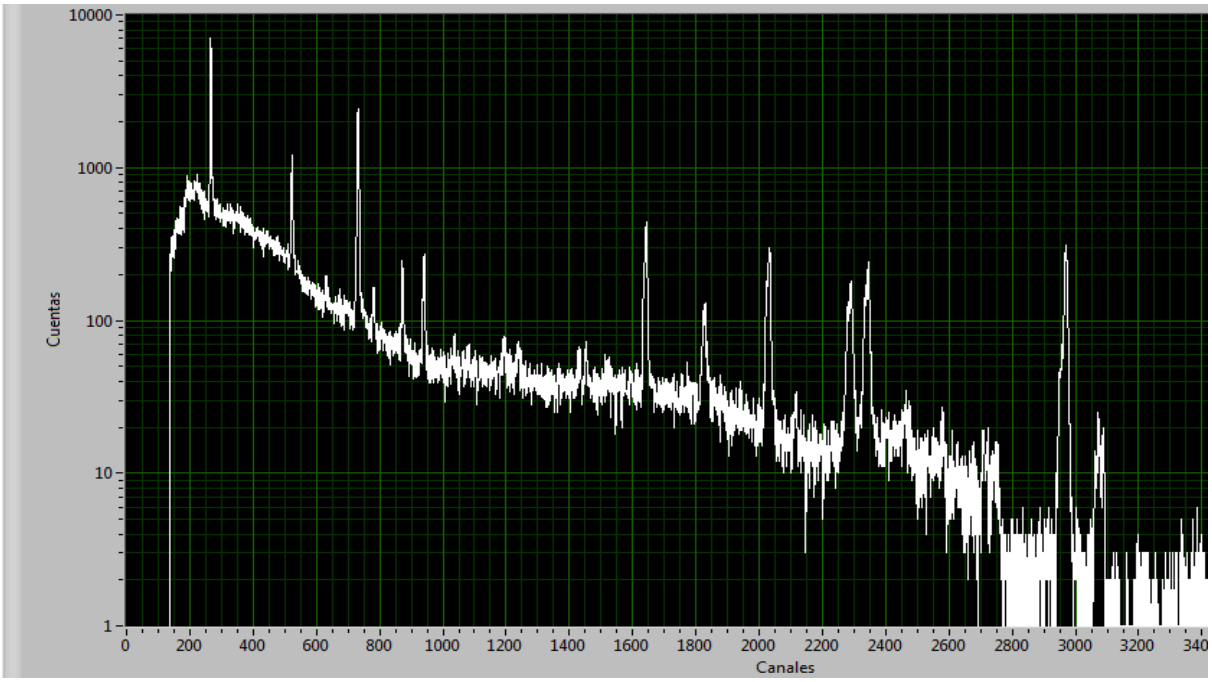


**Figura 4.18. Espectro de Eu 152 con MCA Zync.**

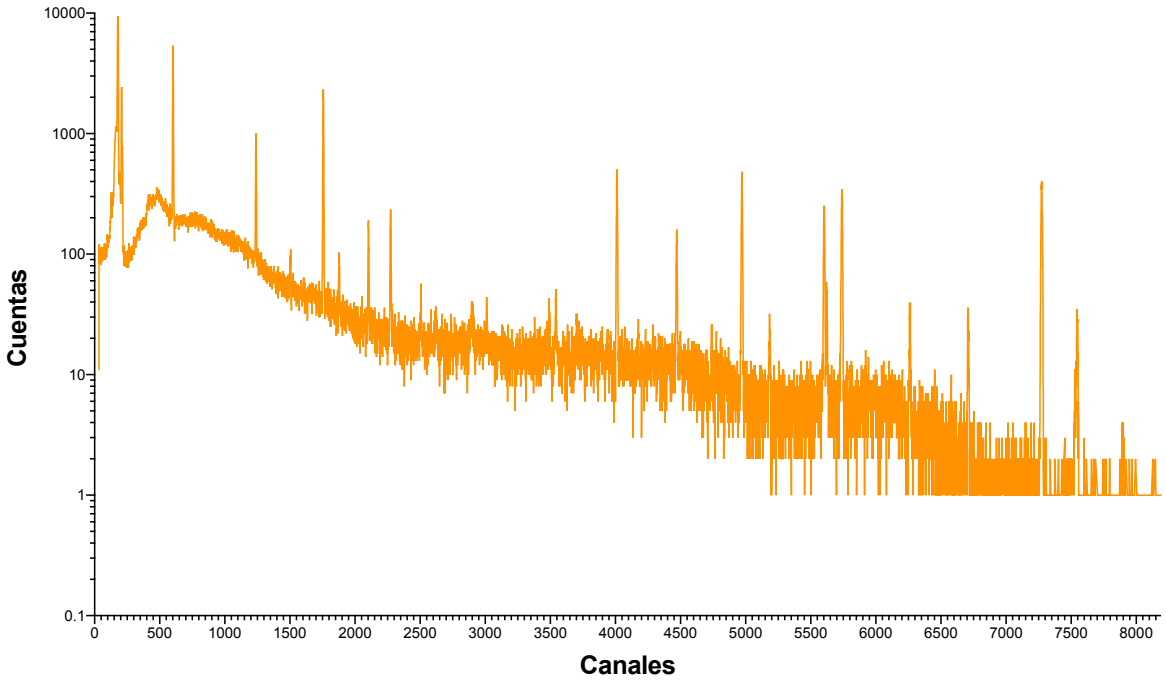


**Figura 4.19. Espectro de Eu 152 con MCA Canberra.**

**Tabla 4.2 Canales y energías de Eu 152.**

| Canal (#) | Energía (MeV) |
|---|---|
| 268 | 0.121 |
| 525 | 0.244 |
| 734 | 0.344 |
| 874 | 0.441 |
| 994 | 0.443 |
| 1646 | 0.778 |
| 1833 | 0.867 |
| 2035 | 0.963 |
| 2292 | 1.084 |
| 2348 | 1.112 |
| 2970 | 1.408 |
| 3074 | 1.457 |

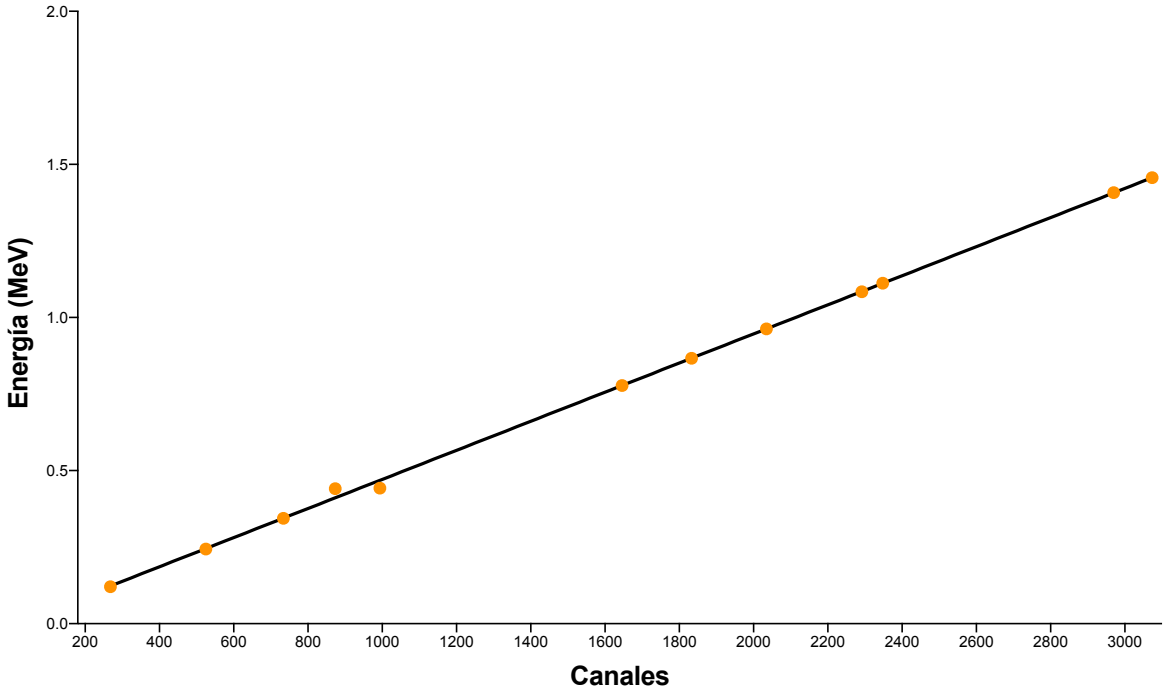


**Figura 4.20. Calibración con Eu 152.**

# Conclusiones

Se logro determinar las señales provenientes de los detectores de radiación mediante un convertidor analógico digital el cual tiene una resolución de 12 bits. Se consiguió diseñar e implementar de un analizador multicanal para espectrometría nuclear embebido en un FPGA, el cual se empaqueto como un núcleo de propiedad intelectual o IPcore. Se hizo uso de uno de los núcleos ARM para el control del hardware embebido y la trasmisión de los datos al exterior de la placa de desarrollo y así poder comunicarse con el instrumento virtual. Se creo un instrumento virtual mediante la plataforma de desarrollo grafico LabVIEW, el cual sirve para el majeo, control del sistema y visualización del espectro, dicho instrumento es una interfaz amigable para el usuario el cual es muy sencillo de operar. Para el analizador multicanal solo fue necesario ocupar el 9% de la capacidad total del FPGA, por lo cual es factible la colección de otras funcionas de alto nivel embebidas, para mejorar el funcionamiento y representación, en el propio dispositivo reconfigurable. Teniendo una analizador de espectro embebido en un FPGA se pueden cumplir los requerimientos de espacio, volumen y peso para colocarlo como carga útil junto a un diodo pin y un amplificador espectroscópico en un nano satélite de construcción mexicana para proporcionar espectros tomados fuera de la atmosfera. Se han hecho pruebas con cesio 137, cobalto 60 y europio 152 para demostrar que el analizador esta funcionando y se realizaron comparaciones con analizadores de Ortec y Canberra. Mediante los espectros obtenidos de las fuentes de cesio 137, cobalto 60 y europio 152, fue posible obtener una recta de calibración haciendo uso de un método de regresión lineal para relacionar los canales del analizador multicanal con sus energías correspondientes. Como trabajo futuro, se puede mejorar el diseño de las rutinas para el procesamiento de los datos como, encontrar los fotopicos y calcular el área bajo la curva de ellos, adecuación del número de canales según las necesidades de el usuario, añadir corrección por deslizamiento a la grafica, probar un conversor análogo digital con una frecuencia de reloj mas elevada para intentar eliminar la necesidad de utilizar una amplificador espectroscópico y hacer manipulación del analizador multicanal desde dispositivos móviles.

# Referencias bibliográficas


ADA PMOD XILINX, F., ADA, I. I. O. & DRIVER, L. 2011. IMPORTANT LINKS for the AD7476A_7477A_7478A.

ADLER, F., THORPE, M. J., COSSEL, K. C. & YE, J. 2010. Cavity-enhanced direct frequency comb spectroscopy: technology and applications. *Annual Review of Analytical Chemistry,* 3**,** 175-205.

AGUDELO, M. O. A. & VALDERRAMA, F. M. A. 2016. Exploración de técnicas de alto nivel en hardware para el desarrollo de redes neuronales.

ÁLVAREZ, R. Á. 2016. Co-diseño hardware-software de un sistema de posicionamiento basado en procesado de vídeo.

APU, A. P. U. 2016. Zynq-7000 All Programmable SoC Overview.

ARAMBURU, X. O. & BISBAL, J. J. 1996. *Las radiaciones ionizantes: su utilización y riesgos*, Univ. Politèc. de Catalunya.

ARCE, J. E. L., RAYGOZA, J. J., CISNEROS, S. O., RIVERA, J. & VILLALOBOS, P. M. 2016. Arquitectura genérica de una red en chip de enrutamiento unidireccional en FPGA. *Pistas Educativas,* 36.

ASHENDEN, P. J. 2010. *The designer's guide to VHDL*, Morgan Kaufmann.

BARRERA, M., ROMERO, L. & VALIÑO, F. 2008. *Puesta a punto de un sistema de espectrometría gamma para la determinación de Cs-137 en suelos españoles*, CIEMAT.

BHASKER, J. 1999. *A Vhdl primer*, Prentice-Hall.

BRANDAN, M. E., PERCHES, R. D. & OSTROSKY, P. 1998. *La radiación al servicio de la vida*, Fondo de Cultura Económica.

