ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

Diseño e implementación de prácticas y guías de laboratorio enfocadas en el uso de tecnología FPGA para la materia de Diseño de Aplicaciones en Telecomunicaciones

PROYECTO INTEGRADOR

Previo la obtención del Título de:

Ingeniería en Telecomunicaciones

Presentado por:

Kevin Alexander Medina Sigüenza

Oscar Sebastián Flores Guerrero

GUAYAQUIL – ECUADOR

Año: 2022

DEDICATORIA

Le dedico este proyecto integrador en primer lugar a mi querida madre Macarena que nunca titubeo en todo este proceso de logros y derrotas, a mis queridos tíos Ciro y Justa que me apoyaron incondicionalmente en mi instancia de la universidad y me ayudaron a crecer personalmente, a mis hermanos que siempre estuvieron presentes en mi mente. Para mi abuelita Luz que fue mi inspiración mientras estuvo en vida, finalmente a mis amigos y a mi pareja Nicole que fueron pilares importantes en mi desarrollo profesional e interpersonal.

OSCAR FLORES GUERRERO

Le dedico este proyecto a mi querida madre Blanca que me ha acompañado todos estos años, animándome a seguir adelante a pesar de las dificultades que se presentaran. A mis hermanos que me han alentado y apoyado incondicionalmente en este proceso, a toda mi familia que de una u otra forma me hacía saber de su apoyo y buenos deseos.

KEVIN MEDINA SIGÜENZA

AGRADECIMIENTOS

Agradecemos en primer lugar a Dios por darnos la oportunidad de alcanzar esta meta tan soñada, a ESPOL por formarnos y educarnos todos estos años, a nuestras familias por alentarnos y apoyarnos incondicionalmente. Finalmente, a nuestros amigos con los que hemos compartido buenos y malos momentos.

KEVIN MEDINA SIGÜENZA

OSCAR FLORES GUERRERO

DECLARACIÓN EXPRESA

“Los derechos de titularidad y explotación, nos corresponde conforme al reglamento de propiedad intelectual de la institución; *Kevin Medina Sigüenza y Oscar Flores Guerrero* damos nuestro consentimiento para que la ESPOL realice la comunicación pública de la obra por cualquier medio con el fin de promover la consulta, difusión y uso público de la producción intelectual"

OSCAR SEBASTIAN FLORES GUERRERO

KEVIN ALEXANDER MEDINA SIGÜENZA

EVALUADORES

Firmado electrónicamente por:
WASHINGTON ADOLFO MEDINA MOREIRA

M.Sc. Washington Medina Moreira

PROFESOR DE LA MATERIA

Firmado electrónicamente por:
JORGE ANDRES BRITO COLLANTES

M.Sc. Jorge Brito Collantes

PROFESOR TUTOR

RESUMEN

En este proyecto integrador se implementan prácticas y guías de laboratorio para la materia de Diseño de Aplicaciones en Telecomunicaciones, basándose en el uso de FPGA de la marca Xilinx. Se trabaja con el software Vivado y en temáticas relacionados a las telecomunicaciones como lo son los sistemas de sensores, protocolos de comunicación y sistema de cámara. Para la primera práctica se implementa el protocolo de comunicación I2C entre un Arduino, que tendrá la función de maestro, y una FPGA que actuará como esclavo; de esta forma se ingresarán los datos mediante el monitor serial de Arduino, y serán enviados hacia la FPGA para que realice el procesamiento requerido, visualizando finalmente los resultados en el monitor antes mencionado. En la segunda práctica se trabaja con sensores conectados a la FPGA, esta realiza la conversión analógica a digital de los datos y los transmite hacia una red Zigbee conformada por módulos Xbee; finalmente los datos detectados son visualizados en una PC mediante una interfaz gráfica diseñada en LabVIEW. Para la tercera práctica se utilizó el módulo de cámara OV7670, en la FPGA se realizó en Verilog la programación necesaria para el correcto funcionamiento del módulo; así también se configuró la salida VGA de la placa para lograr visualizar mediante una pantalla las imágenes captadas. De esta manera, las prácticas implementadas les permiten a los estudiantes familiarizarse con los equipos y softwares utilizados, así como y aprender a desarrollar proyectos desde distintas perspectivas.

Palabras Claves: FPGA, Xilinx, Sistema de sensores, Módulo OV7670, Xbee, Protocolo I2C, Vivado.

ABSTRACT

In this capstone course project, practices and laboratory guides are implemented for the Application Design in Telecommunications course, based on the use of Xilinx brand FPGAs. This work has been done with the Vivado software and includes topics related to telecommunications such as sensor systems, communication protocols and camera systems. For the first practice, the I2C communication protocol is implemented between an Arduino, which will act as master, and an FPGA that will act as slave; in this way, the data will be entered through the Arduino serial monitor and will be sent to the FPGA to perform the required processing, finally displaying the results in the aforementioned monitor. In the second practice, we work with sensors connected to the FPGA, which performs the analog-to-digital conversion of the data and transmits it to a Zigbee network made up of Xbee modules; finally, the sensed data is displayed on a PC through a graphical interface designed in LabVIEW. For the third practice, the OV7670 camera module was used, in the FPGA the necessary programming for the correct operation of the module was carried out in Verilog; likewise, the VGA output of the board was configured to display the captured images on a screen. Thus, the implemented practices allow students to become familiar with the equipment and software used, as well as learning to develop projects from different perspectives.

Keywords: *FPGA, Xilinx, Sensor system, OV7670 Module, Xbee, I2C Protocol, Vivado.*

ÍNDICE GENERAL

ABREVIATURAS

PAO	Periodo Académico Ordinario
FPGA	Field Programmable Gate Array
RAM	Random Access Memory
VHDL	Very High Speed Integrated Circuit Hardware Description Language
ESPOL	Escuela Superior Politécnica del Litoral
FIEC	Facultad de Ingeniería en Electricidad y Computación
I2C	Inter Integrated Circuits
PC	Personal Computer
DIECTQAI	Departamento de Ingeniería Eléctrica, Electrónica, Control, Telemática y Química Aplicada a la Ingeniería
PABX	Private Automatic Branch Exchange
PID	Proporcional Integral Derivativo
ARM	Advanced RISC Machine
EE. UU	Estados Unidos
IMAX	Image Maximum
SPI	Serial Peripheral Interface
PAN	Personal Area Network
SRAM	Static Random Access Memory
PLD	Programmable Logic Device
AMD	Advanced Micro Devices
INTEL	INTegrated ELectronics
Tcl	Tool command languague
RTL	Register transfer level

IP	Intellectual Property
GND	Ground
UART	Universal Asynchronous Receiver-Transmitter
Tx	Transmisor
Rx	Receptor
SDA	Serial Data
SCL	Serial Clock
SCK	Serial Clock
MOSI	Master Output Slave Input
MISO	Master Input Slave Output
CS	Chip Select
bps	Bits por segundo
Mpbs	Megabits por segundo
LAN	Local Area Network
WAN	Wide Area Network
ATM	Asynchronous Transfer Mode
GHz	Giga Hertz
m	metros
mW	mili Wats
dBm	Decibelio referenciado a 1 mili vatio
AP	Access Point
IOT	Internet Of Things
KPI	Key Performance Indicators
IEEE	The Institute of Electrical and Electronics Engineers
CPU	Central Processing Unit

ROM	Read Only Memory
ALU	Read Only Memory
USB	Universal Serial Bus
VGA	Video Graphics Array
ADC	Analog Digital Converter
Pmod	Peripheral module
mV	mili voltios
SoC	System on chip
A/D	Analog/Digital
vaux	auxiliary voltage
MIO	Multiplexed input/output
AXI	Advanced Extensible Interface
XSA	Xilinx Shell Archive
ID	Identificador
hsync	Sincronización horizontal
vsync	Sincronización vertical

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

CAPÍTULO 1

1 INTRODUCCIÓN

En los periodos académicos anteriores, PAO 2020-2021, se realizaron prácticas de laboratorios para la materia de Diseño de Aplicaciones en Telecomunicaciones, en el PAO 1 del año 2022 se continúa con la misma línea. Ahora el enfoque será diferente debido a la nueva adquisición de los dispositivos electrónicos FPGAs Zedboard Zynq-7000.

Las FPGA en el ámbito académico-industria son de suma importancia, teniendo aplicaciones variadas como: procesamiento digital de señales, radio definido por software, automatización de procesos, adquisición de datos, entre otras. Una FPGA se programa a nivel de hardware y al poseer una memoria RAM que es volátil [1], permite sobrescribir el código dando la posibilidad de interactuar y solucionar los problemas que se desarrollen en el trascurso de la actividad siendo así versátil a las adversidades.

VHDL es uno de los lenguajes que se utiliza para la programación de una FPGA, mediante este se definirán las señales con la que se trabajará y cómo se las utilizará. Por medio de este lenguaje también se define qué parte de la FPGA se utilizará y cómo interactuarán entre sí. De esta manera se puede modificar el funcionamiento del dispositivo y adaptarse a las circunstancias [2].

En el área de las telecomunicaciones es usual observar que se implementen sistemas con sensores, actuadores y cámaras junto con placas que les permitan tener un control de estos, pero que usualmente no cuentan con un equipo de la suficiente capacidad para realizar algún tratamiento o procesamiento más complejo de las señales.

Debido a esto es que se busca trabajar en conjunto con placas de mayor capacidad para reliazar procesos más profundos y complejos, segmentando así las distintas tareas a realizar según las cualidades de los dispositivos. Por este motivo, en el presente proyecto se prevé implementar tres

prácticas que integran las capacidades de una FPGA junto con otras placas y módulos de adquisión de datos, logrando así mostrar cómo se pueden incorporar distintas tecnologías para la resolución de problemas.

1.1 Definición del problema

Durante los primeros meses del 2022 el Ecuador ha alcanzado un mayor control de la emergencia sanitaria provocada por el virus COVID-19. Los contagios se han mantenido estables y en menor medida con respecto al año anterior, evidenciándose esto en el porcentaje de ocupación de camas establecidas tanto para centros de salud públicos (3%) como para privados (17%); así lo informa [3].

El porcentaje de ecuatorianos vacunados con la segunda dosis (84,84%) [4], forma parte de los factores que han permitido plantear un retorno a la presencialidad de ciertos sectores que aún mantenían sus actividades de forma virtual.

Gracias a estas mejoras uno de los sectores que retomará la presencialidad es el educativo. De esta forma, el Ministerio de Educación dispuso que en mayo del 2022 las escuelas y colegios del Ecuador reanuden de forma presencial todas sus actividades [5].

Por su parte ESPOL decidió implementar un plan de retorno progresivo para el PAO 1 del 2022, estableciendo modalidades presenciales, virtuales e híbridas para utilizarse según se requieran en sus distintas actividades [6].

Dentro de las actividades que han regresado a la presencialidad, o se planifica que lo hagan, están los laboratorios. En estos los estudiantes desarrollan habilidades en el uso de distintos equipos y escenarios mediante prácticas guiadas. Esto ha llevado a que se reestructuren dichas prácticas, pensando en las temáticas a tratar y en los equipos a utilizar.

En la FIEC uno de los laboratorios que se planifica retome la presencialidad, en el PAO 2 - 2022, es el de Telecomunicaciones [7]. En este se desarrolla la parte práctica de la materia de Diseño de

Aplicaciones en Telecomunicaciones, que corresponde a la carrera de Ingeniería en Telecomunicaciones. Dicho laboratorio ha adquirido nuevo equipamiento, entre ello se encuentra la placa de desarrollo FPGA Zedboard Zynq-7000.

Bajo este contexto, con el regreso de los estudiantes y la llegada de nueva instrumentación, se generó la necesidad de contar con un nuevo plan de prácticas (actividades y guías) que abarquen las temáticas adecuadas para el laboratorio.

1.2 Justificación del problema

Para la obtención del título de Ingeniero en Telecomunicaciones se realizará el siguiente trabajo: Se brindará apoyo al Laboratorio de Diseño de Aplicaciones en Telecomunicaciones mediante el desarrollo de prácticas, intensificando el aprendizaje de los estudiantes en el uso de nuevas tecnologías.

Debido a la pandemia del COVID-19, la ESPOL cambió de modalidad presencial a virtual en los términos académicos 2020-2021. La modalidad virtual no permitió el uso de los laboratorios físicos por dos años, por lo que las prácticas solamente se centraban en el uso de simuladores. En el primer término PAO 1-2022 se estableció la modalidad híbrida en la cual existe presencia del alumnado tanto presencial como virtual [6], permitiendo retomar la presencialidad en ciertos laboratorios.

Se planifica que para el PAO 2-2022 el Laboratorio de Diseño de Aplicaciones en Telecomunicaciones retome la presencialidad, por esto se requiere un cambio completo en las prácticas a desarrollar. Se necesita un reenfoque de estas hacia el uso de equipos físicos y en la implementación de distintas tecnologías.

El Laboratorio de Diseño de Aplicaciones en Telecomunicaciones adquirió nuevos equipos, esto creó la necesidad de contar con prácticas que estén orientadas a interactuar y desarrollar habilidades en su uso. Las aplicaciones de las placas (FPGA) son variadas, por lo que se requiere que las temáticas a tratar sean seleccionadas adecuadamente, buscando obtener el mayor aprendizaje

hacia el entorno de las telecomunicaciones. Por lo que se prefiere trabajar con un enfoque más aplicativo.

La Constitución del Ecuador marca en varios de sus artículos la importancia de manejar, investigar y desarrollar tecnología en el país. El artículo 350 indica que las instituciones de educación superior deber formar académicamente con visión científica y tecnológica, así como en la innovación. El artículo 388 menciona que se destinarán los recursos necesarios para el desarrollo tecnológico y la innovación [8]. De esta forma la adquisición de los estos nuevos equipos, por parte del laboratorio, responde a las directrices planteadas en los artículos mencionados anteriormente.

1.3 Objetivos

1.3.1 Objetivo general

Diseñar e implementar tres prácticas y guías de laboratorio enfocadas en el uso de tecnología FPGA para la materia de Diseño de Aplicaciones en Telecomunicaciones.

1.3.2 Objetivos específicos

- Definir las temáticas a tratar en las prácticas, en base a la planificación del curso.
- Investigar sobre el software Vivado, el funcionamiento del sistema y el proceso para realizar las simulaciones.
- Diseñar una práctica que integre un módulo Arduino junto a una FPGA, mediante el protocolo de comunicación I2C.
- Prototipar un sistema de sensores con conexión inalámbrica, utilizando una FPGA para el procesamiento de las señales y módulos Xbee para la comunicación.
- Diseñar un sistema de cámara utilizando la FPGA y el módulo OV7670.
- Elaborar guías de laboratorio en base a los prototipos, prácticas diseñadas.

1.4 Estado del arte

En la Escuela Politécnica Superior, Universidad de Córdoba [9].Se detectó un problema respecto a las prácticas de laboratorio, por lo cual se desarrollaron practicas presenciales en FPGA con un kit XSA-50, para realizar código VHDL en casa y así complementarlo presencialmente con los ficheros en la FPGA del laboratorio.

En la tesis Herramienta FPGA para el Diseño en Bloques y Propagación VHDL en la asignatura de Sistemas Digitales 1 [10].Las actividades se realizan de manera digital mediante una PC en donde se programa por hardware VHDL los circuitos electrónicos específicamente las puertas lógicas.

En el trabajo investigativo PROTOTIPADO RÁPIDO PARA APLICACIONES SOBRE FPGAs DE XILINX. [11].Con ayuda de diferentes kits de FPGA junto a controladores periféricos, la placa se encarga de enviar la señal lógica a los instrumentos electrónicos (teclado-ratón) para trabajar, de esta manera se optimiza el trabajo.

La FPGA permite programar a nivel de hardware, a la par es reprogramable, por ende, se acopla a las adversidades y dificultades que presenten los proyectos ya que se puede modificar la información sin ningún problema. [1] debido a que su memoria es volátil.

En el DIECTQAI (Departamento de Ingeniería Eléctrica, Electrónica, Control, Telemática y Química Aplicada a la Ingeniería) [12] se han creado diferentes laboratorios con distintas modalidades. En el caso del laboratorio de FPGA se tiene la modalidad presencial en donde el estudiante utiliza los instrumentos de forma física, y la modalidad remota en la que solo se realiza el código y se sube a la plataforma.