CASTRO, M. D., RAMÍREZ, D. R. & PINO, N. L. 2003. DESARROLLO DE UN ANALIZADOR MULTICANAL UTILIZANDO UNA TARJETA CON PROCESADOR DE SEÑALES DIGITALES. (Spanish). *DEVELOPMENT OF A MULTICHANNEL ANALYZER USING A DATA ACQUISITION CARD WITH DIGITAL SIGNAL PROCESSOR. (English)***,** 48-52.

CEBALLOS, N. D. M., DELGADO, N. A. & ARROYAVE, M. 2015. Sistemas de Teleoperación de Robots Basado en Internet.

COLODRO, C., TOLEDO, J., MARTÍNEZ, J. J., GARRIGÓS, J. & FERRÁNDEZ, J. M. 2012. Evaluación de algoritmos de correspondencia estereoscópica y su implementación en FPGA. *Jornadas de Computación Reconfigurable y Aplicaciones (JCRA)***,** 88-93.

CROCKETT, L. H., ELLIOT, R. A., ENDERWITZ, M. A. & STEWART, R. W. 2014. *The Zynq Book: Embedded Processing with the Arm Cortex-A9 on the Xilinx Zynq-7000 All Programmable Soc*, Strathclyde Academic Media.

DAMBACHER, M., ZWERGER, A., FAULER, A., DISCH, C., STÖHLKER, U. & FIEDERLE, M. 2011. Development of the gamma-ray analysis digital filter multi-channel analyzer (GMCA). *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment,* 652**,** 445-449.

DEL MONTE, E., RUBINI, A., BRANDONISIO, A., MULERI, F., SOFFITTA, P., COSTA, E., DI PERSIO, G., DI COSIMO, S., MASSARO, E. & MORBIDINI, A. Silicon Photomultipliers as readout elements for a Compton Effect polarimeter: the COMPASS project. SPIE Astronomical Telescopes+ Instrumentation, 2016. International Society for Optics and Photonics, 991528-991528-13.

DELGADILLO, J. J. M. 2003. Interacción de partículas cargadas con materia.

DIGILENT 2016. PmodAD1 Reference Manual.

DOBAI, R. & SEKANINA, L. Towards evolvable systems based on the Xilinx Zynq platform. Evolvable Systems (ICES), 2013 IEEE International Conference on, 2013. IEEE, 89-95.

GALINDO, G. F. J. 2015. Evaluación e implementación de algoritmos de análisis de la marcha sobre dispositivos configurables.

GARCÍA, L. 2016. Implementación en Hardware de un Entrelazador/Desentrelazador de Datos con Interface AXI Stream. *Difu100ci@ Revista en Ingeniería y Tecnología, UAZ,* 10.
GARCÍA, T. E. 2006. Current status of alpha-particle spectrometry. *Applied radiation and isotopes,* 64**,** 1273-1280.
GETMAN, L. 2011. Creating the Xilinx Zynq-7000 Extensible Processing Platform. *EE times. com*.
GLASSTONE, S. & SESONSKE, A. 1990. *Ingeniería de reactores nucleares*, Reverté.
GONZÁLEZ, G. J. 2014. Caracterización computacional de un detector de germanio hiperpuro de rango extendido (HPGe-XtRa) con simulación Montecarlo y optimización con un algoritmo evolutivo.
GONZÁLEZ, R. R. 2016. Plataforma para realización de rutinas de vuelo autónomo sobre cuadricóptero.
GOÑI ESPARZA, M. 2015. Desarrollo del software de control de los sistemas de posicionamiento de precisión para medida de antenas en milimétricas y submilimétricas.
GYULASSY, M. & WANG, X.-N. 1994. Multiple collisions and induced gluon Bremsstrahlung in QCD. *Nuclear Physics B,* 420**,** 583-614.
HERNÁNDEZ, J. M. S., BUSTAMANTE, R. A. G., VALENCIA-PALOMO, G., MONTAÑO, J. A. H., ABEL GARCÍA BARRIENTOS, J. & FONTES, M. T. 2015. Diseño, Construcción e Implementación de un Controlador de Motores de CD Utilizando un FPGA. *Su objetivo principal es difundir los avances en investigación a nivel posgrado y licenciatura en diversas áreas de la ingeniería, realizados en las instituciones de educación superior del Estado de Sonora.***,** 45.
HERNÁNDEZ, O. E. 2015. Teoría básica para seguridad y protección radiológica.
HUESCA, H. B. 2015. Cálculo de blindajes para atenuar la radiación electromagnética gamma.
IBARRA, C. J. & PABÓN, V. M. 2015. ESPECTROMETRÍA GAMMA DE BAJA RESOLUCIÓN ANALIZADOR DE 1024 CANALES LUDLUM.
JIMENEZ, F. J., LARA, F. R. & REDEL, M. D. 2014. API for communication between Labview and Arduino UNO. *IEEE Latin America Transactions,* 12**,** 971-976.
JÚDEZ, Ó. T. & LÓPEZ, S. C. 2013. Curso breve para docencia en Ingeniería Industrial. Iniciación al transporte de radiación ya su interacción con materiales.
JULIÁN MORENO, J., ESPERANZA CELY, N., FRANCISCO RODRÍGUEZ, J. & FERNEY ANGARITA, É. 2014. Automatización de un banco de mediciones para caracterización a gran señal. *Sistemas & Telemática,* 12.
LEE, P. S., LEE, C. S. & LEE, J. H. 2013. Development of FPGA-based digital signal processing system for radiation spectroscopy. *Radiation Measurements,* 48**,** 12-17.
LEMA, S. A. D. 2016. Desarrollo mediante lenguaje de alto nivel de un sistema de simulación de redes basado en FPGA para aplicaciones multiGbps Ethernet.
LOZA, P. M. & ELIAS, F. G. A. 2010. DISE NO DE UN OSCILOSCOPIO MULTICANAL CON FPGA (PROYECTO LAGO) THE DESIGN AND CONSTRUCTION OF A FPGA MULTI CHANNEL OSCILOSCOPE (LAGO PROJECT). *REVISTA BOLIVIANA DE Fisica,* 16**,** 27-31.
MARTÍNEZ, C. C. 2015. Calibración e intercomparación de métodos de determinación de radón en agua.
MOLINA, J. C., LÓPEZ, M., BERMÚDEZ, J. C. & HUERTA-RUELAS, J. A. Determinación de la reflectancia angular en fotodetectores con luz polarizada. Simposio Metrología, 2010.
MONEDERO, M. P. & LÓPEZ, J. J. V. 2012. Evaluación de semiconductores como detectores de radiación para PET. Detectores de CZT. *Trabajo de Máster en Física Biomédica*.
MORA MIRANDA, R. V. 2016. Simulino connected to LabView through ethernet. *Journal Boliviano de Ciencias,* 12**,** 50.
MORENO, J. A. T., GARCÍA, J. A. C. & ESCOBAR, F. E. M. 2017. Descriptivo y análisis de las radiaciones ionizantes en la ciudad de Tacna. *Ciencia & desarrollo*.

ORAMAS, P. I. & FIGUEROA, D. V. D. G. 2014. Análisis de riesgos para el trabajo con un analizador de diagnóstico de cámaras gamma. *Nucleus***,** 19-23.
ORTEGA, L. S. 2014. Implementación de periféricos en vivado para dispositivos ZYNQ.
PAVÓN, L. J. 2015. Movimiento eficiente de datos para un SOC basado en ZYNQl.
PEDRAZA, R. J. L. 2013. Introducción a la seguridad radiológica.
PELLEGRINI, G., QUIRION, D., RODRÍGUEZ-CARUNCHIO, J., CRUZ, C., JOVER, G., ÁVILA ABELLÁN, J. & MATILLA BARCELÓ, Ó. 2015. Detector de radiación transmisivo.
RAMIREZ, J., PROAÑO, V. & ALULEMA, D. Basic Programmable Logic controller with FPGA and Artificial Neural Networks. Communications and Computing (COLCOM), 2015 IEEE Colombian Conference on, 2015. IEEE, 1-4.
RAMIREZ, J. N. F. J. 2010. Nuclear electronic instrumentation. *Contributions of the National Institute of Nuclear Research to the advance of science and technology in Mexico. Commemorative edition 2010.*
REYES, L. A. 2015. *Transmisor Digital de Datos con Modulación BPSK mediante Tecnología FPGA*.
RÓBERT, K., PÉTER, M., ZOLTÁN, Z. & LÁSZLÓ, H. 2015. Selecting the optimal anti-aliasing filter for multichannel biosignal acquisition intended for inter-signal phase shift analysis. *Physiological Measurement,* 36**,** N23.
ROTH JR, C. H. & JOHN, L. K. 2016. *Digital systems design using VHDL*, Cengage Learning.
SAGLAM, Z., SAHIN, G. & BOYACIOGLU, B. 2016. Compton effect in terms of spintronic. *Results in Physics,* 6**,** 726-727.
SALTOS, G. M. E., PUNINA, Y. & RODRIGO, C. 2014. Determinación de Problemas en Equipos Industriales Mediante Análisis Vibracional a Través del Software Labview.
SÁNCHEZ, G. D. 2014. Desarrollo multiplataforma de un PSoC basado en microprocesador ARM para aplicaciones empotradas.
SÁNCHEZ, R. M. 2013. Diseño de bloques analógicos de alta velocidad y técnicas de procesamiento digital para aplicación en detectores de física nuclear.
SOSA, M. A. 2014. Desarrollo de una sonda Ethernet activa basada en el SoC programable Xilinx Zynq.
TAPIA JÚDEZ, Ó. & CUESTA LÓPEZ, S. 2013. Curso breve para docencia en Ingeniería Industrial: Iniciación al transporte de radiación ya su interacción con materiales.
TOBÍAS, I. G., ALFARO, M. N., VILLEGAS, O. O. V., ABAD, A. F. & CABERTA, R. Ñ. 2016. Implementación de una arquitectura para control y verificación de un sistema de teleoperación por medio de LabVIEW. *CULCyT*.
TRUJILLO, G. G. 1998. Control dosimétrico in vivo en radioterapia externa usando diodos semiconductores. *Rev Cubana Oncol,* 14**,** 129-35.
TSOULFANIDIS, N. 2013. *Measurement and detection of radiation*, CRC press.
VÁZQUEZ, A. C. 2014. Implementación de algoritmos CORDIC con Vivado HLS.
VENEGAS, A. Y. 2015. Cálculo de la eficiencia de un detector gamma de rango extendido en matrices ambientales aplicando el método de Monte Carlo.
XILINX, I. 2011. Spartan-3E FPGA Starter Kit Board User Guide.
ZEDBOARD 2014. Zynq Evaluation and Developement Hardware User´s Guide.

# Anexo A: Diseño de la plataforma hardware y software

Este anexo aborda las etapas para la creación de una arquitectura de hardware y la aplicación del software. El anexo está compuesto por 4 secciones: descripción de archivos VHDL, empaquetado de IPcore, desarrollo de la arquitectura de hardware y programación de la aplicación de software. Los programas utilizados para este anexo son Vivado 2015.4 y SDK 2015.4.

## Descripción de archivos VHDL

La descripción de archivos VHDL se hace mediante la plataforma de desarrollo Vivado. Se comienza abriendo la aplicación de Vivado, una vez iniciado el programa se mostrara la Figura 1, se tiene que presionar el recuadro marcado con el numero (1) para crear un nuevo proyecto, posteriormente se desplegara una ventana en la que se tiene que presionar el botón *next* marcado con el recuadro numero (2) para continuar.

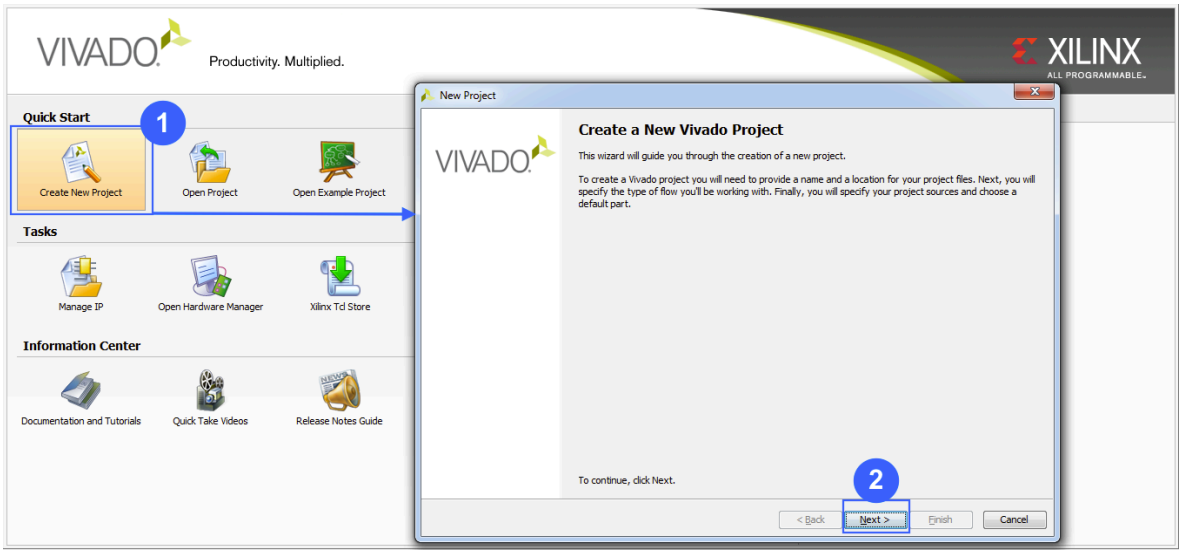


**Figura 1. Creación de nuevo proyecto.**

Después se abre la ventana mostrada en la Figura 2, donde se puede especificar el nombre del proyecto (3) y la dirección en donde se almacenara (4). Posteriormente se presiona el botón *next* (5) para continuar.

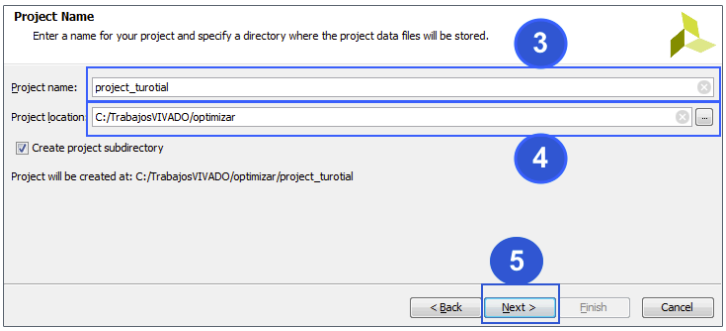


**Figura 2. Nombre de proyecto.**

Seguido a lo anterior aparece la ventana mostrada en la Figura 3, en donde se tiene que seleccionar la opción *RTL Project* y marcar la casilla inferior (6), posteriormente presionar el botón *next* (7) para continuar.

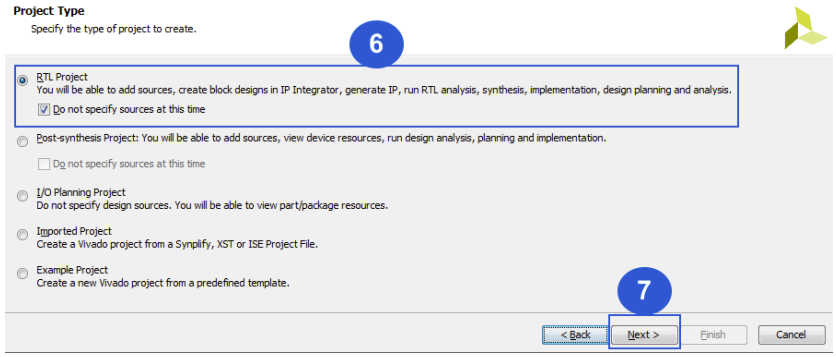


**Figura 3. Tipo de proyecto.**

Continuando con la ventana que aparece en la Figura 4, en la que se selecciona el apartado *Boards* (8) para que se enlisten las diferentes placas de desarrollo, de las cuales se selecciona *ZebBoard* (9) y por ultimo se presiona el botón *next* (10).

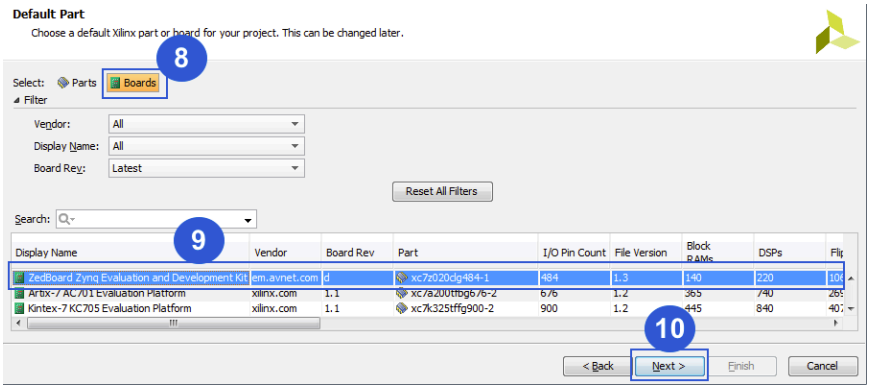


**Figura 4. Selección de placa de desarrollo.**

Se desplegara una ventana en donde se enlistan las características del proyecto y de la tarjeta de desarrollo como lo muestra la Figura 5, aunado a esto se tiene que presionar el botón *finish* (11) para terminar la configuración del proyecto.

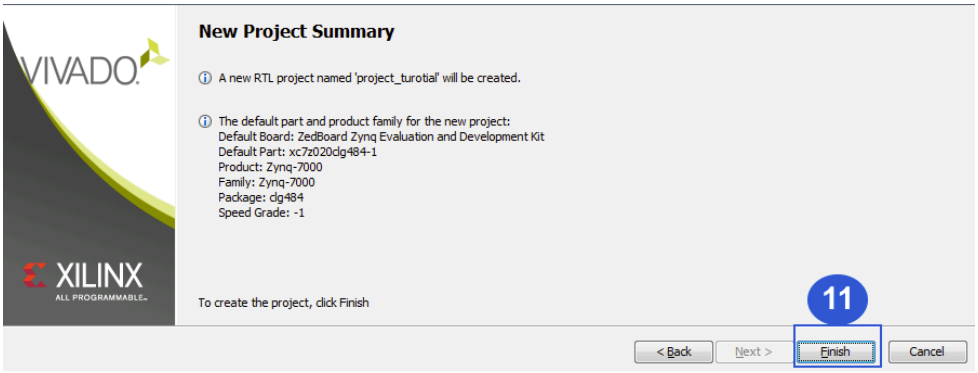


**Figura 5. Resumen del proyecto.**

Entonces aparece la ventana mostrada en la Figura 6 donde se tiene que presionar sobre la opción *Add Source* (12) y así en la ventana emergente marcar la opción de añadir o crear fuentes de diseño (13), ya hecho lo anterior, presionar el botón *next* (14) para proseguir.


**Figura 6. Añadir Fuentes.**

Seguido a esto se visualiza la ventana mostrada en la Figura 7, donde se tiene que oprimir el botón crear archivo (15), después se desplegara una ventana donde se tiene que seleccionar el lenguaje VHDL (16), ingresar el nombre de tu archivo (17) y para continuar presionar el botón *ok* (18).

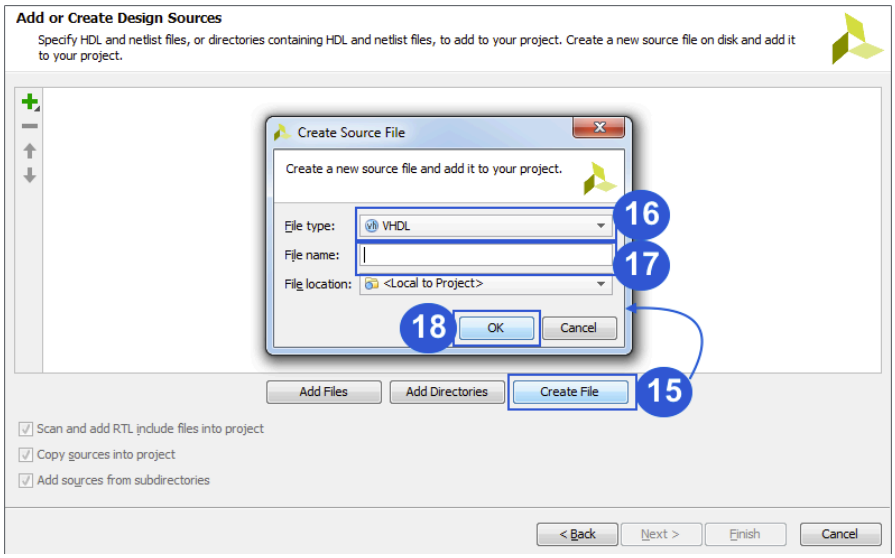


**Figura 7. Crear archivo de fuente.**

En la ventana que se muestra en la Figura 8, se tiene que presionar el botón finalizar (19) para terminar con la creación de archivos.

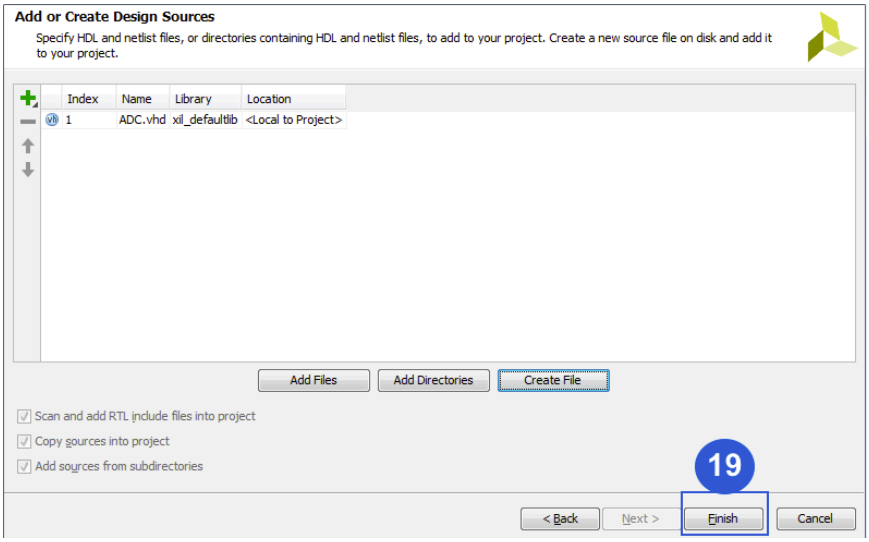


**Figura 8. Añadir o crear fuentes.**

Posteriormente se despliega la ventana mostrada en la Figura 9, donde se puede modificar el nombre de la entidad (20), el nombre de la arquitectura (21) y definir puertos de entrada y salida (22). Para concluir se presiona el botón *ok* (23).

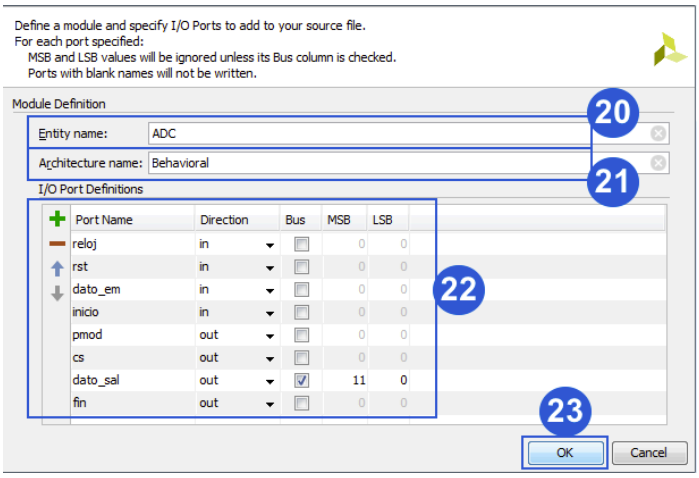


**Figura 9. Configuración de puertos.**

En la ventana principal que se muestra en la Figura 10, en donde se describe el archivo en lenguaje VHDL en el cuadro de dialogo (24).