En [13] se menciona que el gobierno argentino, en busca de mejorar su competitividad económica mediante la exportación de ciertos equipos (PABX, Locutorios, sistemas de radio enlace), decidió capacitar personal en metodologías de diseño de circuitos digitales utilizando FPGA. Esta

inversión se realizó con el fin de que dichos equipos se mantengan actualizados y que logren competir con las demás marcas del mercado.

En la Universidad Autónoma de Querétaro [14], en México, se busca un entrenamiento educacional en FPGA, tanto teórico como práctico, que les permita desarrollar habilidades en tecnologías contemporáneas. Se define que en el laboratorio se tratará temas como protocolos de comunicación, descripción de hardware, PID, entre otros. Dentro de su rúbrica el 50% corresponde al desarrollo de prácticas, así como la habilidad para participar en proyectos multidisciplinarios con los que se logre resolver problemas presentes en la industria.

En un estudio realizado por la Facultad de Ingeniería Eléctrica y Electrónica de la Universidad de Alepo [15], Siria, se destaca que la FPGA forma parte de las tecnologías más utilizadas dentro de los sistemas integrados. Debido a esto se plantea que los planes de estudios en carreras de ingeniería deben responder a tal demanda. Por ello se definen modelos de aprendizaje, así como prácticas con manuales, tutoriales y equipo seleccionado que les permita a los estudiantes desarrollar las habilidades necesarias en dicha tecnología, minimizando la presencia del docente.

En la Universidad Nacional de Singapur se han creado cursos centrados en la enseñanza y capacitación sobre el diseño y desarrollo de sistemas integrados [16]. Se menciona que las asignaturas implementadas tratan las temáticas de: Sistemas embebidos en tiempo real y los aspectos de hardware de dichos sistemas. Cada curso cuenta con un proyecto en el que se utilice una FPGA y demás periféricos, que permitan visualizar e interactuar, con los que se refuerce la teoría aprendida.

En la Universidad Autónoma de Querétaro [17] , México, se decidió desarrollar una plataforma educativa donde los estudiantes puedan diseñar sistemas embebidos, sobre FPGA, enfocados en el procesamiento de imágenes. Los objetivos que se plantearon son que el estudiante logre percibir los retos que representa el diseño de sistemas digitales, así como interactuar con las herramientas y marcos de diseño utilizados en la industria.

En la Universidad de Zhejiang [18], en China, se propuso llevar a cabo un curso de hardware integrado basado en FPGA como complemento al curso convencional de sistemas embebidos basado en ARM. De tal forma se busca adaptar la enseñanza de sistemas integrados a la demanda del mercado. Se indica que en la parte práctica se cuenta con proyectos de distintos niveles de dificultad para ser propuestos a los estudiantes según su destreza.

En la Universidad de Columbia [19], en EE. UU., se desarrolló un artículo sobre la importancia de haber implementado un curso de sistemas embebidos basado en FPGA. Se menciona que la asignatura cuenta con proyectos que abarcan distintas temáticas como video, audio, redes, entre otros; dando de esta forma libertad a los estudiantes de elegir el área con la cual trabajar. También se indica que el manejo tanto de software como hardware, así como la variedad de proyectos, generan en los estudiantes una mayor motivación para desarrollar sus actividades.

En una publicación realizada por la Escuela de Ingeniería Eléctrica e Informática del Instituto de Tecnología de Georgia [20], EE. UU., se mencionan las mejoras obtenidas, en los proyectos realizados por los estudiantes, gracias a la implementación de placas basadas en FPGA. Se señala que trabajar con placas reconfigurables permite aumentar la complejidad de los proyectos, reutilizarlas en otros diseños y minimizar el requerimiento de equipos adicionales.

En la Universidad de Agricultura y Mecánica de Texas [21], EE. UU., se desarrolla una metodología que permite aumentar la velocidad de enseñanza en los cursos de diseño digital, entre ellos los que utilizan FPGA. Se menciona que es de importancia que los estudiantes logren comprender y aplicar los conocimientos adquiridos con prontitud, permitiendo así profundizar en mayor medida sobre la temática. Con esta metodología también se pretende obtener una migración más fluida hacia las posteriores asignaturas.

En la Universidad Técnica de Lisboa, en Portugal, se plantea un proyecto de laboratorio basado en FPGA [22]. Se prevé abarcar temáticas centradas en sistemas embebidos, contar con una programación que permita reutilizar código, optimizar el procesamiento y desarrollar periféricos que

permitan conectar el sistema con sensores y actuadores. También se menciona brindarán a los estudiantes herramientas como tutoriales y guías que les permitan profundizar en la temática, dejando la intervención del docente en dudas puntuales

Mediante los árboles de decisión aplicados en FPGA se plantea reconocer un conjunto de letras y números creando la función de clasificación y regresión, mediante el software Vivado en donde se programará por VHDL [23]. Logrando crear coprocesadores para proyectos a futuro como reconocimiento de imagen.

En la Universidad de Málaga [24]. Se ha implementado una propuesta de laboratorios en línea de FPGA mediante un sitio web donde los estudiantes pueden realizar las actividades remotamente ya que el código se carga directamente a la placa que se encuentra en el laboratorio, a la par trabajan junto a microcontroladores periféricos generando una experiencia de trabajo real.

En la Universidad de la Laguna se ha implementado el Prototipo de cámara IMAX+ como recurso docente para calibración y tratamiento de imágenes en FPGA, el proyecto se basa en el trabajo mutuo entre la placa y el sensor GSENSE400 [25], generando una conexión en el proceso de traslado de información el sensor presenta una interfaz SPI lo cual produce una conexión dúplex entre varios microcontroladores o periféricos [26], ya que se debe tener en cuenta que es un sensor visual por ende se trabaja con valores a tiempo real dependiendo del movimiento del objeto a estudiar.

En la Universidad Jaume I el uso de los módulos Arduino y Raspberry-PI, [27] se ha convertido en el uso diario de los estudiantes por su fácil manejo y aprendizaje y sobre todo por lo económico, en las prácticas de laboratorio de automatización de sistemas se ha incluido el uso de FPGA, para poder realizar prácticas digitales mediante la programación del hardware VHDL para poder crear circuitos digitales que se acoplen a otras tecnologías.

Tal como se ha comentado, las instituciones educativas a nivel mundial han reconocido la importancia de que sus estudiantes manejen y diseñen utilizando FPGA. Se han aplicado laboratorios

en distintas carreras como computación, automatización, electricidad, telecomunicación, con sus respectivos proyectos.

Observando las temáticas tratadas en cada laboratorio mencionado, este proyecto de titulación se decidió diferenciar optando por desarrollar prácticas orientadas al trabajo con sensores, su comunicación y la integración con otras placas de desarrollo. De esta forma se diseñarán tres prácticas utilizando la tecnología FPGA como base.

Para la programación de la placa se trabajará con el software Vivado, el cual ha sido poco utilizado en los laboratorios citados. Será la primera vez que se utilice este software en el laboratorio de Telecomunicaciones, por lo que esto también representa un punto diferenciador del proyecto.

1.5 Alcance

El trabajo se realizará entre los meses de mayo a septiembre correspondientes al primer PAO del término 2022, previo a la obtención del título de Ingeniero en Telecomunicaciones. El tipo de trabajo a desarrollar es de carácter académico y será implementado en el Laboratorio ide Diseño de Aplicaciones en Telecomunicaciones de la FIEC-ESPOL.

Se utilizará conocimiento en base a la materia de Sistemas Digitales en el funcionamiento de las puertas lógicas, a la par de Sistema de Comunicaciones I Y II, debido a los procesos de comunicación y traslado de información; para el diseño de circuitos la materia de Principios de Electrónica. Por otro lado, el uso de los conocimientos del curso de Diseño de Aplicaciones en Telecomunicaciones es crucial para todos los aspectos antes mencionados y la conectividad de los módulos Xbee.

Las prácticas de laboratorio que se van a realizar trabajan paralelamente con el contenido de la materia teórica, por lo que se busca reforzar el conocimiento impartido en clases, a causa de estos factores se desarrollarán 3 actividades.

La primera actividad es la práctica de laboratorio en la que se integra un módulo Arduino junto a una FPGA por medio del protocolo de comunicación I2C. Esta se desarrollará en el mes de junio de

2022, la cual consiste en utilizar la FPGA Zedboard Xilinx Zynq-7000 junto a un módulo Arduino. Se conectará la FPGA en modo esclavo y Arduino en modo maestro mediante el protocolo I2C, para luego desarrollar expresiones aritméticas.

La segunda actividad consiste en crear un sistema de FPGA en conjunto con módulos Zigbee en la implementación de una red PAN, para el control de sensores utilizados en domótica [28]. Esta práctica se desarrollará en el mes de julio de 2022. Consiste en una red de sensores donde las señales serán procesadas por la FPGA y recibidas por el módulo Xbee, luego la información se enviará hacia el otro módulo Xbee realizando una conexión dúplex. Finalmente, los datos censados se visualizarán en una computadora.

La tercera actividad es el sistema de cámara FPGA utilizando el módulo OV7670 [29]. Se desarrollará en el mes de julio de 2022, en donde se trabajará la FPGA directamente con el módulo visual para poder transmitir en tiempo real la imagen o video requerido. Se programará el módulo de acuerdo con las características de la placa mediante un protocolo I2C.

1.6 Metodología

En primer lugar, se realizó una investigación con el fin de conocer cómo otras universidades han implementado laboratorios basados en FPGA. Se identificaron las áreas profesionales cubiertas en los distintos cursos, así como las temáticas tratadas en cada una de las actividades realizadas.

A continuación, se revisó la planificación de la materia de Diseño de Aplicaciones en Telecomunicaciones para en base a esta definir las temáticas a tratar en las prácticas a diseñar. Llegado este punto se estableció el eje principal a seguir de cada una de las prácticas.

Luego se procedió a investigar sobre el uso del software Vivado, ya que por primera vez sería utilizado en el laboratorio. Se estudió el funcionamiento del sistema y los pasos a seguir para poder realizar una simulación, así como una posterior ejecución en conjunto con la FPGA.

Posteriormente se inició con el diseño de la primera práctica. Se realizó la programación necesaria, en Vivado, para establecer la comunicación I2C entre la FPGA y el Arduino. Se corroboró la transmisión de información entre las placas enviando operaciones sencillas hacia la FPGA.

Seguidamente se diseñó la segunda práctica. Se inició con el prototipado electrónico de los sensores. Luego se procedió a realizar la programación en Vivado de los bloques que serán utilizados por las señales provenientes de los sensores. Finalmente se configuraron los módulos de comunicación inalámbrica, Xbee, para proporcionar la transmisión y recepción de las señales tratadas por la FPGA.

Para el diseño de la tercera práctica se procedió a establecer las conexiones entre la cámara (OV7670) y la FPGA. Así también, se realizó la programación en Vivado para establecer la transmisión de la imagen y permitir la visualización en pantalla.

Finalmente se elaboraron las guías correspondientes a cada práctica desarrollada, en base a los formatos manejados por el laboratorio se definieron los objetivos, marcos, actividades y demás puntos a tratar; completando así las actividades planteadas para el proyecto.

CAPÍTULO 2

2 MARCO TEÓRICO

En este capítulo se expondrán las definiciones teóricas las cuales son base para el presente trabajo de titulación. Se iniciará hablando de las FPGA, su funcionamiento básico y características; a continuación, se mencionarán las marcas y software disponibles. Se definirán los protocolos de comunicación que permitirán comunicar a los dispositivos en las prácticas. Se describirán los tipos de redes a implementar, dispositivos y módulos empleados.

2.1 FPGA

Una FPGA (Field Programmable Get Array), como su nombre lo indica es un arreglo programable de compuertas lógicas. Se caracteriza por realizar operaciones matriciales utilizando la lógica matemática programable mediante hardware. Su funcionamiento se da de forma paralela por lo que cada tarea a realizar se asigna a una parte del chip, permitiendo realizar varias tareas de forma simultánea [30].

Una FPGA comúnmente está compuesta por un arreglo matricial de elementos lógicos, entradas, salidas y un esquema de interconexiones entre los distintos elementos. De esta forma se logra programar tanto los componentes lógicos como las interconexiones entre ellos, además posee una memoria RAM (Random Access Memory) lo cual permite sobrescribir la información [31].

Para realizar la configuración de estos dispositivos en primer lugar se debe diseñar el sistema digital utilizando un lenguaje de descripción de hardware, luego se procede a realizar la simulación del sistema. Corroborado el funcionamiento, se continua a programar la FPGA mediante el software utilizado para la descripción y simulación [31]. En la *Ilustración 2.1* se observa la estructura básica de una FPGA donde se tienen bloques lógicos e interconexiones programables, bloques de memoria RAM, señales de reloj y entradas-salidas en la periferia.

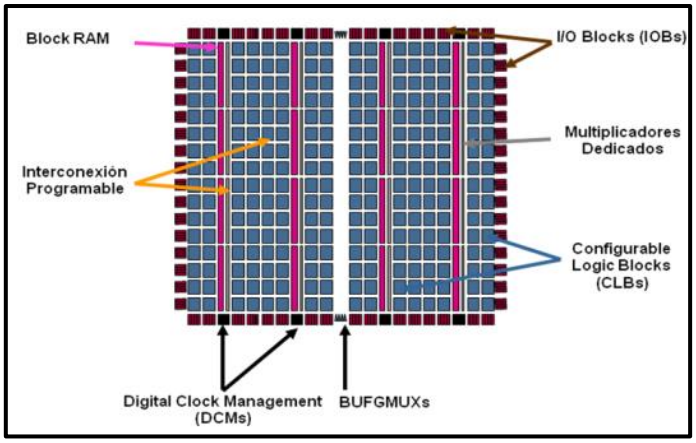

Ilustración 2.1. Estructura de una FPGA [31]

El elemento base de una FPGA es la celda de configuración, esta determina bajo qué forma trabajará cada elemento lógico. Se tienen 3 tipos de celdas: SRAM, anti-fuse y Flash. Las celdas tipo SRAM pierden la información al perder la alimentación, permite disponer de mayor cantidad de RAM, pero se requieren dispositivos extras para realizar el booteo [32].

Las celdas tipo Flash mantienen en todo momento la información aun habiendo quitado la alimentación, además de que no se requieren otros dispositivos para realizar su configuración. Las celdas tipo anti-fuse son las que presentan la mayor confiabilidad y velocidad, pero pueden ser programadas una sola vez, esto se debe a que al ser programadas una pequeña cantidad de aislante térmico se derrite permitiendo o negando la conexión entre los elementos lógicos [32].

Actualmente existen varios fabricantes de FPGA siendo los principales Xilinx y Altera, los cuales manejan su propio software para la programación de las FPGA.

2.1.1 XILINX

En el año 1985 salió al mercado la primera FPGA de la mano de XILINX, siendo la vanguardia de los PLD [31]. Hoy en día XILINX pertenece a la compañía AMD, domina el mercado de ventas de este dispositivo electrónico el cual se programa mediante hardware por medio de Verilog o VHDL utilizando el software gratuito Vivado [33]. Presenta una cartera de productos variada (Serie Spartan, Aritx, Kintex, Virtenx) según la aplicación a realizar y el procesamiento requerido.

2.1.2 ALTERA

Es una marca de FPGA perteneciente a Intel, es un modelo de circuito el cual se puede configurar mediante hardware utilizando Verilog o VHDL. Junto a XILINX son las compañías que sobresalen y dominan el mercado de FPGA. La programación se realiza mediante el software gratuito Quartus [34]. Altera ofrece series como Cyclone, MAX, Arria, Stratix para ser aplicadas según el procesamiento, consumo y costo que se requiera en la aplicación.

2.2 Lenguajes de descripción de hardware

Para realizar la programación de una FPGA es decir indicar cómo se utilizarán las entradas, salidas disponibles y su interconexión, se cuenta con dos opciones en cuanto a lenguajes para la descripción del hardware los cuales son VHDL y Verilog.

2.2.1 VHDL

VHDL (Very High Speed Intragrated Circuits Hardware Description Languague) es un lenguaje de descripción de Hardware que trabaja en base a la matemática lógica y compuertas, mediante este concepto logra describir circuitos electrónicos configurando los bloques lógicos y las conexiones entre sí [35].