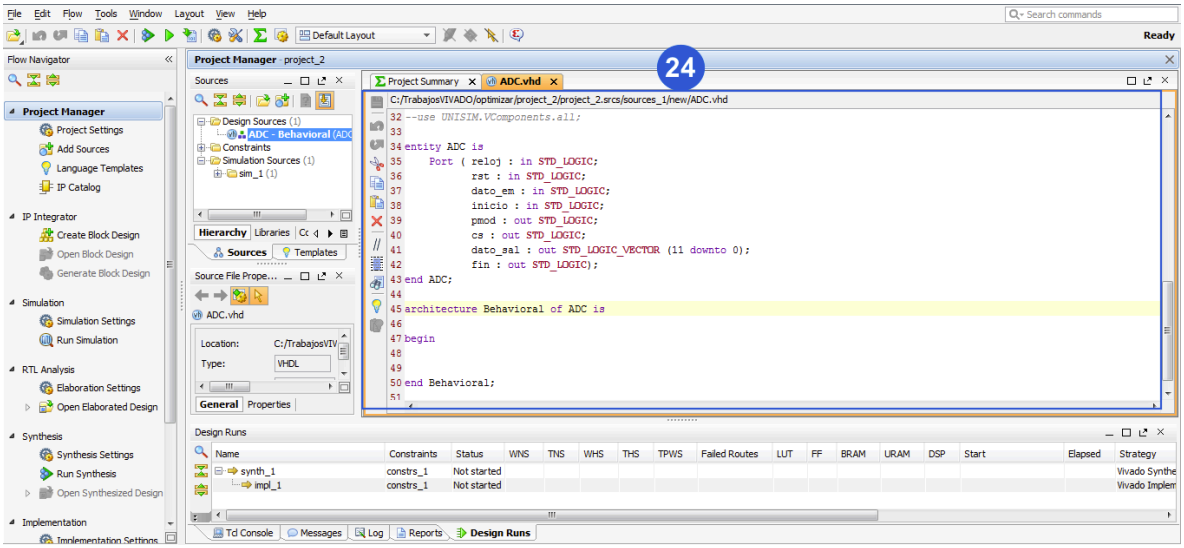


**Figura 10. Código VHDL.**

## Empaquetado del IPcore

Para empaquetar los archivos VHDL en un núcleo de propiedad intelectual (IPcore) se tiene que seguir las instrucciones de la sección *Descripción de archivos VHDL,* hasta el paso numero 14 visualizado en la Figura 6, seguido a esto se podrá observar la Figura 11, donde se tendrá que activar las opciones (15), después presionar el botón añadir archivos (16), localizar

los archivos VHDL requeridos y presionar el botón *ok*, dichos archivos se enlistaran (17). Para terminar el proceso de añadir archivos VHDL se presiona el botón *finish* (18)

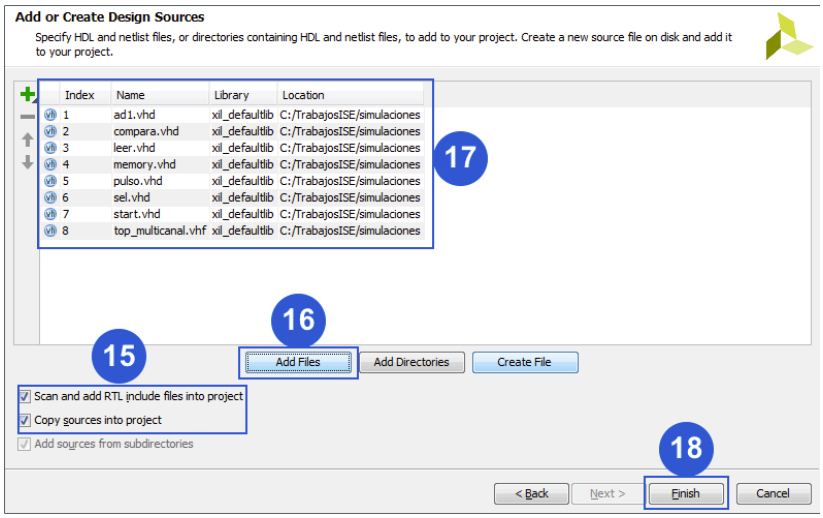


**Figura 11. Añadir o crear fuentes para empaquetado.**

Aunado a esto y como se muestra en la Figura 12, en la barra de menú seleccionas la opción *tools* (19) seguido de la selección de crear y empaquetar IPCore (20).

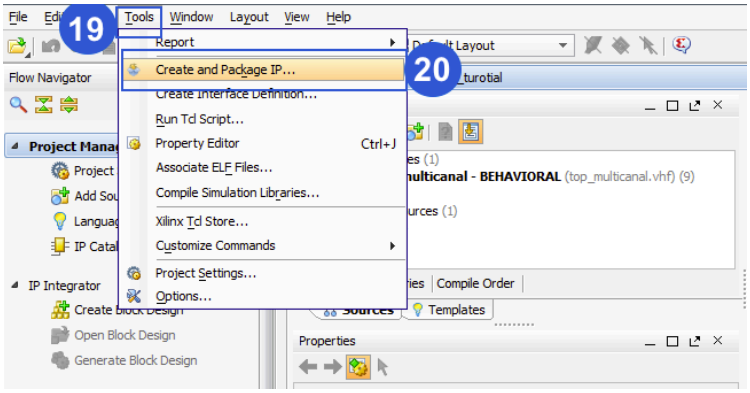


**Figura 12. Crear y empaquetar ip.**

En la ventana mostrada en la Figura 13, se presiona el botón *next* (21) para continuar con la creación del IPcore

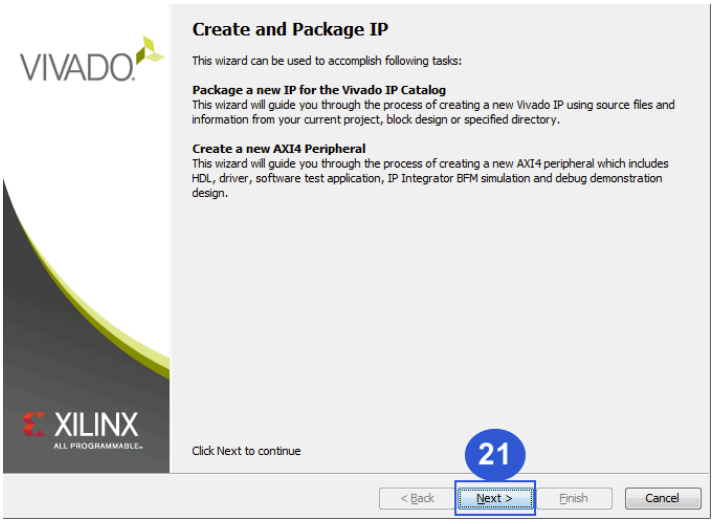


**Figura 13. Descripción del empaquetado.**

Después se selecciona la opción empaquetar su actual proyecto (22) como lo muestra la Figura 14, seguido a esto se oprime el botón *next* (23) para continuar.

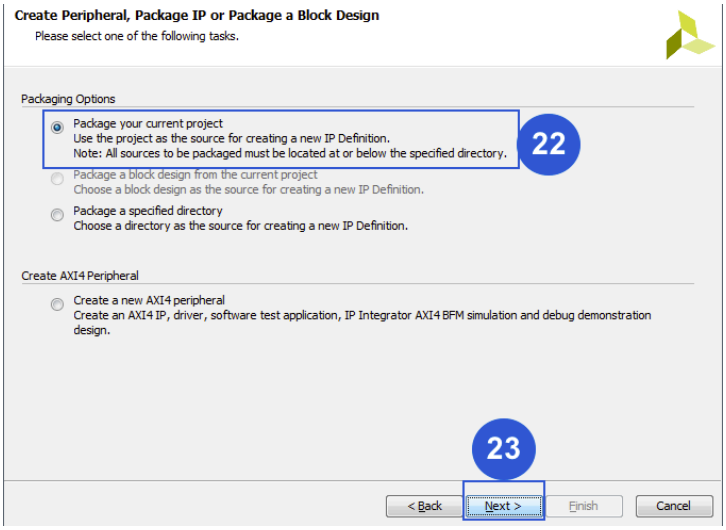


**Figura 14. Opciones de empaquetado.**

Inmediatamente aparece la ventana mostrada en la Figura 15, en donde se selecciona la ubicación para guardar la IPcore (24), además se elige la opción incluir archivo .xci (25), después presionar el botón *next* (26) y en la siguiente ventana oprimir *finish* para proseguir.

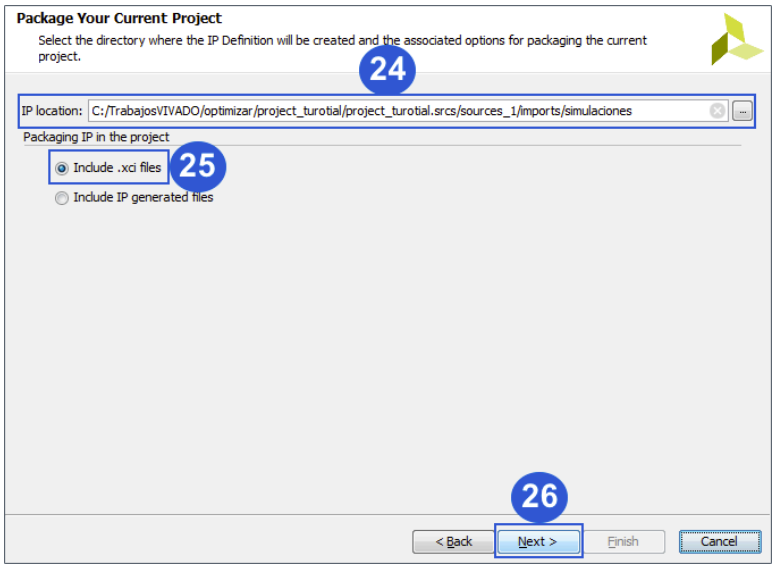


**Figura 15. Empaquetado del proyecto actual.**

En la ventana revisión y empaquetado aparecen las características del IPcore como lo muestra la Figura 16, una ves ahí se selecciona la opción *review and package* (27) y después activar el link editar configuración de empaquetado (28).

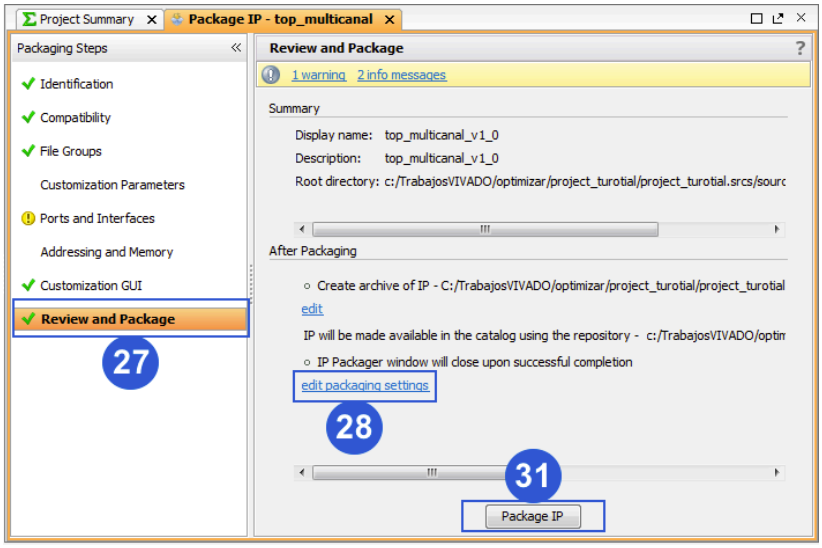


**Figura 16. Revisión del empaquetado.**

En la ventana que se despliega, marcar las tres casillas de la sección de después del empaquetado (29) como lo muestra la Figura 17, para después presionar el botón *ok* y volver a la Figura 16 donde se tendrá que oprimir el botón *Package IP* (31), para así terminar con la creación y empaquetado del IPcore.