Para diseñar el sistema se deben definir las entradas, salidas, así como su interconexión. En VHDL la definición de entradas y salidas se realiza mediante la entidad y la descripción de su interconexión mediante la arquitectura, estas dos son las secciones claves dentro del código de VHDL. La entidad se encarga de describir el exterior del circuito, es decir las entradas y salidas con las que se cuenta. Se definen su número, nombre y tipo. Los valores de estos puertos no son modificables en la programación interna del circuito. En esta sección de la programación también son definidos valores genéricos o constantes [36], en la Ilustración 2.2 se observa la estructura básica de la entidad.

```
entity nombre is
     generic (cte1: tipo := valor1; cte2: tipo:= valor 2; …);
     port (entrada1, entrada2, … : in tipo;
           salida1, salida2, …: out tipo;
           puertoi : modo tipo);
end nombre;
```

Ilustración 2.2. Estructura básica de la entidad [36].

La arquitectura se encarga de describir cómo se interconectarán las entradas y salidas definidas en la entidad, es decir se describe el funcionamiento interno del circuito. En la Ilustración 2.3 se observa su estructura básica. Es esta parte es donde se colocarán las sentencias, el código propio de VHDL. La primera sentencia por colocar es la que indica el nombre de la entidad con la que se relaciona la arquitectura. También se definen señales auxiliares que pueden ser utilizadas dentro de la arquitectura, pero sin afectar a la entidad [35].

```
architecture arch_name of entity_name is
-- declaraciones de la arquitectura:
-- tipos
-- señales
-- componentes

begin
-- código de descripción
-- instrucciones concurrentes
-- ecuaciones booleanes
-- componentes
     process (lista de sensibilidad)
     begin
     -- código de descripción
     end process;
end arch_name;
```

Ilustración 2.3. Estructura básica de la arquitectura [35].

2.2.2 Verilog

Es un lenguaje de descripción de hardware que presenta similitudes al lenguaje C. De igual forma que en VHDL se deben definir las entradas y salidas, así como su interconexión. A diferencia de VHDL ambos pasos se realizan en un solo bloque o módulo, tal como se observa en la Ilustración 2.4. Dentro de dicho módulo se definen el número, nombre y tipo de salidas y entradas. A continuación, se especifican las señales auxiliares que se utilizarán dentro del circuito, y finalmente se describe las interconexiones de las entradas, salidas y demás señales [37]. Debido a esto se suele mencionar que la sintaxis de Verilog es más sencilla que la de VHDL.

```
module mi_circuito (
   input x, y,
   input z,
   output f1, f2
   );
   wire cable_interno;
   reg  variable_a;
   ...

endmodule
```

Ilustración 2.4. Estructura básica del código en Verilog [37].

2.3 Software de diseño

Dependiendo de la marca de la FPGA que se vaya a utilizar, Xilinx o Altera, se utilizará el software Vivado o Quartus según corresponda:

2.3.1 Quartus

Es el software libre de ALTERA-INTEL que permite programar FPGA ya sea mediante VHDL o Verilog. Incluye herramientas que permiten implementar jerarquías en los proyectos, así como métodos de síntesis, compilación y simulación. La estructura de la programación va desde el diseño (código), síntesis, asignación de pines y finalmente el cargado a la placa. En la Ilustración 2.5 se tiene la interfaz de Quartus.

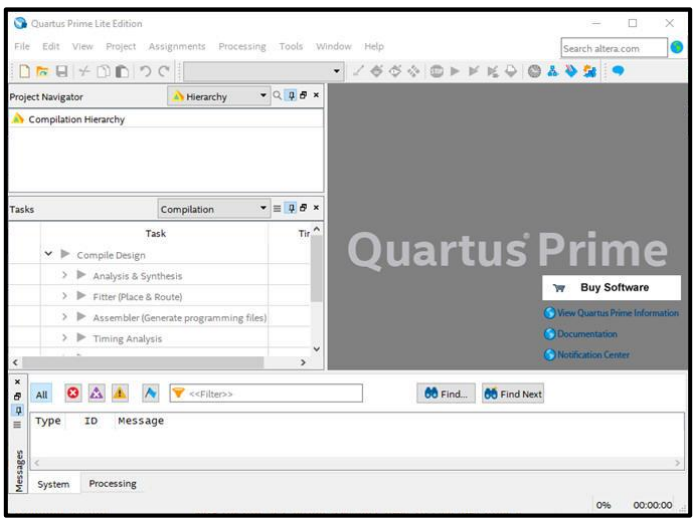

Ilustración 2.5. Interfaz principal de Quartus.

2.3.2 Vivado

Es el software libre de XILINX-AMD el cual permite programar por medio de hardware la FPGA sea por lenguaje de descripción VHDL o Verilog. Provee una interfaz gráfica para el usuario y sus herramientas se encuentran escritas en el formato nativo Tcl (Tool command languague). Para mejorar el desempeño se pueden utilizar algoritmos como RTL (Register transfer level), IP (Intellectual property) para la integración de núcleo, simulación, síntesis, entre otros [38].

En la Ilustración 2.6 se observa la interfaz principal donde se tienen los distintos componentes a ser utilizados. Estos componentes son: barra de menú (1), barra de herramientas (2), navegador de flujo (3), ventana de archivos generados (4), buscador de comandos (5), área de trabajo (6), barra de estado del proyecto (7), selector de diseño (8), barra de estado (9) y la ventana de resultados (10) [38].

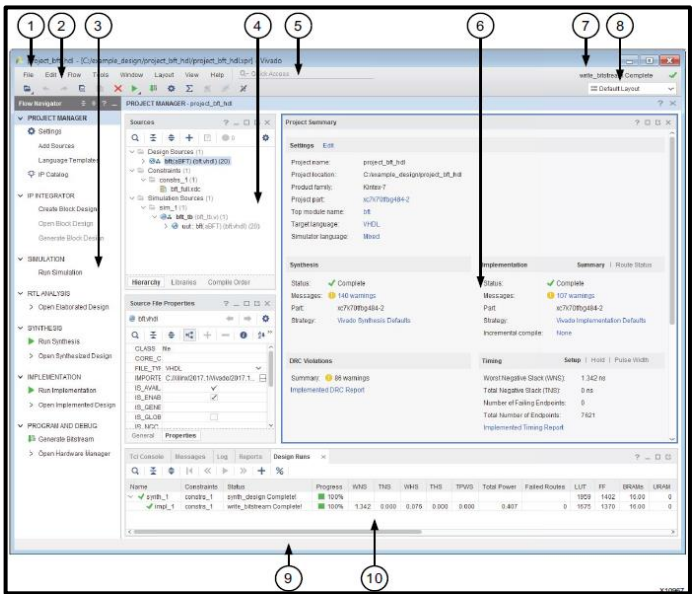

Ilustración 2.6. Interfaz de Vivado [38].

2.4 Protocolos de comunicación

Un protocolo de comunicación es un grupo de reglas o normas que se siguen dentro de una red [38]. El protocolo define políticas, requerimientos, procedimientos, límites y formatos que determinan la forma en la que dos o más dispositivos intercambiarán información entre sí dentro de la misma red. Principalmente los protocolos se dividen en dos grupos: comunicación serie y comunicación en paralelo.

2.4.1 Comunicación en paralelo

El protocolo de comunicación en paralelo permite establecer una comunicación transmitiendo simultáneamente cierta cantidad de bits, tal como se observa en la Ilustración 2.7. Para realizar el envío se necesitan tantos canales (cables) como bits se vayan a transmitir, así para los bits de datos como para los de control [39]. Este tipo de comunicación tiene un alcance de uno pocos metros, ya que a grandes distancias los distintos canales comienzan a presentar interferencias (diafonía) entre sí. Por otro lado, se alcanzan altas velocidades ya que todos los bits se transmitían a la vez. Este tipo de protocolo es poco usado debido a las limitantes que presenta.

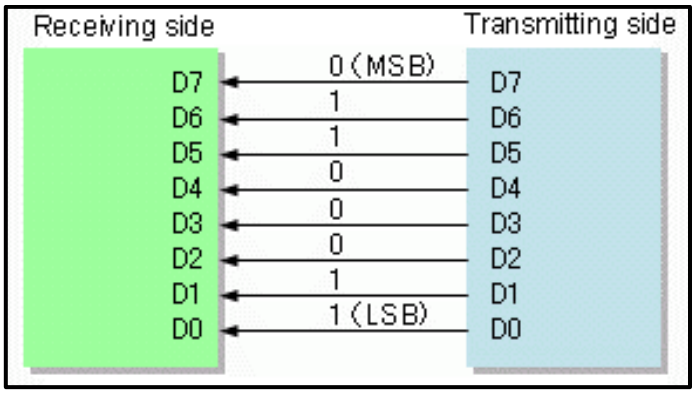

Ilustración 2.7. Ejemplo de transmisión en paralelo [40].

2.4.2 Comunicación en serie

Los protocolos de comunicación serie o serial permiten establecer una comunicación enviando un bit a la vez en secuencia [41], tal como se observa en la Ilustración 2.8. En principio esto quiere decir que se requiere de un solo canal (cable) para transmitir la información. En comparación con una comunicación en paralelo una transmisión serial presenta un mayor alcance, pero con una menor velocidad.

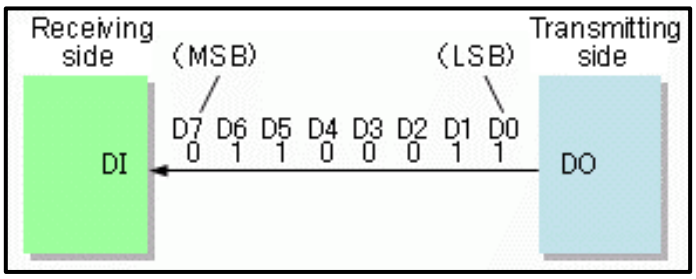

Ilustración 2.8. Ejemplo de transmisión serial [40].

De forma general en una comunicación serial los protocolos utilizarán canales extra para definir referencias, controles o corrección de errores. Dentro de este tipo de comunicación tienen dos grandes grupos: síncrona y asíncrona.

2.4.2.1 Comunicación sincrónica

Como su nombre lo indica, es una comunicación en la que no se cuenta con una señal de sincronismo entre los dispositivos conectados. La información se envía en forma de tramas, a los datos se añaden bits de inicio y parada para indicar cuando comenzar la lectura. Las tramas no siempre tendrán la misma cantidad de bits de información. Entre los dispositivos conectados se debe acordar una velocidad para el envío y lectura de los bits. Cada dispositivo manejará su reloj según el tiempo acordado [42].

Generalmente se cuenta con 3 conexiones entre los dispositivos: una para transmitir, una para recibir y una referencia [42] tal como se observa en la Ilustración 2.9. El dispositivo 1 cuenta con una línea dedicada a la transmisión y una para la recepción. De igual forma el dispositivo 2 cuenta con una línea de transmisión y recepción independientes. La tercera línea GND (Ground) les permite a ambos dispositivos tener la misma referencia para medir el voltaje.

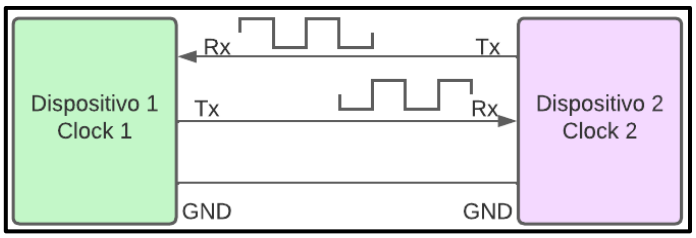

Ilustración 2.9. Ejemplo de comunicación serial asíncrona.

2.4.2.1.1 Protocolo UART

UART (Universal Asynchronous Receiver-Transmitter) es uno de los primeros protocolo serie que trabaja con solo dos líneas o cables, además de una conexión a tierra. Actualmente es poco utilizado, aunque es ideal para aplicaciones simples y de bajo costo [44]. Las conexiones son las descritas anteriormente en la comunicación asíncrona.

Al no contar con una señal de sincronismo los dispositivos deber acordar ciertos parámetros, uno de ellos es la velocidad a la que se enviarán los bits para que de esta forma sepan cuando leer los datos. El formato de una trama se observa en la Ilustración 2.10, a los bits de datos se agregan bits de inicio y paradas, así como bits opcionales de paridad para corregir errores [44].

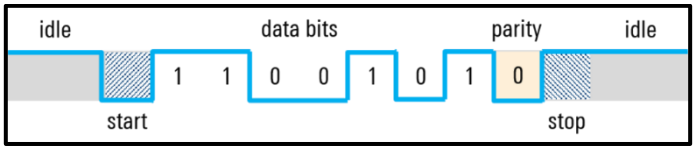

Ilustración 2.10. Formato de la trama UART [44].

Inicialmente las líneas de datos se mantienen en alto. Para iniciar una transmisión uno de los dispositivos debe registrar un cambio de alto a bajo en la línea, en ese momento comenzará a leer los bits según la velocidad acordada. Los datos se componen de 5 a 9 bits. Luego de los datos se encuentran los bits de paridad, estos le permitirán conocer al receptor si ha ocurrido algún error ya que su paridad debe coincidir con los datos. Finalmente, para indicar que la transmisión a finalizado se debe registrar un cambio de bajo a alto, este puede tener una duración de 1 o 2 bits [45].

2.4.2.2 Comunicación síncrona

En una comunicación síncrona se cuenta con una señal de sincronismo entre los dispositivos. La señal de reloj compartida indica a los dispositivos cuando leer los datos. La información es enviada en forma de frames y dentro de este el número de bits siempre será el mismo. A diferencia de la comunicación asíncrona ahora no se requieren bits de inicio y parada, además gracias a la señal sincronismo la velocidad de transmisión aumenta [43].

Entre los dispositivos se tienen 3 conexiones: una para los datos, una para la señal de sincronismo y una referencia [43] tal como se observa en la Ilustración 2.11. La línea SDA (Serial Data) será exclusiva para la transmisión de la información. La línea SCL (Serial Clock) será exclusiva para la señal de reloj, generalmente solo uno de los dispositivos se encarga de proveer esta señal a los demás. Finalmente, los dispositivos comparten una referencia GND para medir los niveles de voltaje.

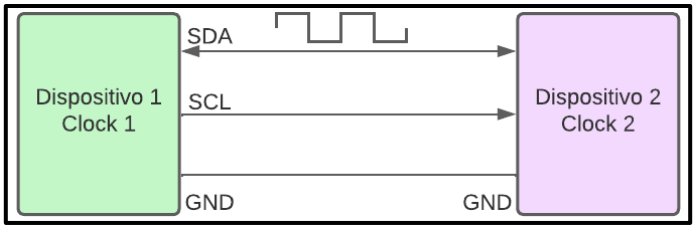

Ilustración 2.11. Ejemplo de comunicación serial síncrona.

El modo de comunicación que se observa en la Ilustración 2.11es half-duplex, aunque hay protocolos serial síncronos que implementan más líneas de datos para obtener una comunicación full-duplex. Dentro de los protocolos de comunicación serial síncrona están I2C y SPI.

2.4.2.2.1 I2C

El protocolo I2C (acrónimo para Inter Integrated Circuit) se compone de dos líneas, SDA y SCL, así como de una referencia. A los dispositivos conectados se los define como maestro o esclavo, permitiendo tener redes de: un maestro y un esclavo, un maestro y varios esclavos, o varios maestros y varios esclavos [44] tal como se observa en la Ilustración 2.12.

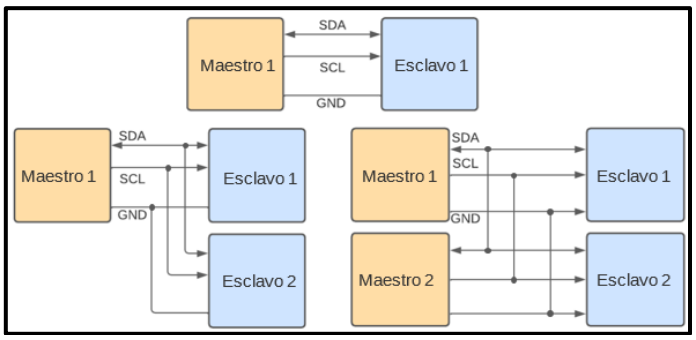

Ilustración 2.12. Configuraciones de maestro-esclavo.

Dependiendo de la aplicación necesitada la configuración de maestro-esclavo puede permitir que el maestro trabaje como emisor y el esclavo como receptor, o para que ambos sean emisor y receptor en diferentes tiempos. Al trabajar con más de un maestro se deberán configurar mecanismos para cada uno pueda transmitir en distintos instantes. En una red multimaestro, cuando sea tiempo de transmitir de un maestro puede considerar a los demás maestros como esclavos [44].

Eléctricamente las interfaces SDA y SCL son una señal a colector o drenador abierto, de tal forma que para un 0 lógico provean 0 voltios y para un 1 lógico provean alta impedancia [44]. Para obtener el voltaje correspondiente al 1 lógico se utilizan resistencias Pull-Up en SDA y SCL, de esta

forma al proveer una alta impedancia la fuente al otro extremo de las resistencias proveerá el voltaje correspondiente al 1 lógico. En la Ilustración 2.13 se observa el circuito a detalle:

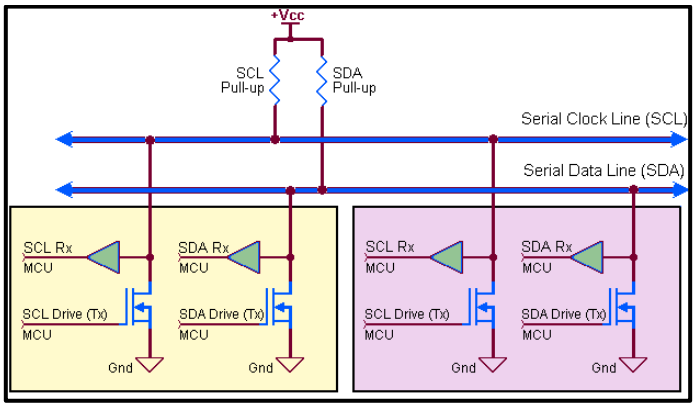

Ilustración 2.13. Hardware de las interfaces SDA y SCL [45].

En la Ilustración 2.14 se observa el proceso de comunicación y los distintos valores que tomarán las señales SDA y SCL según el estado de la transmisión. Se evidencian las condiciones de inicio y fin, así como el instante de captura de datos. Se observan bits de control para escritura o lectura, así como de reconocimiento [46].

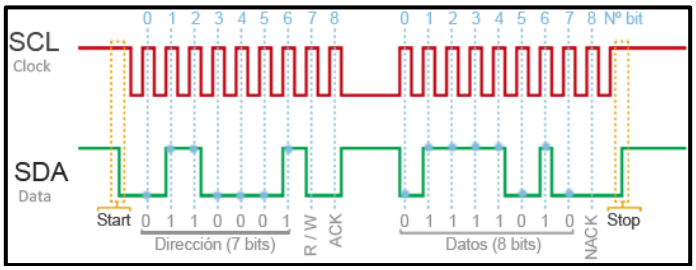

Ilustración 2.14. Estados y bits generados durante una transmisión I2C [47].

2.4.2.2.2 SPI

SPI (Serial Peripheral Interface) es una interfaz desarrollada en 1985 por la compañía Motorola, con la finalidad de permitir la comunicación a corta distancia. Los dispositivos se definen como maestros o esclavos. Se manejan cuatro líneas tal como se observa en la Ilustración 2.15. Una de ellas provee la señal SCK (Serial Clock) de sincronismo, la señal MOSI (Master Output Slave Input) se encarga de la transmisión de maestro hacia esclavo, MISO (Master Input Slave Output) maneja la transmisión de esclavo hacia maestro. La cuarta línea CS (Chip Select) está orientada a la selección del chip, es decir que al trabajar con más de un esclavo esta indica con quien se desea comunicar el maestro [51].

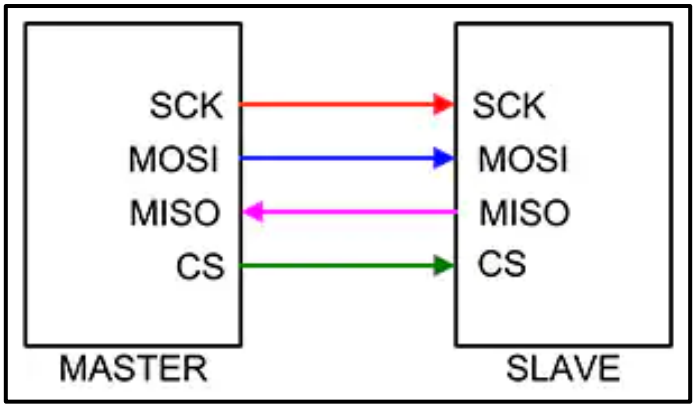

Ilustración 2.15. Líneas utilizadas en SPI [51].

Al trabajar con múltiples esclavos se puede optar por dos tipos de conexiones: directa o encadenada [51]. Estas hacen referencia a cómo se va a conectar la línea CS con los distintos esclavos, si será una dedicada a cada esclavo o una sola señal CS para todos. Estas conexiones se observan en la Ilustración 2.16.

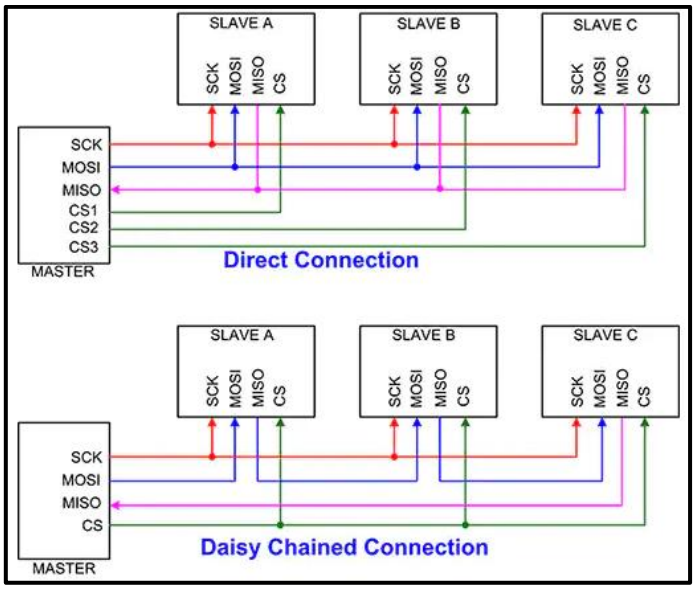

Ilustración 2.16. Tipos de conexiones en redes multi esclavo [51].

2.5 Redes de comunicaciones

Cuando se conectan dispositivos entre sí, muchas veces no es factible definir una conexión directa entre ellos. Mas bien es mejor conectarlos a una red comunicación. Estas redes se pueden clasificar según el alcance que posean, entre ellas se tienen:

2.5.1 PAN

Una Personal Area Network generalmente se compone por un único dispositivo y con un alcance aproximado de 10 metros, permitiendo conectar dos o más dispositivos móviles o estacionarios. Generalmente son redes que permiten conectar celulares, computadoras, etc. Las tasas de transmisión se suelen dar entre los 10bps – 10Mbps. Mayormente las implementaciones se realizan para definir enlaces de dispositivos Bluetooth, Zigbee, infrarrojo entre otros; debido a esto una PAN suele ser de poca complejidad y bajo costo [59].

2.5.2 LAN

Las Local Area Network se conforman por un conjunto de dispositivos interconectados entre sí, generando el medio necesario para el intercambio de información entre ellos. El alcance de estas redes es pequeño, encontrándose en edificios, casas o un pequeño conjunto de estos.

Generalmente los dueños de los dispositivos que se encuentran en red son los propietarios de la red LAN, por lo que ellos son los que estarán a cargo de su mantenimiento y monitoreo. Estos propietarios también serán los encargados de elegir los dispositivos que compondrán la LAN. La velocidad de los datos obtenida en las redes LAN es mucho mayor a la que se obtiene en una red WAN. Para su configuración se pueden aplicar tecnología conmutadas (ethernet, fibra) o inalámbricas [60].

2.5.3 WAN

Las Wide Area Network son las redes más amplias, abarcando grandes áreas geográficas. Para establecer la comunicación utiliza rutas de acceso público y otras provistas por proveedores de telecomunicaciones. La estructura de esta red está compuesta por conmutadores o nodos interconectados, los cuales generaran los caminos necesarios para que la información circule entre los distintos nodos y alcance su destino.

Típicamente estas redes utilizaban tecnología de conmutación por circuitos o por paquetes, aunque actualmente se cuenta con nuevas técnicas como lo son la transmisión por tramas o frame relay, así como las redes ATM [60].

2.6 Tecnologías inalámbricas

A continuación, se mencionan algunas de las tecnologías de comunicación inalámbricas disponibles en el mercado.

2.6.1 Bluetooth

Es una tecnología de comunicación inalámbrica ya que no necesita de una conexión serial o paralela para transmitir datos como se observa en la Ilustración 2.17, el cambio de información entre transmisor y receptor se produce trabajando en frecuencia de 2.4 GHz. En un área de poco alcance vinculándose bajo el mismo perfil, ya que si un dispositivo es compatible con otros perfiles no se puede vincular. [48]. Bluetooth de clase 2 es la vanguardia de esta tecnología permitiendo incrementar la distancia en línea de vista (20 m) entre los dispositivos todo esto dependerá de los obstáculos en el entorno a trabajar. [49] En la Tabla 2.1 se tienen las características generales de las distintas versiones de bluetooth.

	ALCANCE	CONSUMO MEDIO	POTENCIA	DBM
Bluetooth Clase 1	100 metros	100 mW	Alta	20
Bluetooth Clase 2	20 metros	2,5 mW	Medio alta	4
Bluetooth Clase 3	1 metro	1 mW	Media	0
Bluetooth Clase 4	0,5 metros	0,5 mW	Baja	-3

Tabla 2.1. Características de las versiones de Bluetooth.

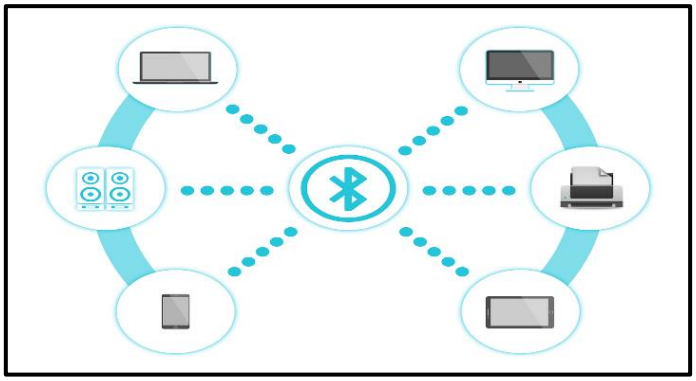

Ilustración 2.17 Tecnología Bluetooth [50]

2.6.2 Wifi

Es una tecnología móvil que permite la comunicación de diferentes dispositivos (Laptops-Celulares-Tabletas-etc), bajo una misma red compartiendo transmisión y recepción de datos como se observa en la Ilustración 2.18. Mediante un Router-Hub-Punto de acceso (AP) se transmite la señal mediante ondas de radio dentro de un radio de cobertura permitiendo el acceso a internet. [51].Existen dos tipos de tecnología Wifi la 2.4Ghz la cual posee menor velocidad de conexión, pero mayor interferencia y alcance mientras que la 5Ghz posee mayor velocidad de conexión, pero menor alcance e interferencia. [52]

Ilustración 2.18. Tecnología wifi [53]

2.6.3 Lora

Es una tecnología inalámbrica la cual se utiliza en la transmisión de datos en un radio de gran cobertura a baja velocidad entre sensores y actuadores IOT como se observa en la Ilustración 2.19. Lora Wan es la unión de esta tecnología junto a redes WAN permitiendo crear proyectos inteligentes y subirlos a la nube mediante redes IOT dando acceso a la visualización en tiempo real a las KPIs y medidores de variables. [54]

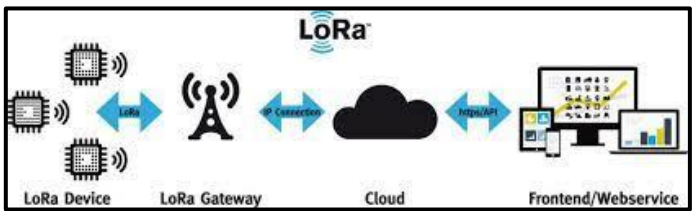

Ilustración 2.19 Tecnología Lora [55]

2.6.4 Zigbee

Es un protocolo de comunicación inalámbrica, trabaja en el nivel bajo de las capas de comunicación, trabaja con topología tipo malla posee bajo consumo de energía debido a que trabaja mayormente

con baterías, al ser tecnología inalámbrica su tasa de transmisión de datos es a baja velocidad, se utiliza en mayor parte en el área de domótica como se observa en la Ilustración 2.20 debido al coste por proyecto que resulta económico, con la ventaja de poder enviar los datos a la nube mediante IOT [56].

Ilustración 2.20. Tecnología Zigbee [57]

2.7 Módulos de comunicación inalámbricas

2.7.1 Xbee

Es un dispositivo electrónico el cual envía información por medio inalámbrico, trabaja con el protocolo IEEE.802.15.4 mediante el mismo crea redes punto-punto y punto-multipunto, se basa en los estándares de la tecnología Zigbee. Se utilizan en procesos que requieren una latencia mínima pero una alta transmisión de datos [58].

2.8 Microcontroladores

Un microcontrolador es un dispositivo capaz de realizar distintas funciones y con la capacidad de reprogramarse. Su estructura consta de entradas, salidas y memoria tanto de almacenamiento como para el control. El hardware completo de un microcontrolador se encuentra dentro de un chip, y es mediante un software que se especifican las funciones a realizar.

Los bloques que básicamente componen un microcontrolador son: CPU, memoria ROM, memoria RAM, entradas y salida. El CPU es el responsable de procesar las instrucciones que se han configurado. A su vez el CPU también cuenta con una memoria interna encargada de almacenar

registros de 8 o 32 bits. La manera en la que estos bloques interactúan entre si será definida por la arquitectura, entre las principales están: Von Neuman y Harvard.

La arquitectura Von Neuman cuenta con un CPU el cual contiene un ALU (Unidad Aritmética Lógica) y una unidad de control. También posee una memoria principal, una secundario, entradas y salidas. El procedimiento que realiza es tomar los datos de la memoria principal, procesarlos y devolverlos a la memoria principal. La memoria se encarga de almacenar tanto las instrucciones como los datos generados, por esto es por lo que en sus inicios se podían presentar cuellos de botella y reducir el rendimiento.

Por otro lado, la arquitectura Harvard almacena de forma separada los datos y las instrucciones a realizar. Se cuenta con un UAL, memoria para instrucciones, memoria para datos, CPU, entradas y salidas.

2.9 Domótica

Es el proceso que se realiza en un espacio determinado en conjunto mediante tecnología y redes, en actividades de dispositivos electrónicos, sin presencia del ser humano es decir crear un sistema automatizado ya sea controlando electricidad-agua-ventilación-luz entre otros, todo dispositivo que tenga conexión a una red se puede automatizar y unirse al proceso domótico. [61]

2.10 Sensores

Son dispositivos eléctricos que se encargan de medir o trabajar junto a una variable especifica, para generar un proceso mediante una alerta con condiciones establecidas. Van sumamente relacionados con el mundo de la domótica ya que mediante los mismos automatizan procesos.

2.10.1 Sensor de temperatura

Se encarga de medir la variable de temperatura mostrando una señal digital o analógica, es utilizado comúnmente en sistemas domóticos, trabajando en modo de alerta para activar sistemas o un proceso determinado ya sea activar un sistema hidráulico o un sistema de ventilación. [62]

2.10.2 Sensor de movimiento

Es un dispositivo electrónico se utiliza en el ámbito domótico en proyectos de ventilación y de iluminaria, ya que detecta el movimiento o presencia de objetos en un área determinada, ya sea por rayos infrarrojos o por ondas ultrasónicas al detectar activa una función preestablecida generando un proceso de automatización. [63]

2.10.3 Sensor de humo

Es el componente que sirve de alarma para el detector de humo, se utiliza para prevenir incendios, trabaja internamente con un sensor fotoeléctrico el cual es un diodo el mismo genera una señal continua en condiciones normales, pero en presencia de humo la señal no es continua lo cual produce una alarma activando el sistema. [64]

CAPÍTULO 3

3 DISEÑO DE LA SOLUCIÓN

En esta sección se hablará sobre la solución propuesta basada en la investigación realizada en el capítulo anterior. En primer lugar, se realizó un diagrama de bloques (Ilustración 3.1) el cual servirá de guía para definir cada uno de los procesos a realizar durante el diseño de la solución propuesta.