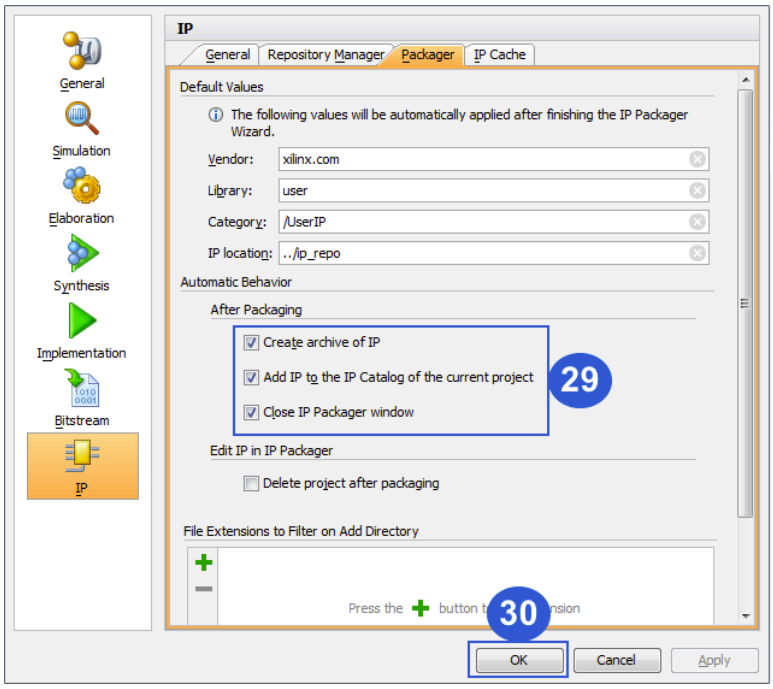


**Figura 17. Ajustes del empaquetado.**

# Desarrollo de la arquitectura de hardware

Esta sección se inicia siguiendo todos los pasos de la sección *Descripción de archivos VHDL,* hasta el paso 11 de la Figura 5 de la sección, continuando así con la Figura 18. En esta ventana se seleccionara *Create Block Design* (12), en la ventana emergente se especificara el nombre deseado para el bloque (13) y después se activa el botón ok (14).

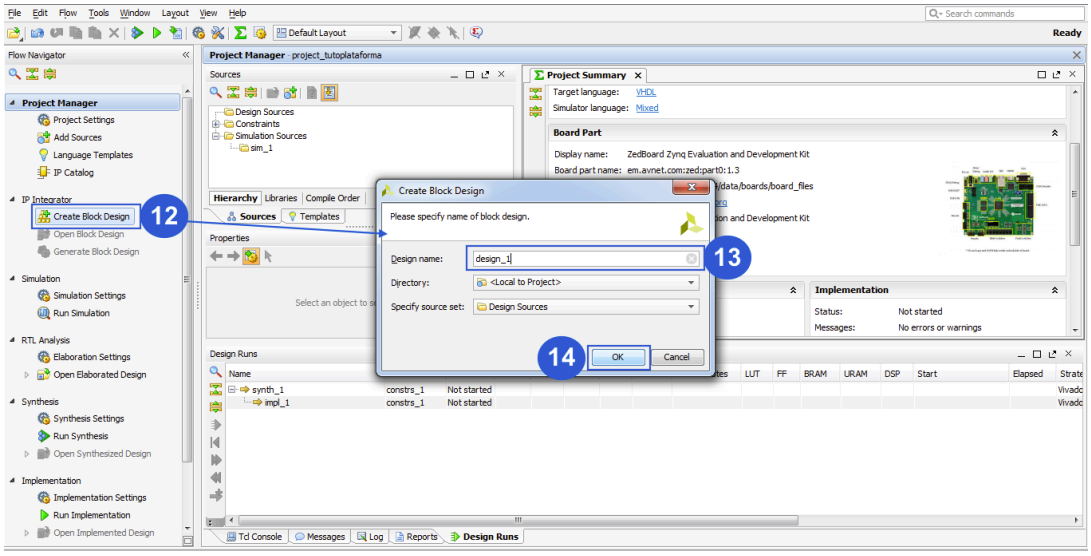


**Figura 18. Crear el bloque de diseño.**

Posteriormente se presiona el botón add IP (15), después se escribe la palabra zynq en el cuadro de búsqueda (16) y se selecciona la primera opción (17).

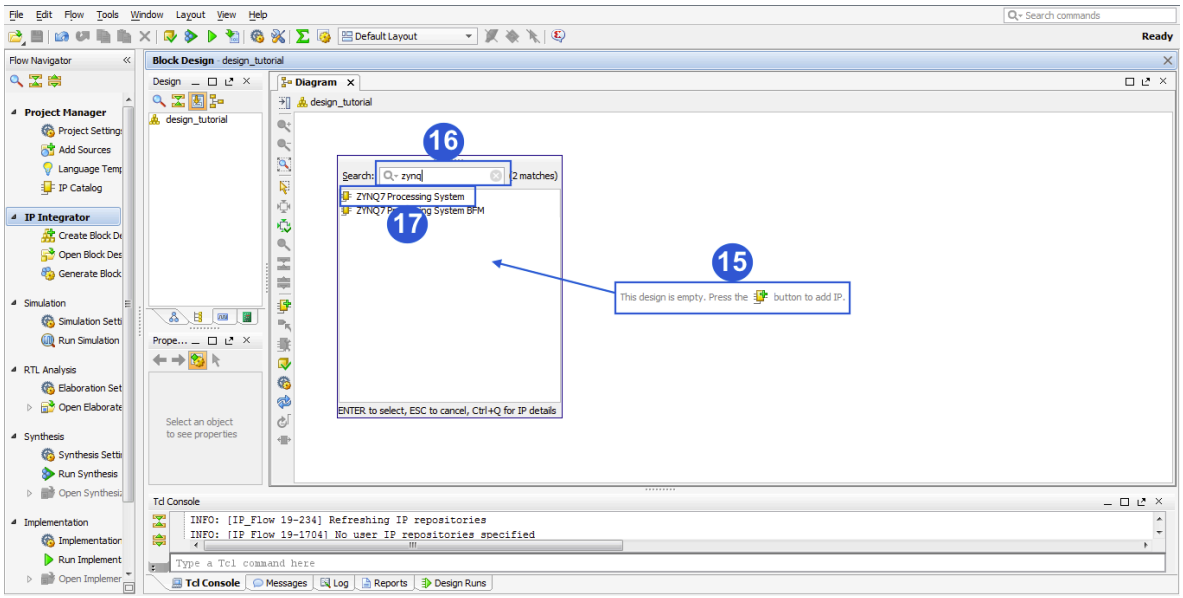


**Figura 19. Añadir bloque Zynq.**

Ya que se visualice el bloque añadido, se activa el link *Run Block Automation* (18) y en la ventana emergente se selecciona el botón *ok* (19) para arrancar el bloque. Esto se puede observar en la Figura 20.

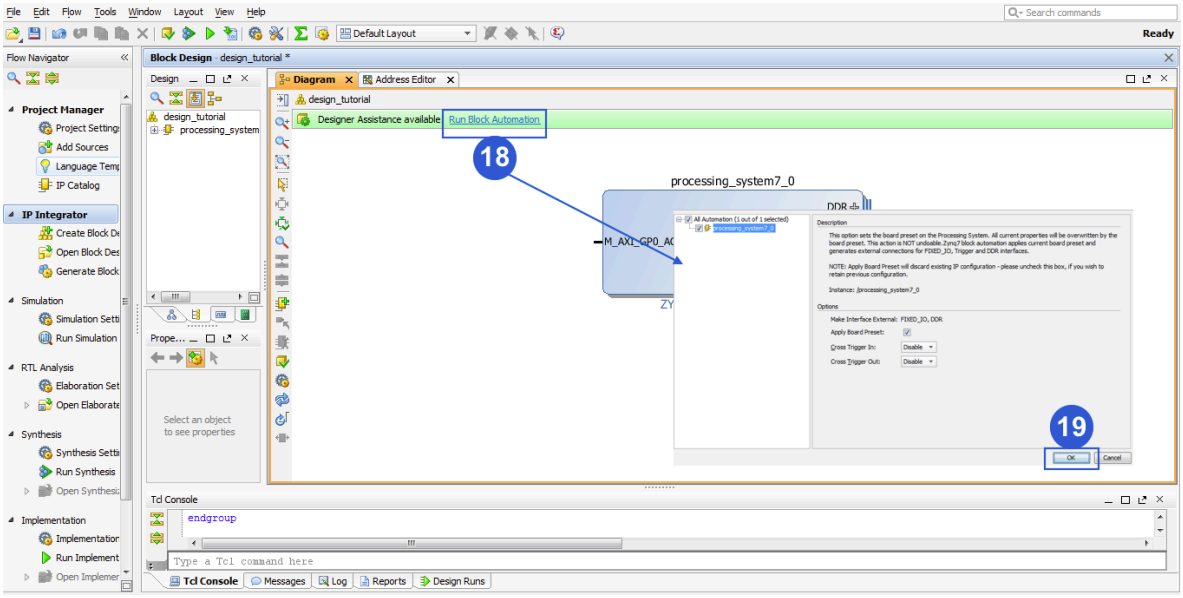


**Figura 20. Arrancar el bloque automáticamente.**

Para continuar, como se observa en la Figura 21 se agregan los bloque gpio´s, seleccionando la opción añadir ip (20), en la ventana emergente se escribe gpio (21) y se

selecciona la opción *AXI_GPIO* (22), estos pasos se tienen que repetir 6 veces para agregar 6 bloque iguales.

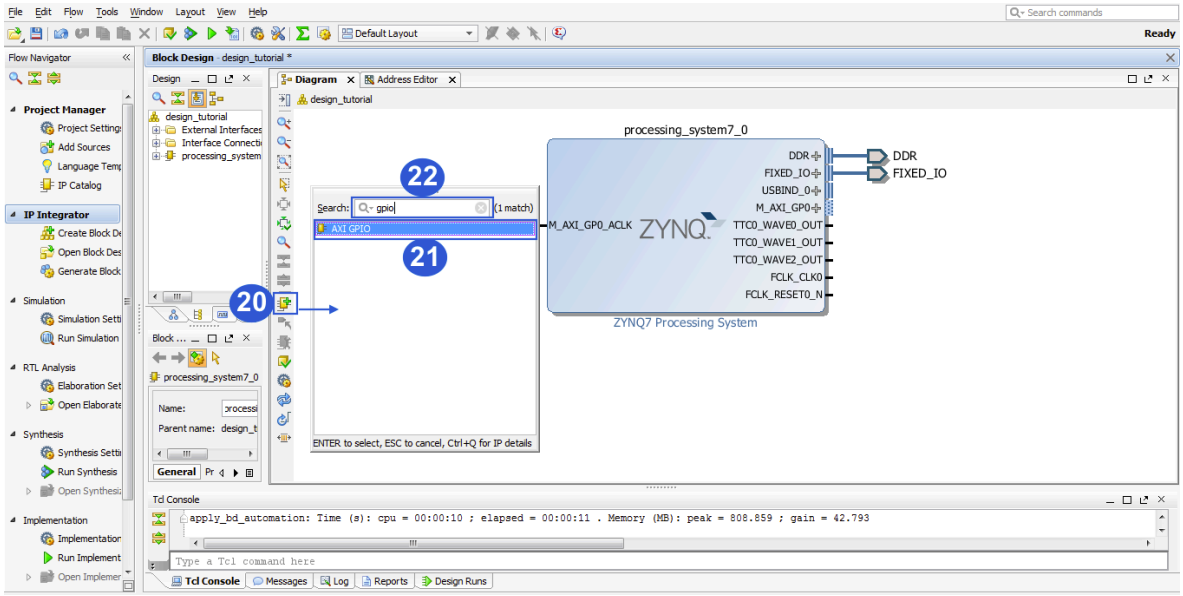


**Figura 21. Añadir bloques GPIO.**

Después, como se muestra en la Figura 22, se añade el IPcore creado en la sección anterior, para esto se presiona configuración de proyecto (23) y en la ventana emergente se selecciona la pestaña manejador de repositorio (24), enseguida se presiona el signo más (25) una vez hecho esto, se busca la dirección donde se guardó el IPcore y se añade, posteriormente se oprime el botón ok (26).