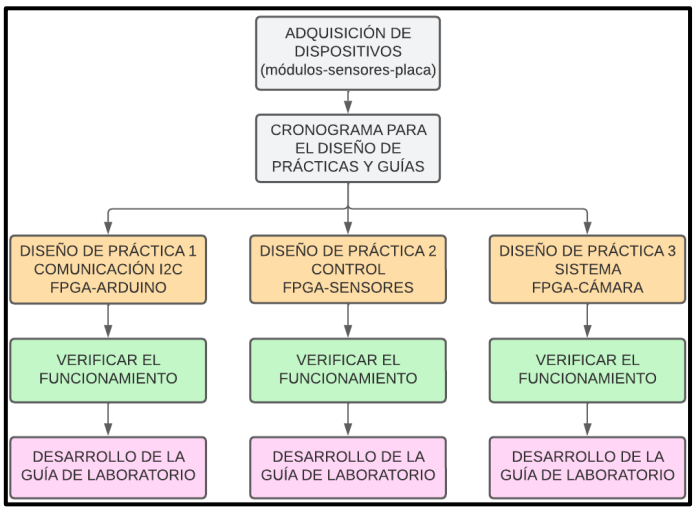

Ilustración 3.1 Diagrama de bloques general.

Para la primera practica se debe realizar una comunicación serial I2C entre la FPGA y el módulo Arduino como se observa en la Ilustración 3.2 la placa se configura en modo esclavo por ende el microprocesador cumple la función de modo maestro como se observa en el primer bloque.

Se conecta por el BUS I2C debido a las librerías que presenta Arduino en donde solo se utiliza 2 pines, a la par se necesita un Core en la FPGA el cual es el I2C slave, el cual se comporta como una máquina de estados donde recibe datos que le llegan desde el BUS I2C.

Los datos que envía Arduino se modifican mediante la configuración VHDL de la FPGA en donde se pueden realizar cambios aritméticos, posterior a ello se visualizaran los resultados en una interfaz gráfica.

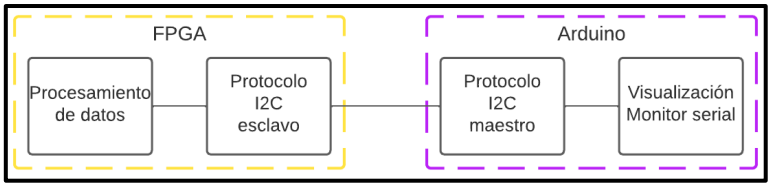

Ilustración 3.2 Diagrama de bloques practica 1.

Para el diseño de la segunda práctica se tiene el diagrama observado en la Ilustración 3.3. El bloque de adquisición de datos se compone de dos circuitos electrónicos que conforman un sensor de temperatura y uno de movimiento, ambos proporcionan señales analógicas. Estas señales ingresan a la FPGA, específicamente al bloque conversor analógico a digital. Mediante el software Vivado se programa el procesador de la placa para que realice la conversión de las señales.

En Vivado también se programa el procesador para que mediante el protocolo UART las señales digitales sean direccionadas hacia los correspondientes pines de salida de la FPGA. Aquí finaliza la configuración y el trabajo en la FPGA.

En el bloque de comunicación Zigbee se maneja la parte inalámbrica. Los pines para la comunicación UART de un módulo Xbee (emisor) se conectan a los pines de salida UART en la FPGA, este módulo se encargará de la transmisión inalámbrica de los datos. El puerto micro USB de un segundo Xbee se conecta al puerto USB de una PC mediante cable, este segundo Xbee se encargará de la recepción. Ambos módulos serán configurados mediante el software XCTU.

El bloque de visualización se maneja en una PC, aquí se configurará una interfaz en LabVIEW. Se manejará un visualizador tanto para el sensor de temperatura como para el de movimiento. Estos bloques son los que componen el diseño de la segunda práctica, más adelante se indicara específicamente las configuraciones a realizar.

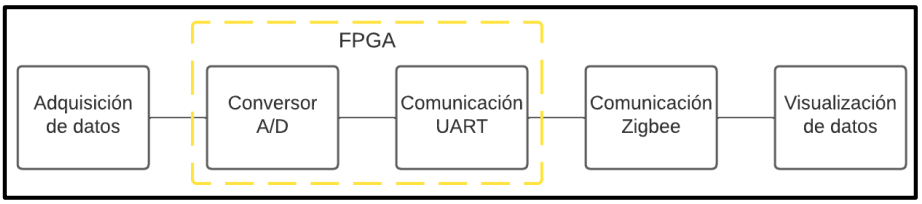

Ilustración 3.3 Diagrama de bloques de la práctica 2.

En la práctica 3 se necesita visualizar imágenes y video a tiempo real, para lo cual se utiliza el protocolo I2C para comunicar el módulo OV7670 hacia la FPGA como se puede observar en la Ilustración 3.4

El Arduino actuará en modo maestro mientras que la FPGA en modo esclavo, así el microprocesador enviará datos mediante el protocolo I2C hacia el esclavo. Dentro de la FPGA se programarán los procesos a realizar a la información recibida desde el Arduino.

Para la visualización de las imágenes se utilizará un monitor, se conectará la salida VGA de la FPGA con una pantalla que permitirá visualizar los datos enviados y el procesamiento realizado en ellos.

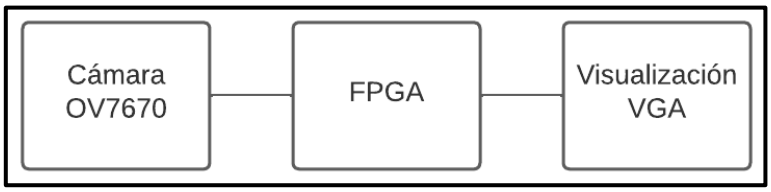

Ilustración 3.4 Diagrama de bloques de la practica 3

3.1 Diseño del prototipo

Se procederá a presentar el diseño de los prototipos de cada práctica, indicando los equipos a utilizar y las conexiones necesarias.

El prototipo de la primera practica se observa en la Ilustración 3.5 en donde se puede apreciar una conexión de 3 cables, siendo 1 de tierra común entre ambos dispositivos y los otros 2 restantes se utilizan para poder realizar la comunicación I2C donde el pin SCL es la señal de reloj y el pin SDA es la señal de datos. En este protocolo los dispositivos pueden trabajar en modo escritura o lectura.

El dispositivo maestro, en este caso el Arduino Uno, es el que emitirá una señal de reloj para definir el sincronismo con la FPGA. Para la programación de la FPGA se realizó un código en VHDL para el I2C slave, junto con un programa debounce para que no existan fallos durante la comunicación.

Luego, se realiza un tercer archivo VHDL general en el cual se establecerán dos entradas las cuales son clk y reset; y otras dos salidas/entradas las cuales son SCL que se encarga de la señal del reloj y una SDA que se encarga de la señal de datos.

En el dispositivo Arduino se genera un único código maestro mediante la librería Wire. Se define la dirección correspondiente del slave para lograr definir la comunicación, y mediante el monitor serial de Arduino se observan los datos enviados y recibidos.

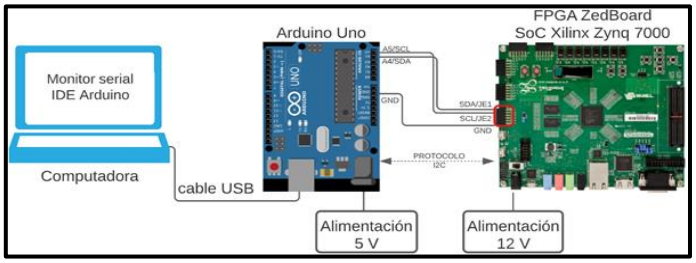

Ilustración 3.5 Prototipo de la práctica 1.

El prototipo de la segunda práctica se observa en Ilustración 3.6. Se tienen dos bloques de sensores, uno de temperatura y otro de movimiento conectados a los pines de entrada del ADC. A continuación, el procesador realizará la conversión de las señales para luego dirigirlas, mediante el protocolo de comunicación UART, hacia el Pmod (módulo periférico) JE.

Los pines de este módulo se conectarán hacia los correspondientes pines UART del módulo Xbee, permitiendo así que las señales se transmitan de forma inalámbrica. El segundo módulo Xbee se conecta mediante su puerto micro USB hacia el puerto USB de la PC. Finalmente, mediante un programa en LabVIEW se leerá el puerto serial al cual se conecta el Xbee, permitiendo así visualizar las señales censadas.

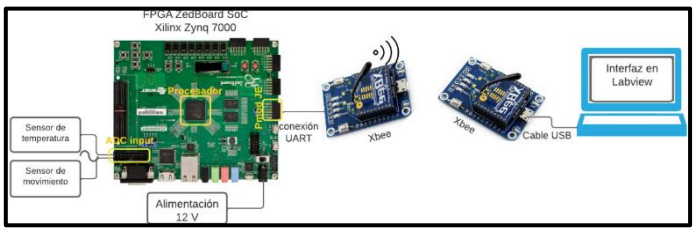

Ilustración 3.6 Prototipo de la práctica 2.

En cuanto a los sensores, fueron realizados con electrónica básica. El sensor de movimiento está conformado por un par emisor-receptor infrarrojo, en la Ilustración 3.7 se tiene el diagrama esquemático. Un fotodiodo (emisor) junto con una resistencia (R1) es polarizado a 5 voltios, de igual forma se visualiza la polarización de un fototransistor (receptor). Mientras esté polarizado el fotodiodo emitirá constantemente una luz infrarroja, la cual será recibida por el fototransistor ubicado al frente de este.

Mientras el fototransistor se encuentre recibiendo luz trabajará prácticamente como un cable (cero voltios), permitirá el paso de corriente hacia la resistencia (R3) conectada en su pin emisor. Por otro lado, al momento de verse cortada el flujo de luz el fototransistor dejará de conducir trabajando prácticamente como circuito abierto. La señal para medir será el voltaje Vout1 de la resistencia R3, ya que variará su valor según haya o no luz.

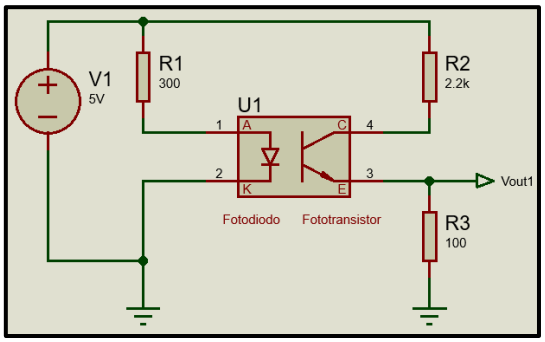

Ilustración 3.7 Diagrama esquemático del sensor de movimiento.

Para el sensor de temperatura se utilizó un LM35, este es un sensor que arroja 10mV por cada grado centígrado. En la Ilustración 3.8 se observa el diagrama esquemático del sensor, se encuentra polarizado por una fuente de 5V y en su pin 2 (Vout2) está el voltaje para medir.

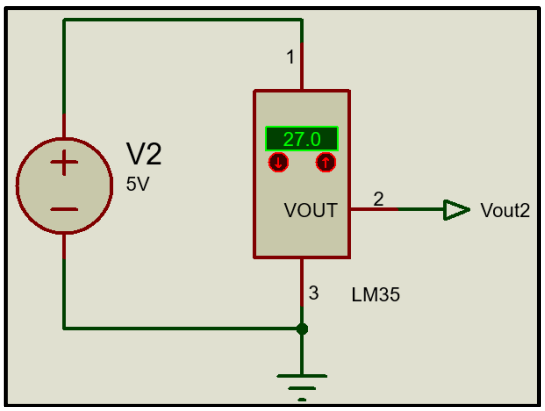

Ilustración 3.8 Diagrama esquemático del sensor de temperatura.

El prototipo de la tercera practica se observa en la Ilustración 3.9, consiste en conectar los 18 pines del módulo OV7670 hacia la FPGA, específicamente en los Pmod JB-JC-JD. Los pines están divididos en 8 digitales, 2 correspondientes a Vcc y GND, 2 van a trabajar como señal de reloj, 2 que trabajaran como líneas SDA-SCL, 2 de señal de pixel de reloj, 1 de sincronización vertical y 1 de sincronización horizontal.

La primera parte de la práctica se encargará de establecer las conexiones mediante bloques configurados en hardware con Verilog, así se controla la imagen-video. La segunda parte se compone en la visualización mediante la construcción del bloque que controlará la salida VGA, para finalmente conectar una pantalla a esta. Hay que tener en cuenta que la FPGA cuenta con varios pines de 3.3V con los cuales se logrará energizar el módulo OV7670.

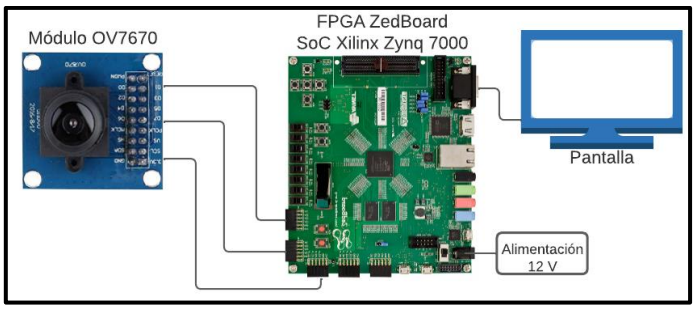

Ilustración 3.9 Prototipo de la práctica 3

3.2 Implementación del prototipo

En esta sección se hablará sobre cada uno de los elementos utilizados, así como las configuraciones necesarias para el correcto funcionamiento de cada una de las prácticas.

3.2.1 Implementación de la práctica 1

Para la primera practica se implementó un protocolo I2C entre FPGA y Arduino, para ello se requieren los materiales mencionados a continuación.

3.2.1.1 Materiales y Costos

En la Tabla 3.1 se detallarán los materiales utilizados para el desarrollo de la primera práctica.

Material	Cantidad	Detalle	Costos
Cable USB – micro USB	1		$ 2
Arduino	1		$ 20
FPGA ZedBoard	1	Xilinx Zynq®-7000 SoC	$ 475
Jumpers	varios		$ 1
Fuente 12V	1	Incluida con la FPGA	$ 1

Tabla 3.1 Materiales y costos practica 1

Los materiales en los que no se observan precios, vienen incluidos en el paquete de la FPGA. Sumando todos los valores se obtiene total de $499.00, aunque se debe considerar que el paquete de la placa fue provisto por la universidad.

3.2.1.2 Programación en VIVADO

El desarrollo de la siguiente practica de laboratorio requiere la conexión de la FPGA y el Arduino uno, dichos dispositivos se pueden visualizar en la Ilustración 3.10.

Ilustración 3.10 FPGA y Arduino

Se desarrollaron 3 archivos VHDL como se observa en la Ilustración 3.11 el primer archivo VHDL consiste en el bloque controlador I2c_Slave, el segundo archivo consiste en un bloque debounce el cual será un respaldo para que el archivo principal no sufra inconvenientes y por último el archivo I2C_prueba el cual procesara las señales que requiera el prototipo.

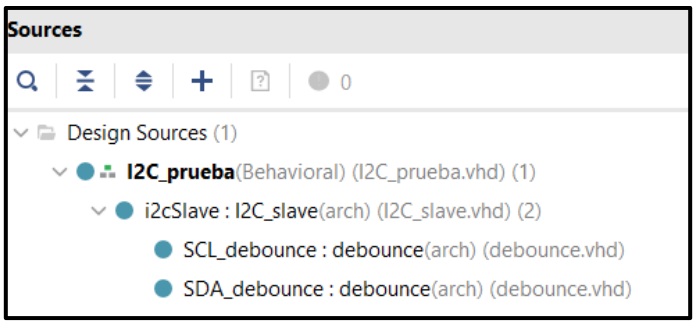

Ilustración 3.11 Archivos VHDL Protocolo I2C

Al momento de compilar los archivos en VHDL se procede a realizar el RTL análisis, lo cual permite desarrollar diagramas de bloques en base a los códigos pre desarrollados, tal como se puede observar en la Ilustración 3.13.Para corroborar que los bloques no presentan errores se puede ver la información de compilación en la Ilustración 3.12.

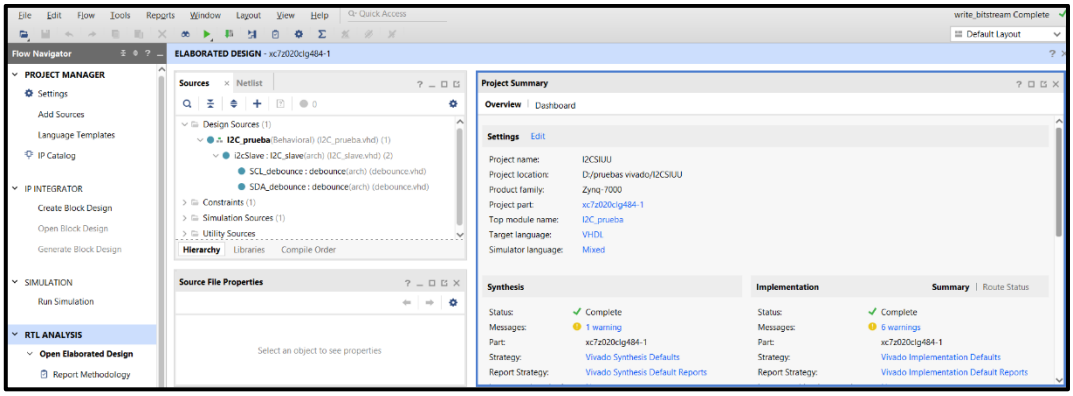

Ilustración 3.12 compilación del RTL ANALYSIS

En la Ilustración 3.13 se observa el bloque I2C_Slave el cual presenta 4 entradas y 3 salidas internas, se debe tener en cuenta que el protocolo I2C soporta solo buses de datos de 7 bits, aunque teóricamente son 8, ya que el bit suelto se encarga de identificar si el dispositivo trabaja en modo de escritura o lectura.

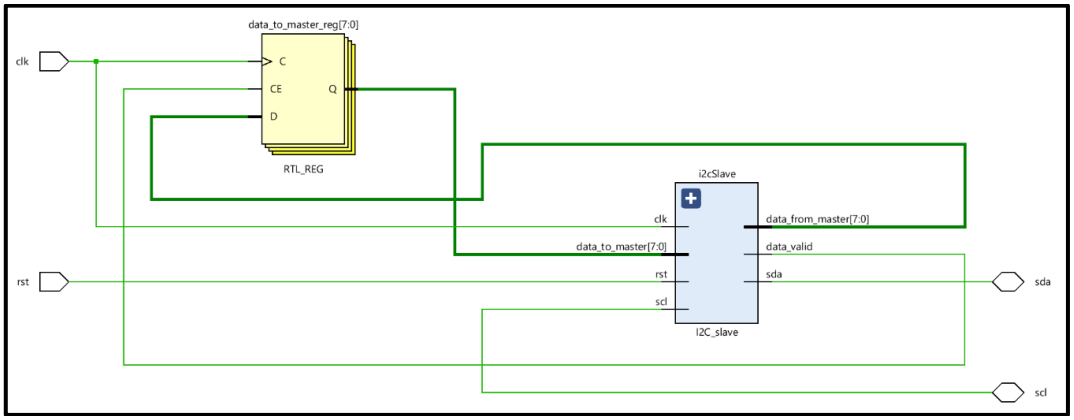

Ilustración 3.13 Diagrama de bloques producido por el RTL ANALYSIS

En la Ilustración 3.13 se observa las líneas clk y rst en el lado izquierdo las cuales son entradas del protocolo I2C las cuales se encargan de controlar el proceso que genera las entradas y salidas del lado derecho, las cuales son sda (datos) en esa línea se envían los buses de datos de información con la respectiva dirección en cambio la línea scl (reloj), se encarga de generar el sincronismo entre los dispositivos en este caso el Arduino y la FPGA.

En la Ilustración 3.14 se escoge la opción de Generate Bitstream-Open Hardware Manager-Open Target lo cual sirve para poder crear el archivo de compilación para cargar a la FPGA.

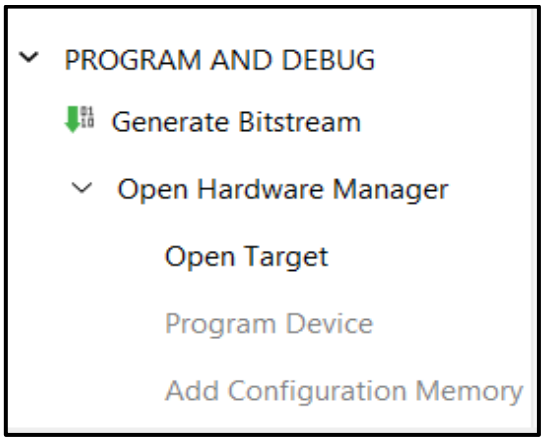

Ilustración 3.14 Generate Bitstream practica 1

Al completar los pasos, se genera la Ilustración 3.14 donde se observa el local host, que significa la dirección del dispositivo.

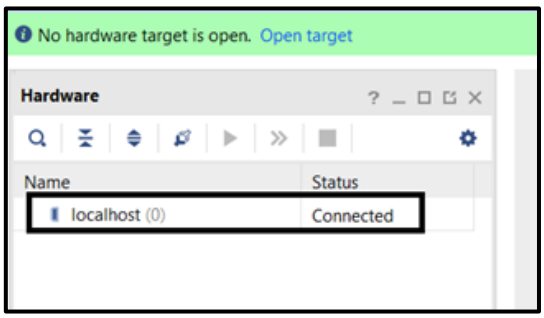

Ilustración 3.15 Local host.

Al momento de encender la FPGA se detecta el dispositivo, como se observa en la Ilustración 3.16, en donde local host aparece conectado.

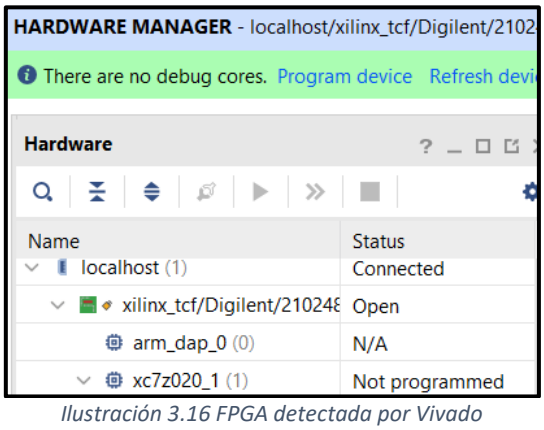

Ilustración 3.16 FPGA detectada por Vivado

Al finalizar la programación del dispositivo en la pestaña Hardware se observa en la Ilustración 3.17 el proceso finalizado.

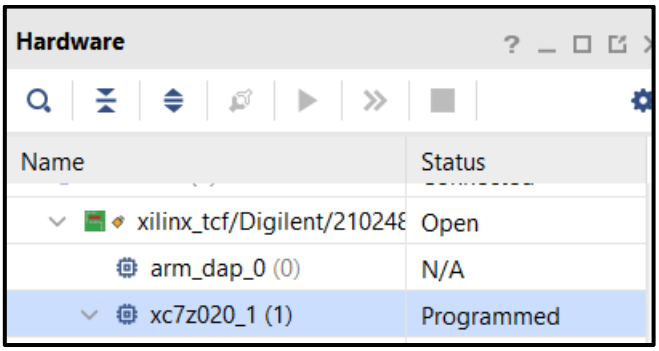

Ilustración 3.17 FPGA programada de la practica 1

3.2.1.3 Programación en Arduino

Para la implementación de I2C en Arduino se cuenta con la librería Wire. En primer lugar, se definen las librerías a utilizar, en este caso Wire. A continuación, se definen las variables a utilizar durante el programa. En la sección de "void setup" se inicializan las librerías tanto Wire para trabajar con I2C, como Serial que permitirá visualizar los datos en el monitor serial. Dentro de "void loop" se observarán los procesos a realizarse de forma continua en Arduino. Inicialmente se pedirá ingresar un número y se lo almacenará en la variable "num"; el código descrito se observa en la Ilustración 3.18.

I2C_master

```
#include <Wire.h> // se inicializa la librería de I2C

// se definen variables y mensajes a presentar
byte num;
int out;
String mensaje1="Enviar a FPGA: ";
String mensaje2="Recibido de FPGA: ";

void setup(){
 Wire.begin(); // se inicializa librería I2C
 Serial.begin(9600); // se inicializa librería serial
}

void loop(){

    // se ingresa el dato a enviar a la FPGA
    Serial.print(mensaje1);
        while(Serial.available()==0)
      {
      }
    num=Serial.parseInt();
    Serial.println(num);
```

Ilustración 3.18 Código de Arduino parte 1.

El dato ingresado se envía hacia el esclavo de dirección 3, junto con un "H" que le indicará a la FPGA en qué estado colocarse. Una vez enviados los datos se procede a hacer un request del dato

procesado por la FPGA. Una vez se recibe el nuevo valor se procede a mostrarlo mediante el monitor serial, junto con un mensaje. Esta sección del código se observa en la Ilustración 3.19.

```
    // se envian datos hacia la FPGA
    Wire.beginTransmission(0x3); // inicio de transmisión con dirección 3
    Wire.write("H"); // se envía una H para activar el estado
    Wire.endTransmission();

    Wire.beginTransmission(0x3);
    Wire.write(num); // se envía el número ingresado
    Wire.endTransmission();

    delay(1);

    // se reciven datos de la FPGA
    Wire.requestFrom(3,1); // se piden datos al slave con dirección 3
    while(!Wire.available()){;}
    out=Wire.read(); // se lee el dato enviado por la FPGA

    // impresión del número ingresado (num) + 5
    Serial.print(mensaje2);
    Serial.println(out);
    delay(1000);
}
```

Ilustración 3.19 Código de Arduino parte 2.

3.2.1.4 Conexión FPGA-Arduino

Los pines por conectar entre la FPGA y el Arduino se muestran en la Tabla 3.2. Mientras que las conexiones físicas se presentan en la Ilustración 3.20, donde el clave verde pertenece a la línea SDA, el clave rojo a la línea SCL y el clave naranja a la línea de tierra.

FPGA	ARDUINO
SDA PIN JA1	SDA PIN A4
SCL PIN JA2	SCL PIN A5
GROUND	GROUND

Tabla 3.2 Conexión FPGA-Arduino

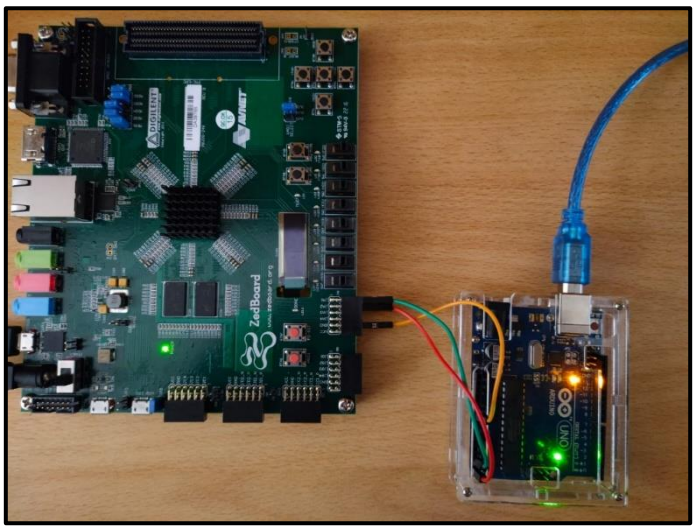

Ilustración 3.20 Conexiones FPGA-Arduino

3.2.2 Implementación de la práctica 2

Para la segunda práctica se implementará un sistema de sensores a través de una FPGA y con comunicación inalámbrica, para ello se requieren de los materiales mencionados a continuación.

3.2.2.1 Materiales y costos

En la Tabla 3.3 se detallarán los materiales utilizados para el desarrollo de la segunda práctica, así como sus correspondientes costos.

Material	Cantidad	Detalle	Costo
Cable USB – micro USB	1	Incluida con la FPGA	
Xbee	2	S2C	$ 40 c/u
Xbee Explorer	2	Mini USB	$ 10 c/u
FPGA ZedBoard	1	Xilinx Zynq®-7000 SoC	$ 475
Jumpers	varios		$ 1
Fuente 12V	1	Incluida con la FPGA	
Sensor LM35	1		$ 0.35
Fuente 5V	1	Obtenida de la FPGA	$ 5
Emisor infrarrojo	1		$ 0.35
Receptor infrarrojo	1		$ 0.35
Resistencias	3	300 Ohms, 2.2k Ohms, 2.2k Ohms	$ 0.05 c/u
Transistor 2N2222	1		$ 0.50
Diodo Led	1	Rojo	$ 0.20
Fuente 12V	1	Obtenida de la FPGA	

Tabla 3.3 Materiales de la práctica 2.

Los materiales en los que no se observan precios, vienen incluidos en el paquete de la FPGA. Sumando todos los valores se obtiene total de $582.90, aunque se debe considerar que el paquete de la placa fue provisto por la universidad.

3.2.2.2 Implementación de los sensores

En la Ilustración 3.21 se observa la implementación de los sensores. En el literal a) se tiene al par emisor-receptor uno frente al otro, polarizados y con sus respectivas resistencias; en el pin emisor del fototransistor se observa un cable naranja de donde se obtiene la señal Vout1 a medir. En el literal b) se tiene al sensor LM35 polarizado y con su pin de salida conectado al cable azul, en este se medirá la señal Vout2 de interés.

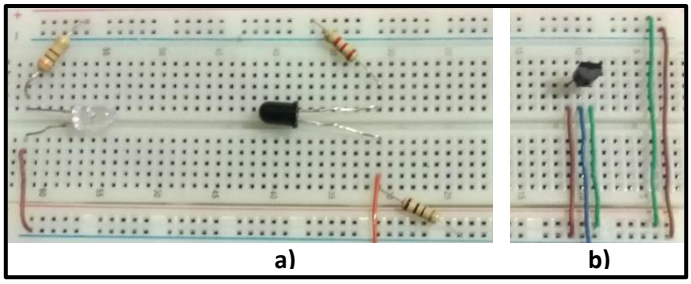

Ilustración 3.21 Sensores en protoboard: a) movimiento y b) temperatura.

3.2.2.3 Configuración del procesador

Para la configuración de la FPGA se utilizó el software Vivado. Habiendo creado un proyecto y un diseño de bloques, se colocan los componentes de interés como se observa en la Ilustración 3.22 donde se tienen al bloque ZYNQ7 que corresponde al procesador y el bloque XADC wizard que comprende al conversor A/D.

En el XADC se activa la opción de Channel Sequencer para utilizar varias entradas del conversor, se deshabilitan todos los offsets, alarmas, sensores y se habilita un Channel averaging de 16. Finalmente se activan los voltajes auxiliares a utilizar, en este caso en vaux0 y vaux8 se activan las columnas de channel y average. Con estas configuraciones se indica al conversor que recibirá dos señales por medio de sus entradas vaux0 y vaux8, además de deshabilitar los sensores y alarmas. En la Ilustración 3.23 e Ilustración 3.24 se observan las configuraciones realizadas.

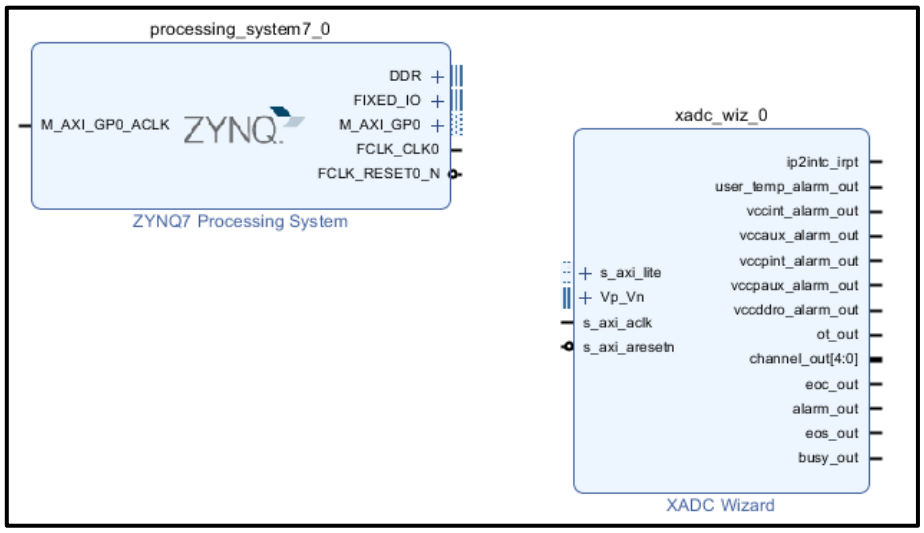

Ilustración 3.22 Bloques del procesador y ADC.