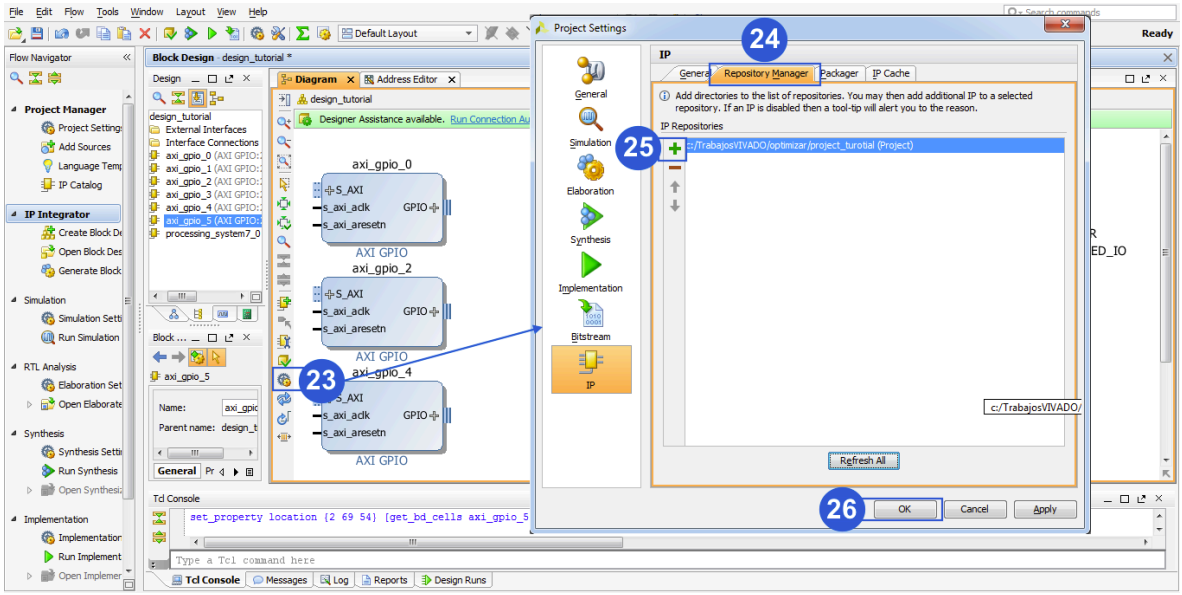


**Figura 22. Agregar IPcore.**

Enseguida se repiten los pasos 20, 21 y 22 de la Figura 21 para añadir el IPcore creado buscándolo por nombre, en la Figura 23 se muestra como añadir los puertos de entrada y salida a la entidad multicanal, seleccionando los puertos (27, 28) y después presionando el botón añadir puerto (29).

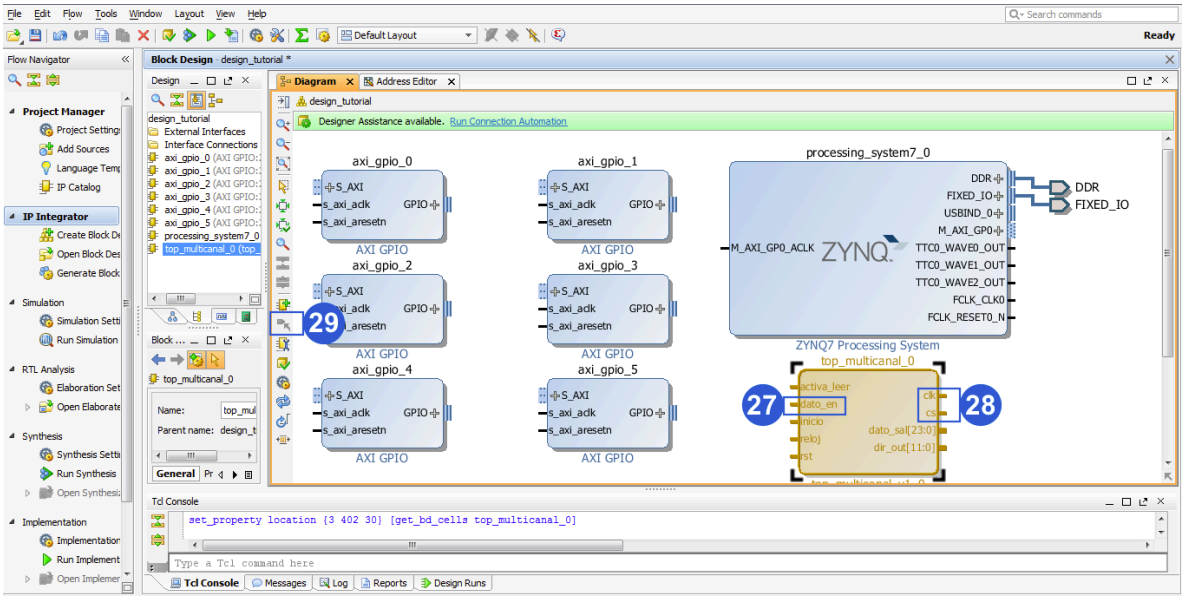


**Figura 23. Asignación de puertos.**

Cada gpio se tiene que configurar adecuándose a las necesidades, en la Figura 24 se muestra la configuración de estos, dando doble click sobre cualquier gpio aparecerán la ventana de ajustes de *axi gpio*, una vez ahí seleccionar la pestaña *IP Configuration* (30), después se selecciona si el periférico es de entrada o salida (31) y después especifica el número de bits a utilizar (32), después oprime el botón ok para continuar (33). Estos pasos se siguen para 5 de los 6 gpio´s los cuales son los que tienen la comunicación con la entidad multicanal, el sexto gpio se configura como lo indica la Figura 25, de igual manera para ingresar a la ventana de justes *axi gpio*, se despliega el menú (34) y se selecciona la opción *leds 8bits* y este gpio es utilizado como indicador encendiendo y apagando los 8 leds (35).

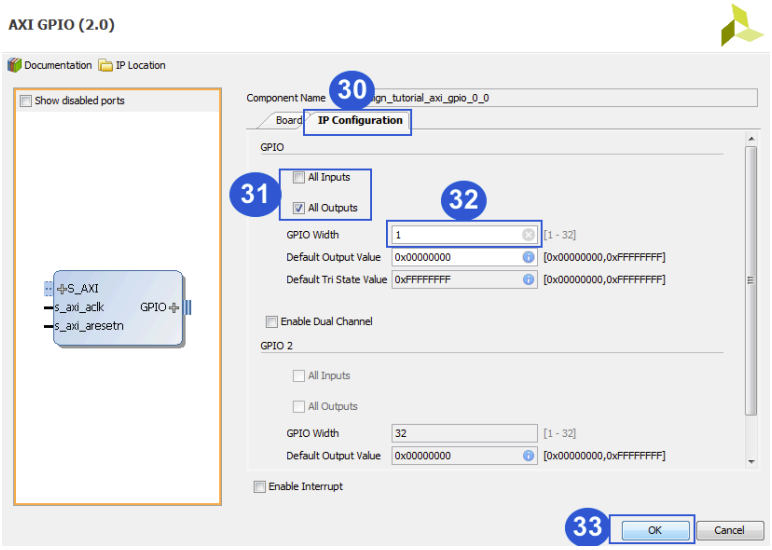


**Figura 24. Configuración de GPIO's.**

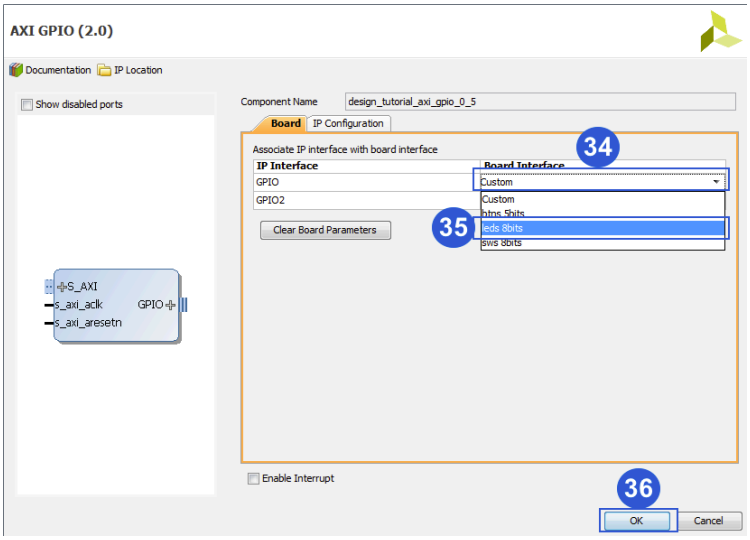


**Figura 25. Conexión de GPIO con led's.**

Una vez configurados los gpio´s, se pasa a conectar los 5 primeros a sus entradas y salidas respectivas de la entidad multicanal (37) como lo muestra la Figura 25, después se activa el link *Run Connection Automation* (38) y en la ventana que se despliegue seleccionar todas las casillas (39), para terminar presionar el botón *ok* (40).

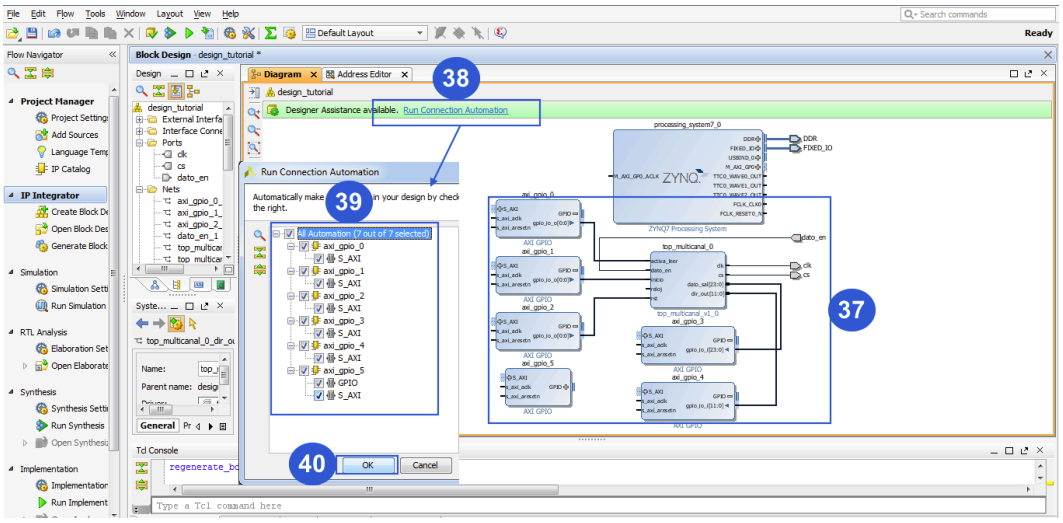


**Figura 26. Auto conectar bloques.**

Para concluir con las conexiones, la entrada reloj de la entidad multicanal se conecta al reloj del sistema como lo muestra en la Figura 27, después se ingresa a la configuración del chip Zynq dando doble click sobre su bloque (42).

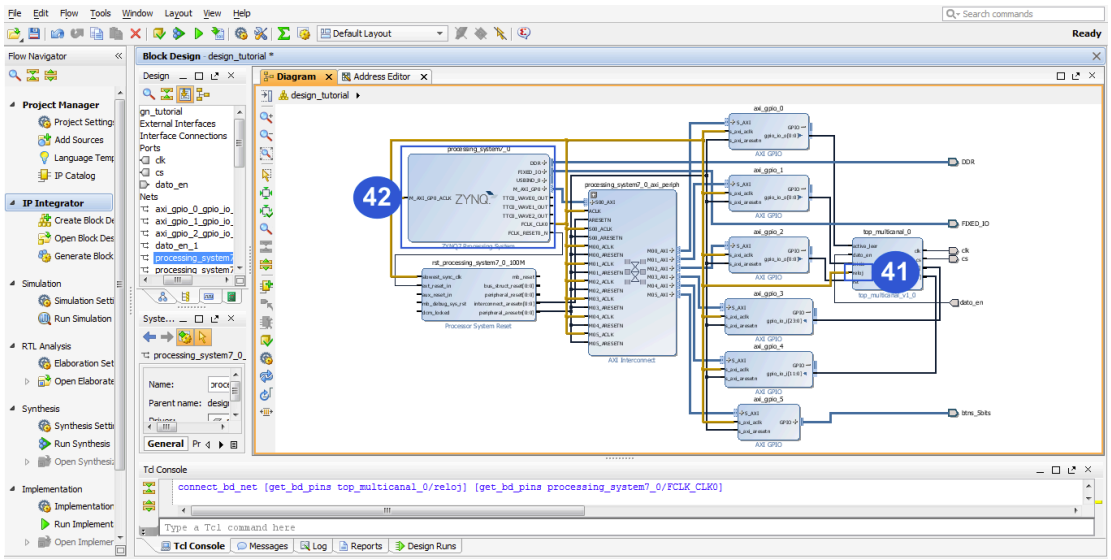


**Figura 27. Conexión de reloj de sistema.**

Dentro de la configuración del Zynq, se ajusta la frecuencia de reloj seleccionando la pestaña *clock configuration* (43), después se activa el primero de los cuatro osciladores, se configura a 20 MHz (44) y para terminar se oprime el botón ok (45).