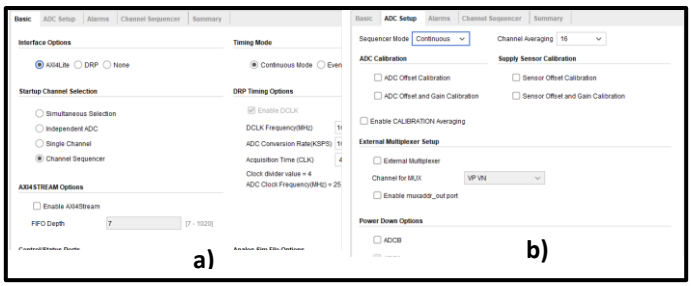

Ilustración 3.23 Configuración del XADC: a) basic, b) ADC setup.

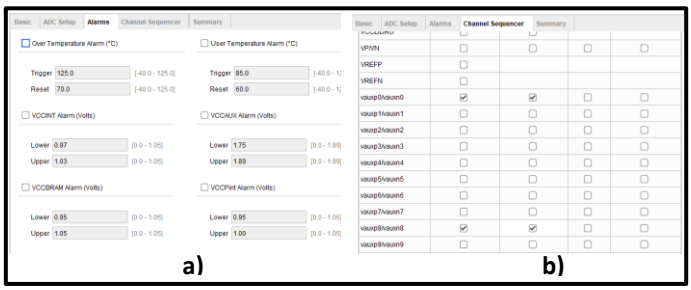

Ilustración 3.24 Configuración del XADC: a) alarms, b) channel sequencer.

Una vez trabajado el XADC se configura el procesador ZYNQ7, se dará doble click sobre el desplegándoce una ventana para configurarlo. Se le indicará que habilite el protocolo UART 0 para la transmisión de los datos hacia el exterior, esto lo realizará mediante el Pmod (Módulo periférico) JE. Se establecen los puertos MIO (Multiplexed Input/Output) 10 y 11 a utilizar, así como sus características entre ellas: velocidad, nivel de voltaje a utilizar, pull-up. En la Ilustración 3.25 se observa la configuración realizada en la pestaña MIO configuration de la ventana desplegada.

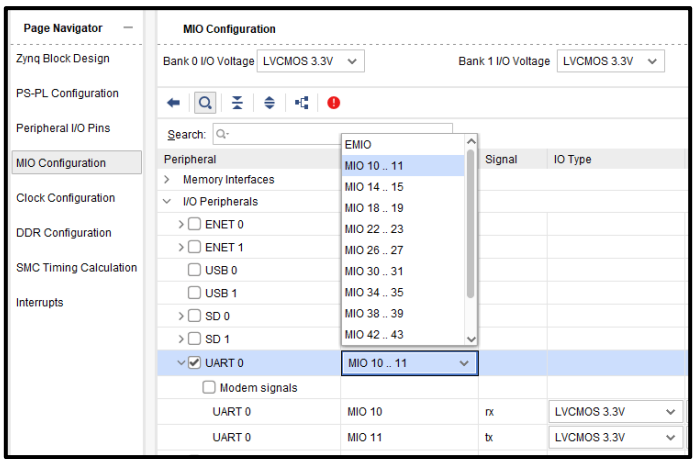

Ilustración 3.25 Configuración del protocolo UART 0 y sus puertos.

Ahora se procede a expandir las señales vaux0 y vaux8 del bloque XADC, se observarán nuevos valores “p” y “n” para ambas. Se selecciona cada una y con click derecho se elige la opción make

external, obteniendo así dos entradas (p y n) para cada vaux (voltaje auxiliar) tal como se observa en la Ilustración 3.26.

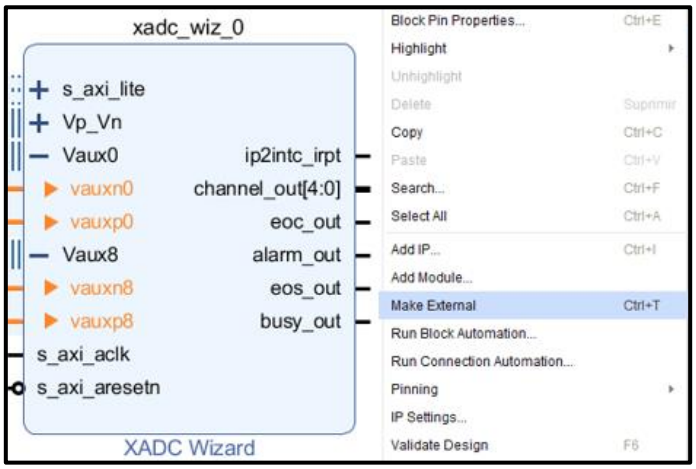

Ilustración 3.26 Definiendo entradas del XADC.

Una vez configurados ambos bloques se presentan como se visualiza en la Ilustración 3.27. A continuación, se automatizan las conexiones y los bloques, además de validar el diseño. El diagrama obtenido se observa en la Ilustración 3.28, los bloques AXI interconnect y Processor System Reset se generan automáticamente por el software.

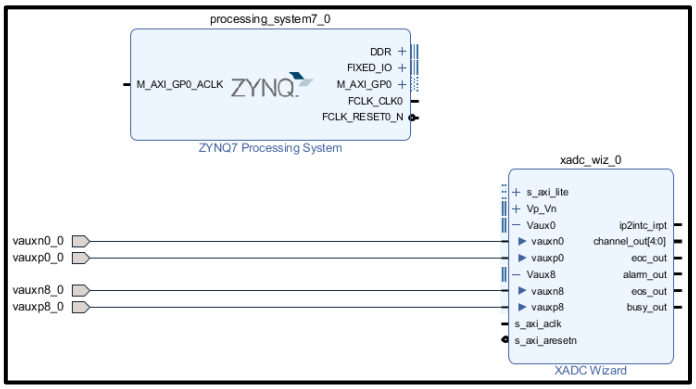

Ilustración 3.27 Bloques XADC y ZYNQ7 configurados.

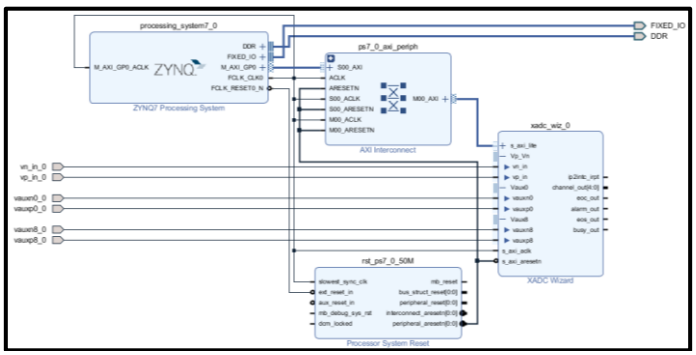

Ilustración 3.28 Diseño final del procesador y conversor A/D.

Una vez creados los bloques se procede a crear el HDL Wrapper, obteniendo dos nuevos archivos tal como se presenta en la Ilustración 3.29. Finalmente se genera el Bitstream (Ilustración 3.30) que se encargará de programar la FPGA, durante este proceso se realizará la síntesis e implementación.

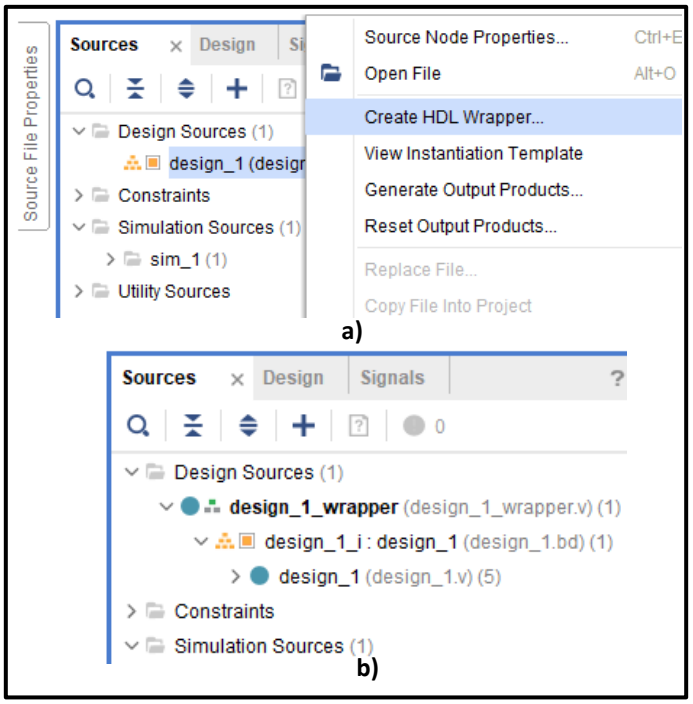

Ilustración 3.29 a) creación de HDL Wrapper, b) archivos generados.

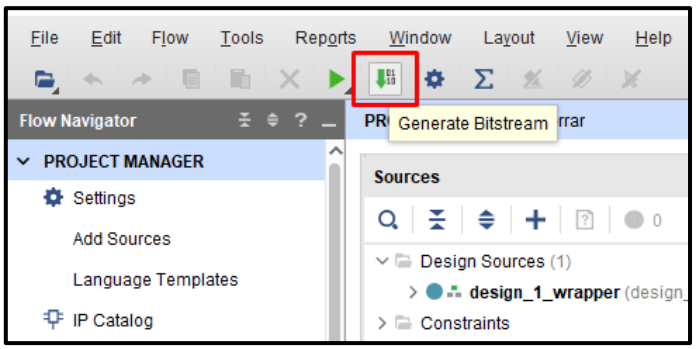

Ilustración 3.30 Generando bitstream.

Una vez generado el bitstream se procede a exportar el hardware, generando un archivo con extensión xsa. Al exportar el hardware se debe incluir el bitstream, y se aconseja que la ruta a guardar del archivo sea la misma que la del proyecto de Vivado. En la Ilustración 3.31 se describe el proceso y en la Ilustración 3.32 se observa el archivo generado.

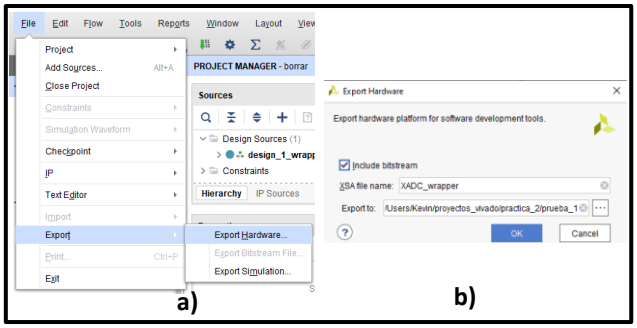

Ilustración 3.31 a) exportando hardware, b) ruta e inclusión del bitstream.

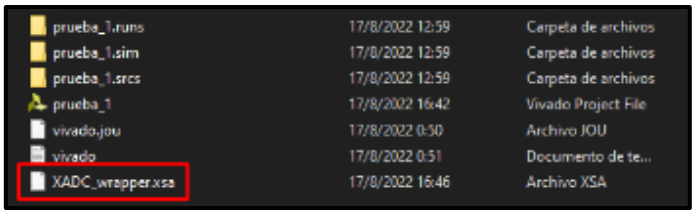

Ilustración 3.32 Archivo XSA generado.

Ya que se está trabajando con bloques predefinidos es necesario generar una plataforma en la que se codificará las instrucciones para utilizar dichos bloques. En este caso se leerá y presentará las señales censadas por la FPGA, dicha plataforma estará basada en el archivo xsa generado. Cabe recalcar que el hardware (ADC y UART) ya se encuentra definido en el archivo xsa, la plataforma que se creará en Vitis le indicará al hardware donde almacenar la información provista por los sensores y cómo presentarla por pantalla.

Al iniciar Vitis se debe seleccionar un workspace, la ruta de este debe ser la misma del archivo xsa para así crear la plataforma en base a este archivo. Se trabajará con el template Hello world en el cual se tendrá una estructura básica de un programa en C. Una vez generada la plataforma aparece una nueva ventana donde se observa el archivo Hello world en C dentro de un proyecto llamado sensores_system. En la Ilustración 3.33 se presenta la plataforma generada.

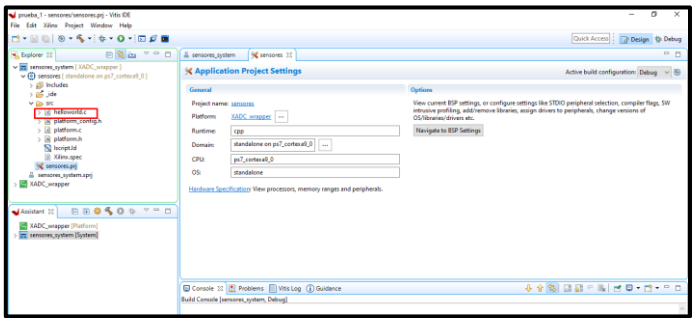

Ilustración 3.33 Plataforma generada.

Se modificará el código de Hello world con las instrucciones necesarias para que el convertidor ADC, definido en el archivo xsa, lea e imprima los valores censados tal como se observa en la Ilustración 3.34. Se realiza un debug del proyecto con el cual se reiniciará el sistema, programará la FPGA y se ejecutará el proyecto.

sensores_system | sensores | *helloworld.c

```c
// librerias
#include <stdio.h>
#include "xsysmon.h"
#include "xparameters.h"
#include "xstatus.h"
#include "xil_exception.h"
#include "xil_printf.h"
#include "sleep.h"

#define C_BASEADDR 0x43C00000 // registro del XADC

int main()
{
    // se definen variables para cada Vaux
    u16 data0 = 0;
    u16 data8 = 0;
    while (1)
    {
        // escritura de datos
        data0 = Xil_In32(C_BASEADDR + 0x240); // la señal convertida A/D en Vaux0 se almacena en el registro
        data8 = Xil_In32(C_BASEADDR + 0x260); // la señal convertida A/D en Vaux8 se almacena en el registro

        // conversion de datos a mV [data*(1/2^16)*1000]
        data0 = (data0 * 0.015258789);
        data8 = (data8 * 0.015258789);

        // impresion de los valores
        xil_printf("%03d %03d",data8,data0);

        usleep(1000000); // retardo de 5s
    }
    return 0;
}
```

Ilustración 3.34 Código en C.

3.2.2.4 Configuración de los módulos Xbee

Para la configuración de los módulos Xbee se utilizó el software XCTU, definiendo así diferentes parámetros para su correcta operación. Los dispositivos trabajarán como coordinador y como router. En la Ilustración 3.35 se observar los valores más importantes a configurar en el módulo coordinador, entre ellos están: PAN ID, habilitar el modo coordinador, dirección en alto y dirección en bajo. El PAN ID es un número asignado a la red, mientras que la dirección en bajo indicará el dispositivo con el que puede comunicarse el coordinador; dicho valor estará asociado con el módulo router.

Por otro lado, en la Ilustración 3.36 se observan las configuraciones necesarias para que el segundo Xbee trabaje como router. El PAN ID definido será el mismo que en el coordinador, mientras que la dirección en bajo indicará que este módulo se comunicará con el coordinador. Finalmente, en la pestaña sleep mode se define el modo de operación, en este caso como router.