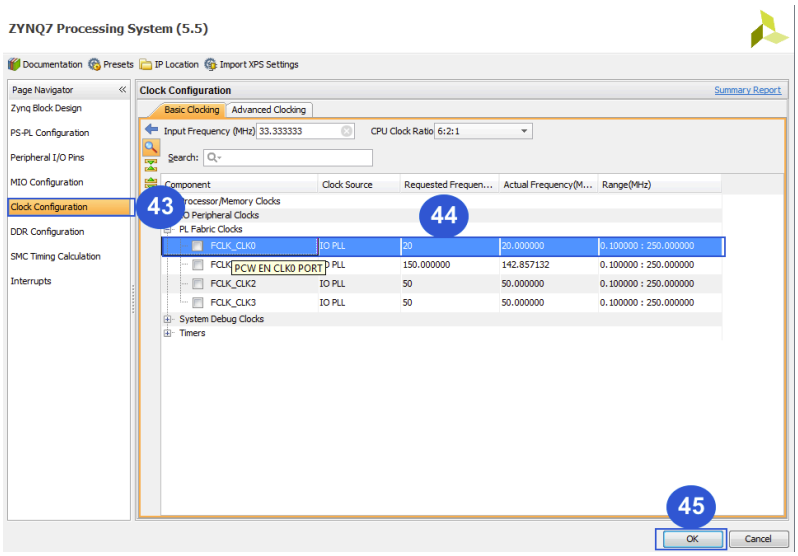


**Figura 28. Configuración de reloj.**

Para conectar los puertos que van dirigidos hacia un pin de la tarjeta de desarrollo, se tiene que hacer un archivo de restricciones donde se indican a donde van conectados. En la Figura 29 se muestra como añadir un archivo de restricciones, iniciando por presionar el botón añadir fuentes (46), seguido de seleccionar *Add or create constraints* (47), después se oprime el botón *next* (48), en la ventana emergente que se visualiza, se presiona el botón añadir archivo (49) y se busca donde este guardado dicho archivo de restricciones, por ultimo se oprime el botón *finish* (50) y se observa el archivo de restricciones (51).

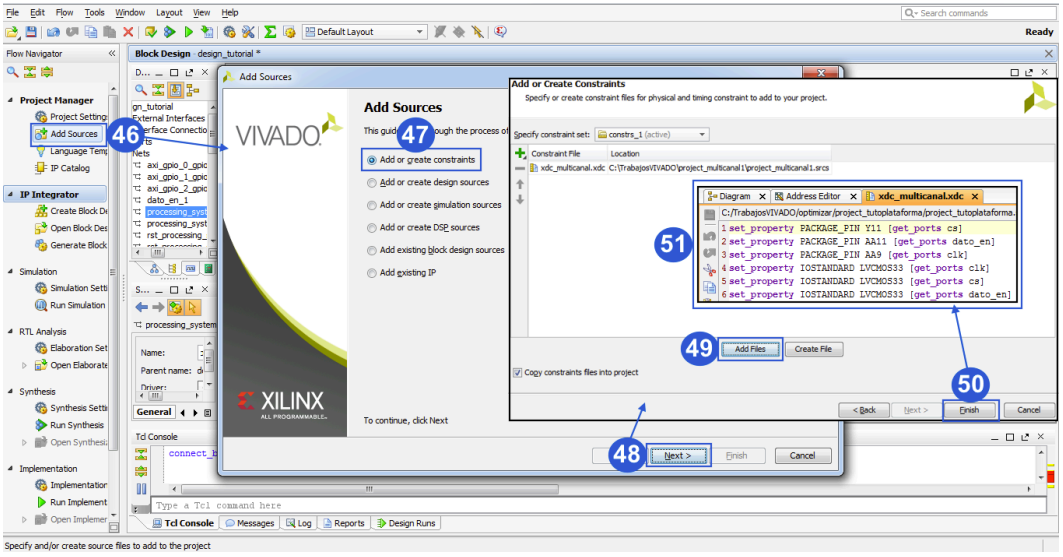


**Figura 29. Añadir restricciones.**

Para comprobar que todas las interconexiones del proyecto estén correctas se valida el diseño, para esto se presiona el botón validar diseño (52) y una vez terminado, oprimir el botón ok (53), esto se ve ilustrado en la Figura 30.

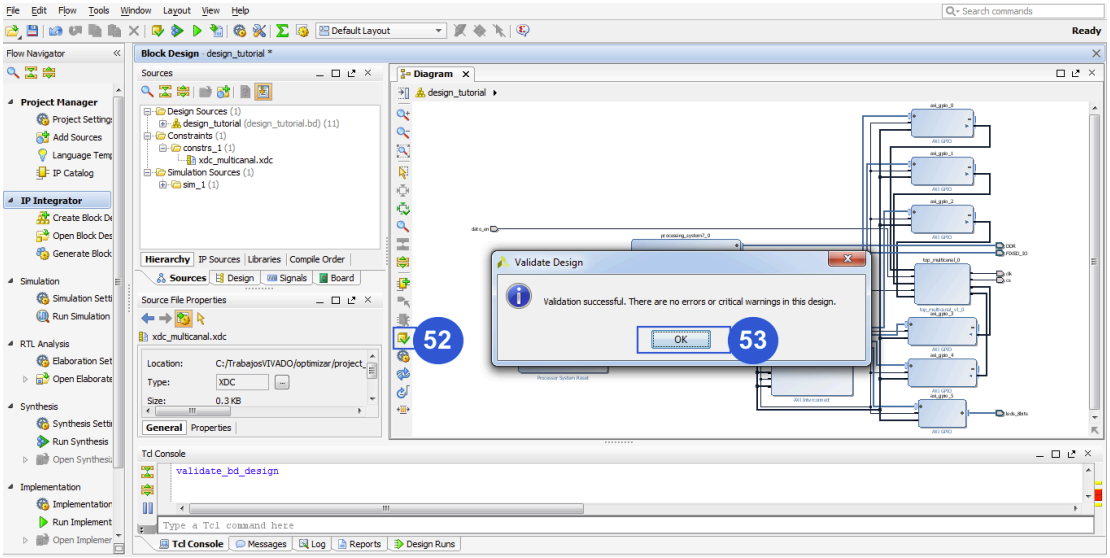


**Figura 30. Validar el proyecto.**

Después como lo muestra la Figura 31 se pasa a la creación del código HDL, para esto se selecciona la pestaña fuentes (54), se selecciona y se da click derecho sobre el diseño del proyecto (55), en el menú desplegado se selecciona *Crate HDL Wrapper* (56), en la ventana emergente se elije la opción para que vivado actualice solo el HDL (57) y para terminar se presiona el botón ok (58).

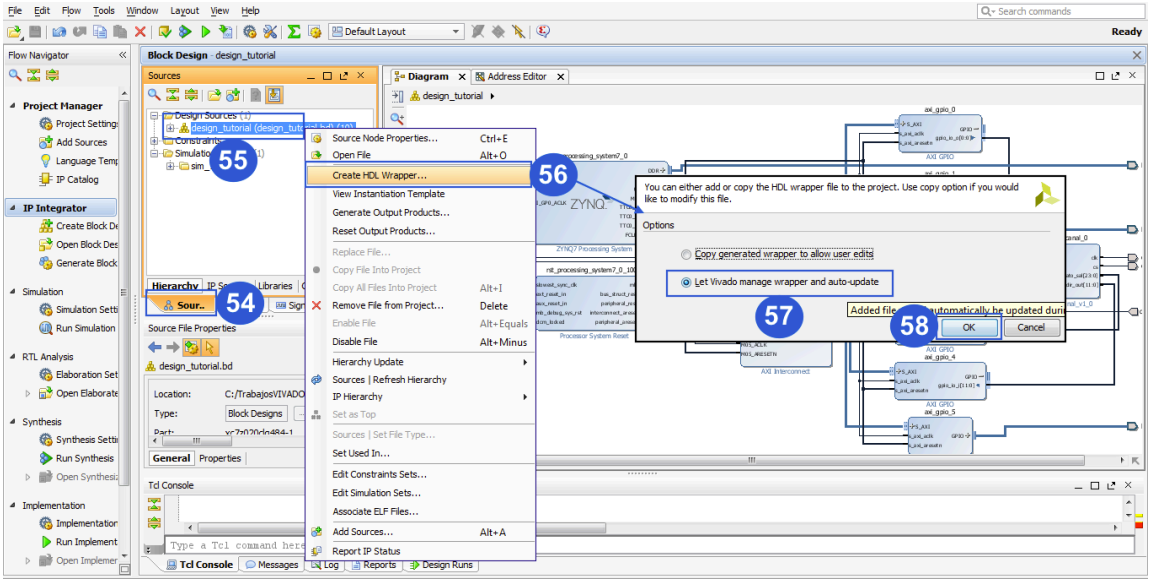


**Figura 31. Crear HDL.**

Posteriormente se tiene que generar el archivo bitstream, para esto se presiona la etiqueta *Generate Bitstream* (59), se despegara una nueva ventana que advierte que el proyecto no esta implementado y ahí se selecciona el botón *Yes* (60), para que antes de que genere el bitstream, sintetice e implemente el proyecto, una vez terminada la generación de bitstream se desplegara una nueva ventana donde se tiene que seleccionar la opción *Open Implemented Disign* (61) y presionar el botón ok para terminar (62). Lo anterior mencionado se observa en la Figura 32.

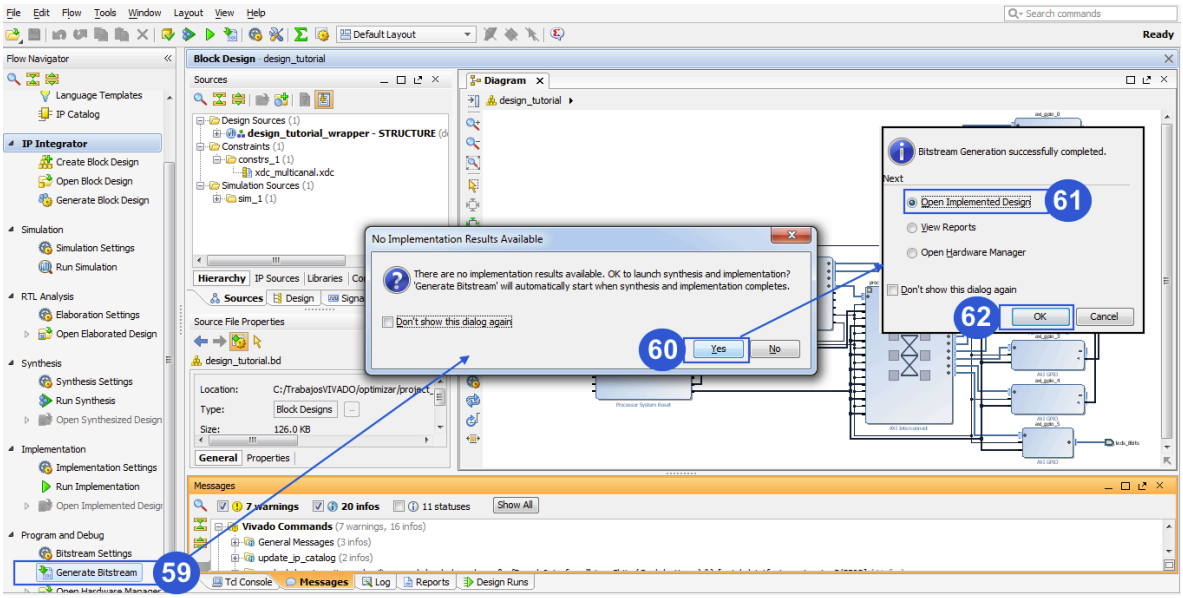


**Figura 32 Generar archivo Bitstream.**

Después como se observar en la Figura 33 se tiene que exportar el hardware, para esto en la barra de menú se selecciona la opción *file* (63), en el menú que se despliega se elige *Export* (64), se desplegara un submenú en el que se tiene que escoger la opción *Export Hardware* (65), en la ventana emergente se marca la casilla de incluir bitstream (66) y presionar el botón ok para finalizar la exportación.

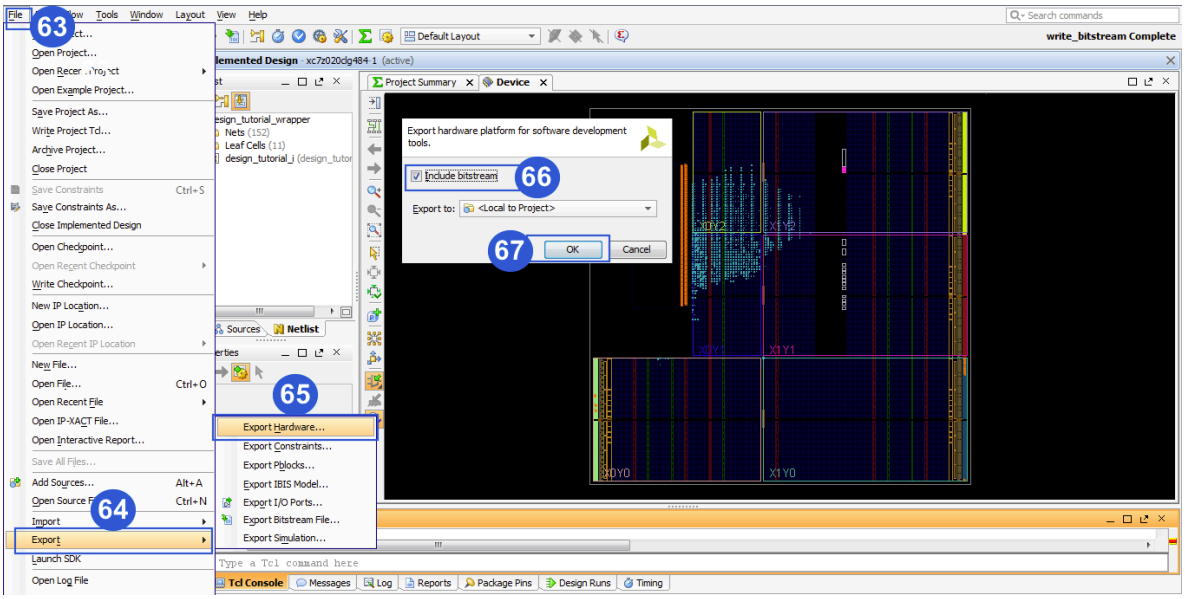


**Figura 33. Exportar el hardware.**

Y por ultimo como se visualiza en la Figura 34, en la barra de menú se elige la opción *file* (68), en el menú que se despliega se selecciona *Launch SDK* (69) y en la ventana emergente se presiona el botón *ok* (70) para terminar la sección de desarrollo de arquitectura hardware.

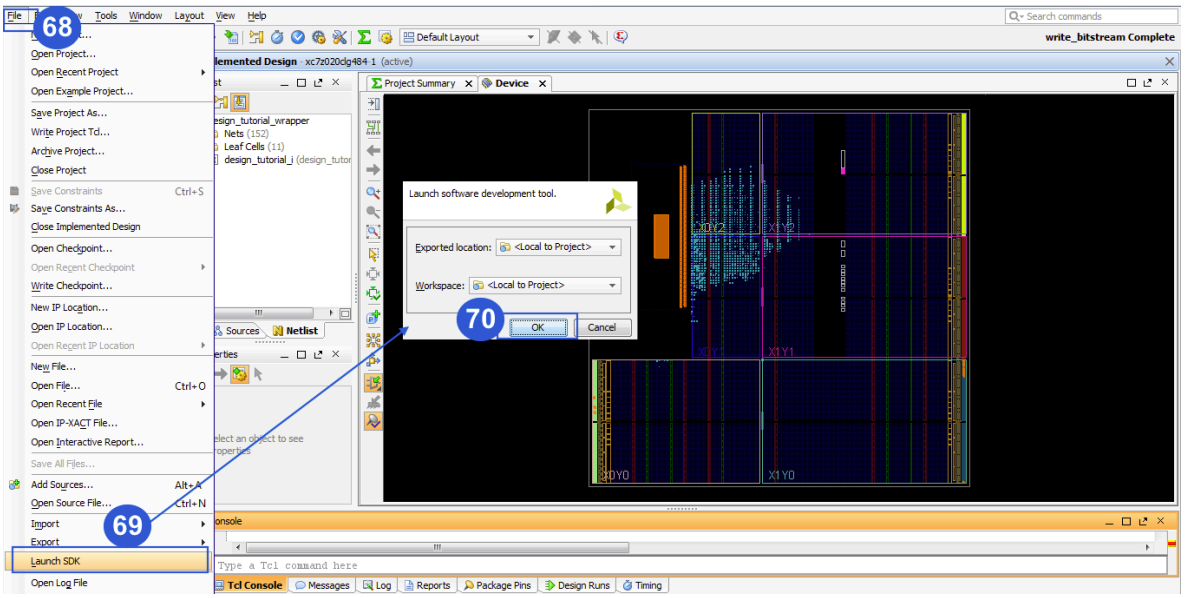


**Figura 34. Lanzamiento del SDK.**

# Programación de la aplicación de software

La sección de programación se desarrolla en la aplicación SDK (Software Development Kit), una vez abierto el SDK, en la barra de menú se elige *File* (1), en el menú desplegable se selecciona la opción *New* (2) y en el submenú se oprime *Application* Project (3), en la ventana que aparece, se puede indicar el nombre de la aplicación (4) y para terminar se presiona el botón *Finish* (5).

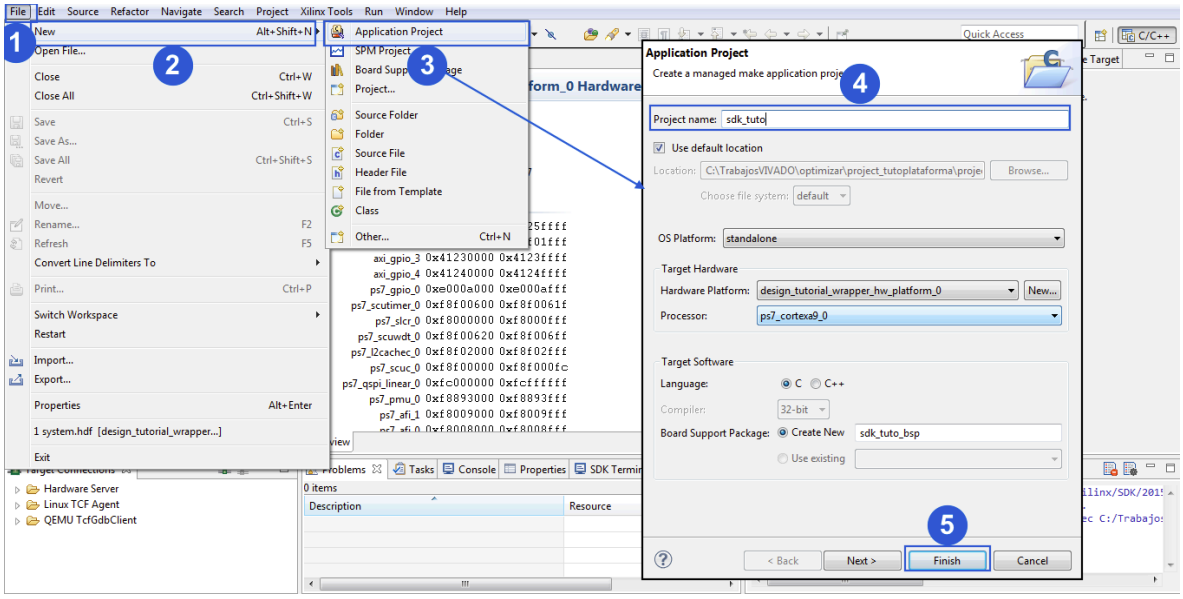


**Figura 35. Crear nueva aplicación.**

Para programar la aplicación de software se tiene que abrir el archivo *helloworld.c* (6), se puede observar y modificar el código en lenguaje *C* (7) en la Figura 36, una vez programada la aplicación se pasa a programar el FPGA presionando el botón indicado (8) y en la ventana emergente se selecciona el botón *Program* (9). En la placa de desarrollo se encenderá un led en color azul al haberse programado la FPGA con éxito.

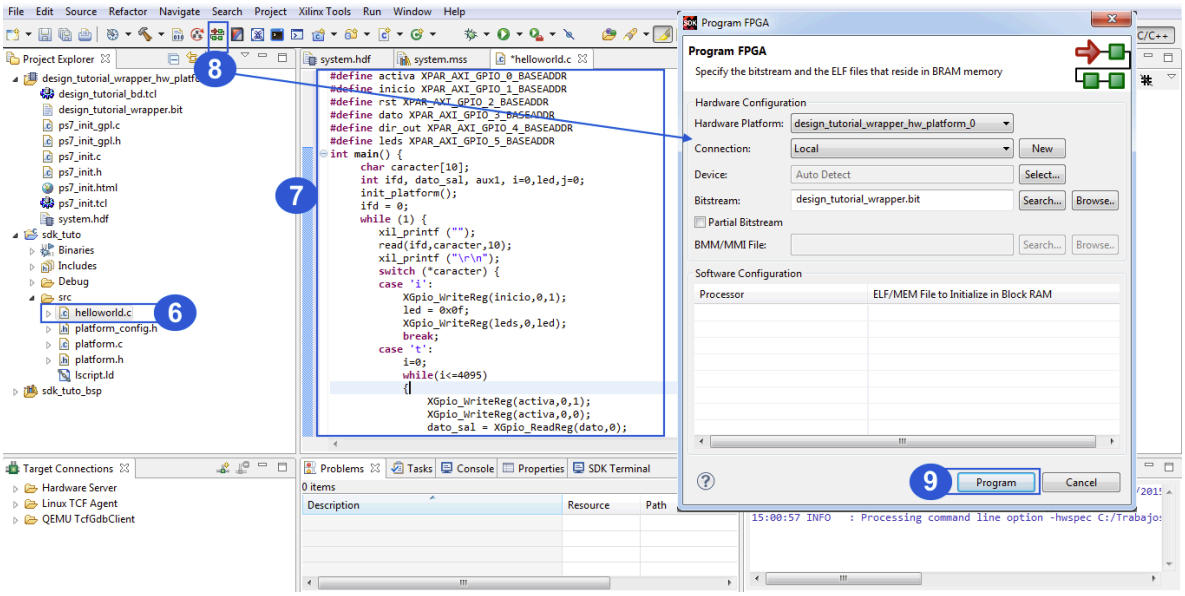


**Figura 36. Programar FPGA.**

Por último se tiene que programar la aplicación de software sobre el procesador, para esto se tiene que seleccionar y dar click derecho sobre la carpeta *sdk* (10), en el menú desplegado seleccionar *Run As* (11) y en el submenú se elige la opción *Launch on Hardware* (12). Al haber realizado lo anterior ya estará embebido el hardware y programado el software y por consiguiente el sistema funcionando.

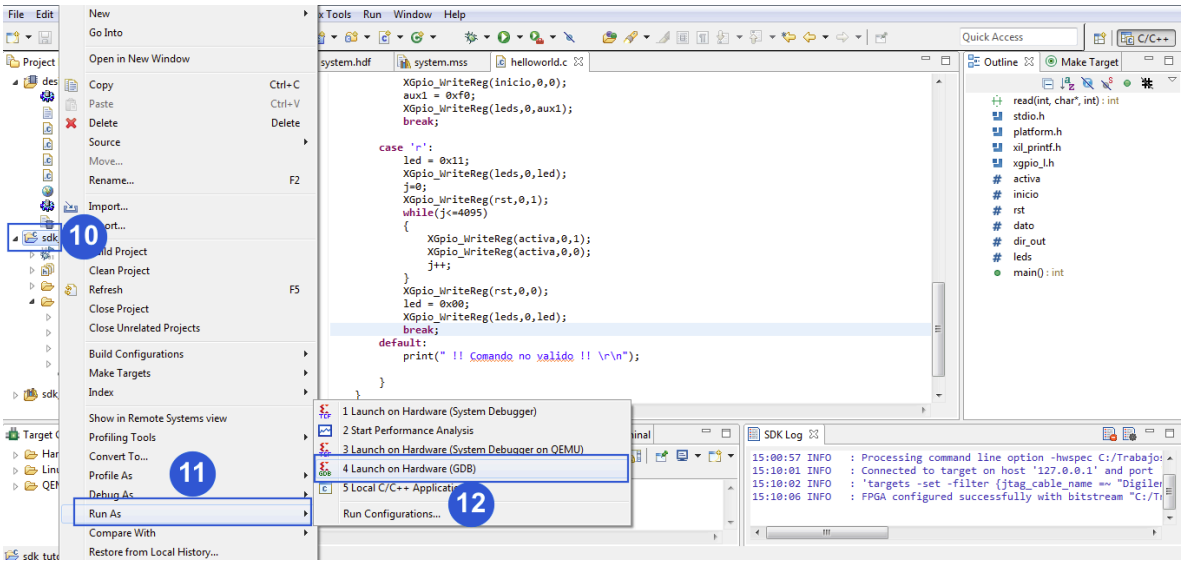


**Figura 37. Arrancar la aplicación.**