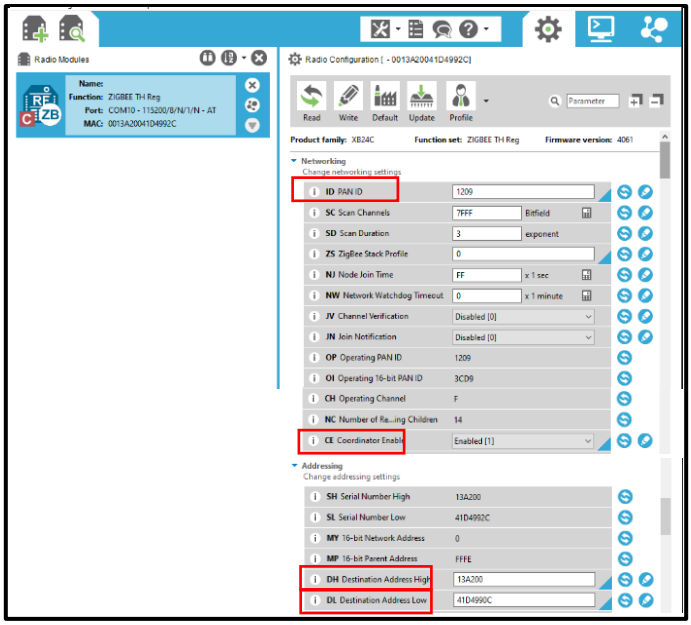

Ilustración 3.35 Configuración del Xbee coordinador.

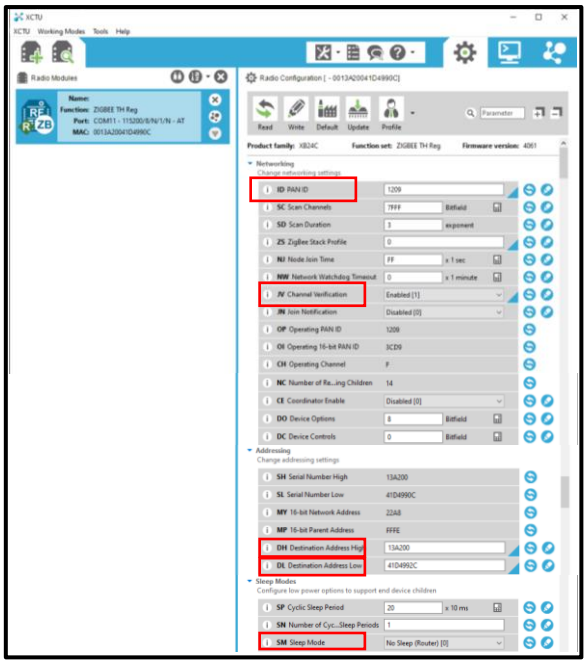

Ilustración 3.36 Configuración del Xbee router.

3.2.2.5 Interfaz en LabVIEW

Para visualizar los datos recibidos por el módulo Xbee conectado a la PC, se diseñó una interfaz en LabVIEW la cual permite visualizar de mejor forma las señales censadas. El diagrama de bloques realizado utiliza bloques VISA para inicializar y cerrar el puerto serial a leer, junto con un bucle while para presentar constantemente los valores del puerto. Dentro del bucle se leerán los datos, con el bloque Scan from string se separarán los valores recibidos para presentarlos en diferentes indicadores.

Para el indicador de movimiento se utiliza un comparador ya que la señal tendrá determinado valor al haber presencia de obstáculos o no entre el par emisor-receptor. Para el indicador de temperatura se realiza una división para diez ya que los valores recibidos son centenas. En la Ilustración 3.37 se observa el diagrama descrito.

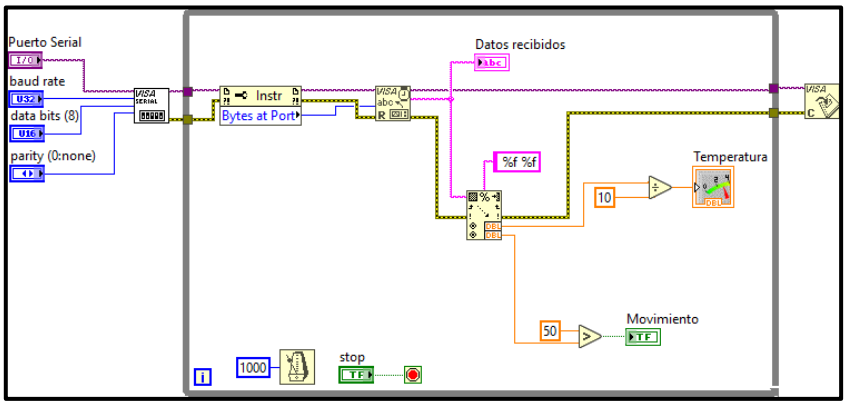

Ilustración 3.37 Diagrama de bloques de la interfaz.

En el panel frontal de la interfaz se observan los controles para el puerto serial, baud rate, paridad y los bits de información. Se presentan los indicadores de temperatura mediante un bloque meter, mientras que para el indicador de movimiento se utiliza un led; también se colocó un bloque para visualizar los valores originalmente recibidos. Esta interfaz se observa en la Ilustración 3.38.

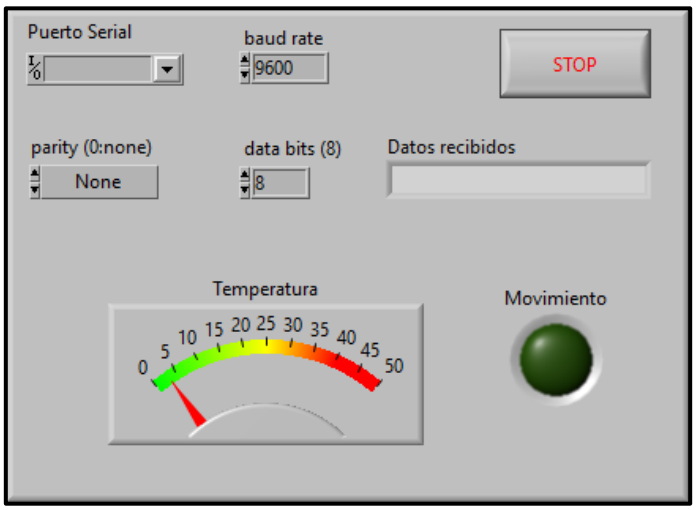

Ilustración 3.38 Panel frontal de la interfaz.

3.2.2.6 Conexiones con la FPGA

Uno de los módulos Xbee se conectará con el Pmod JE de la FPGA, estableciendo así comunicación mediante el protocolo UART. A continuación, en la Tabla 3.4 se definirán los pines utilizados y su visualización en la Ilustración 3.39:

FPGA - JE	Xbee	Detalle
JE2	Rx	Cable blanco
JE3	Tx	Cable cafe
5 V	Vcc	Cable amarillo
GND	GND	Cable verde

Tabla 3.4 Pines de conexión FPGA – Xbee.

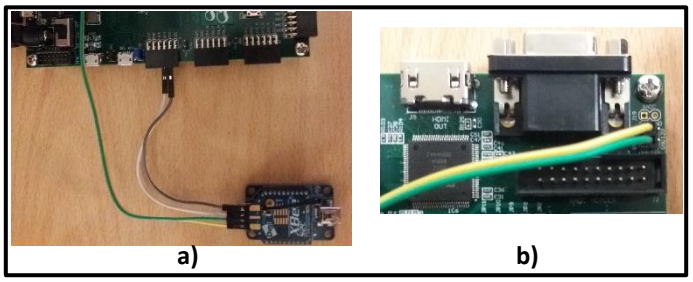

Ilustración 3.39 Conexiones a) Xbee – JE, b) Xbee – 5V y GND.

Para la conexión entre la FPGA y los sensores se presenta la Tabla 3.5, así también se visualiza el cableado en la Ilustración 3.40.

FPGA – Heder XADC	Sensores	Detalle
3	Emisor del fototransistor	Vaux0P, cable naranja
6	GND	Vaux0N, cable blanco
7	GND	Vaux8N, cable blanco
8	Salida del LM35	Vaux8P, cable rojo

Tabla 3.5 Pines de conexión entre FPGA - sensores.

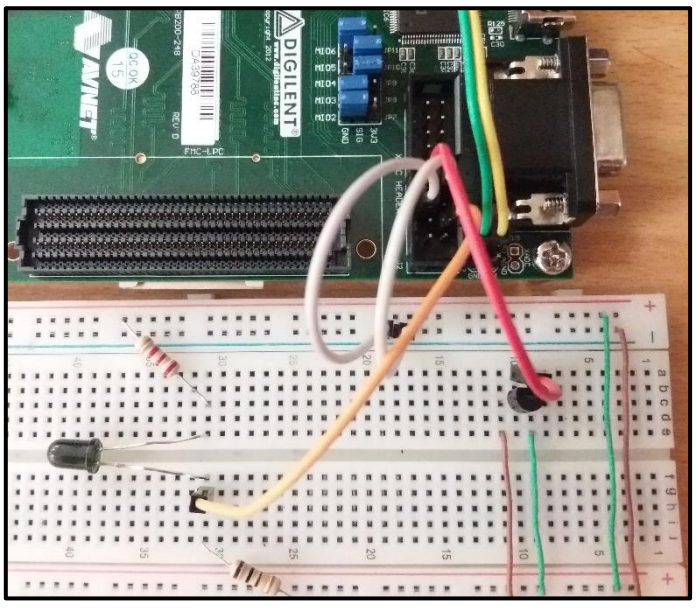

Ilustración 3.40 Conexiones sensores - XADC Header.

El segundo módulo Xbee se conecta a la PC mediante el cable USB – micro USB. Finalmente, en la Ilustración 3.41 se observa la conexión de todos los componentes del sistema.

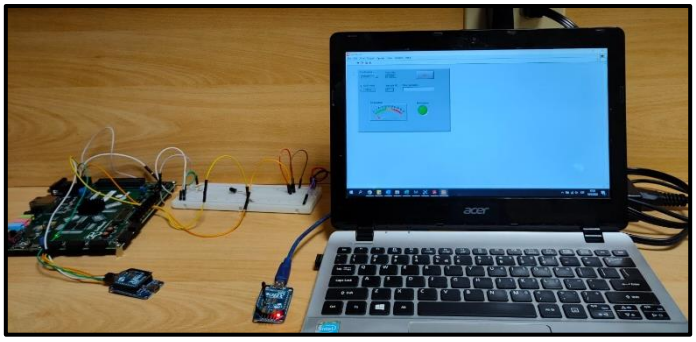

Ilustración 3.41 Conexiones del sistema completo.

3.2.3 Implementación de práctica 3

3.2.3.1 Materiales y Costos

En la Tabla 3.6 Materiales y costo de la práctica 3. se detallan los materiales utilizados para el desarrollo de la tercera práctica.

Material	Cantidad	Detalle	Costos
Cable USB – micro USB	1		$ 1
FPGA ZedBoard	1	Xilinx Zynq®-7000 SoC	$ 475
Jumpers	varios		$ 1
Fuente 12V	1	Incluida con la FPGA	
Modulo ov7670	1	Modulo de camara	$ 8
Fuente 12V	1	Obtenida de la FPGA	

Tabla 3.6 Materiales y costo de la práctica 3.

Los materiales en los que no se observan precios, vienen incluidos en el paquete de la FPGA. Sumando todos los valores se obtiene total de $485, aunque se debe considerar que el paquete de la placa fue provisto por la universidad.

3.2.3.2 Programación Vivado

El desarrollo de la siguiente practica de laboratorio requiere la conexión de la FPGA y el módulo OV7670, dichos dispositivos se pueden visualizar en Ilustración 3.42.

En la práctica número 3, el sistema de cámara FPGA y módulo OV7670 se programó mediante Verilog. Se crearon varios archivos .sv a la par de la incorporación de bloques, como se puede observar en la Ilustración 3.43, que permitirían manejar todos los pines con los que cuenta el módulo.

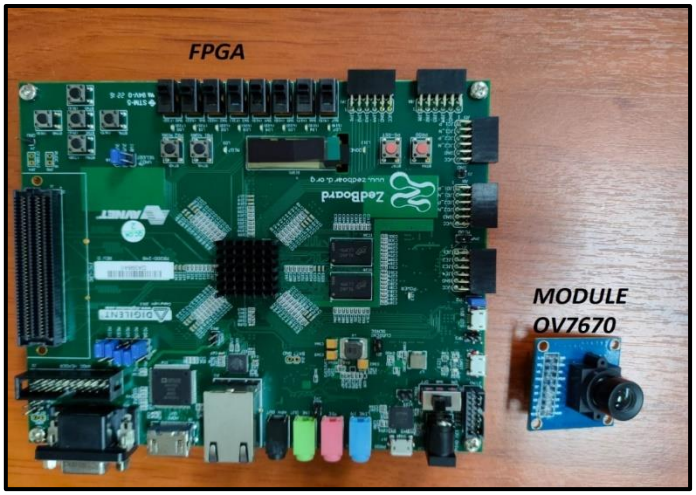

Ilustración 3.42 FPGA y Module OV7670

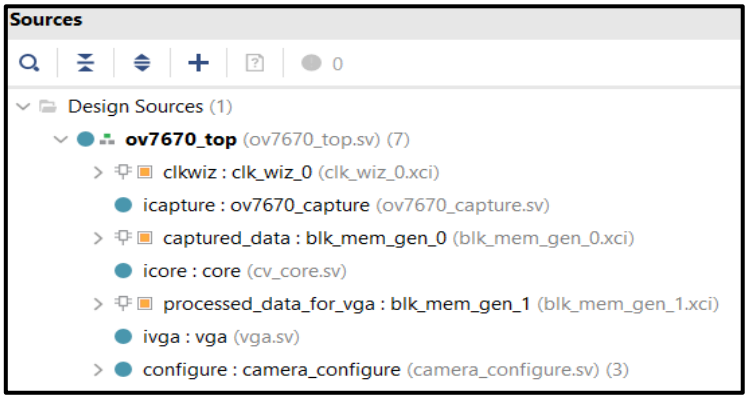

Ilustración 3.43 Archivos .sv y bloques, práctica número 3.

Al momento de compilar los archivos en Verilog se procede a realizar el RTL análisis, lo cual permite desarrollar diagramas de bloques en base a los códigos pre desarrollados, tal como se puede observar en la Ilustración 3.45. Para corroborar que los bloques no presentan errores se puede ver la información de compilación en la Ilustración 3.44.

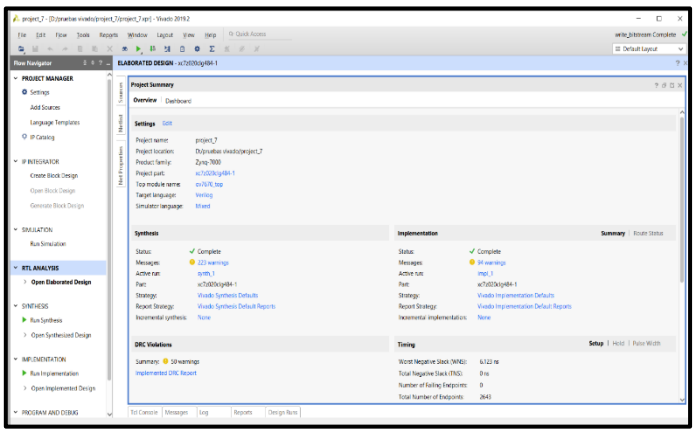

Ilustración 3.44 Overview del RTL análisis

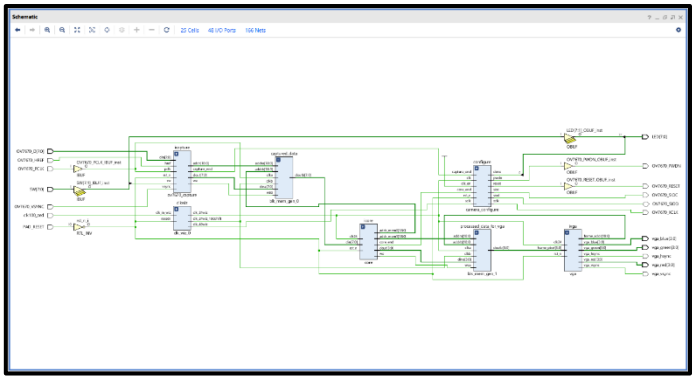

Ilustración 3.45 Diagrama de bloques RTL análisis

Los bloques ov7670_capture y blk_mem_gen_0 van completamente relacionados ya que comparten salidas como entradas, las cuales luego de un proceso de programación mediante bloques permiten crear una salida dout [7:0] de 8 bits la cual es la entrada del bloque Core, también se debe tener en cuenta las entradas digitales OV7670_D[7:O]-OV7670_HREF_OV7670_PCLK,las cuales son provenientes del módulo de cámara esta etapa es denominada captura de imagen y se puede visualizar en la Ilustración 3.46.

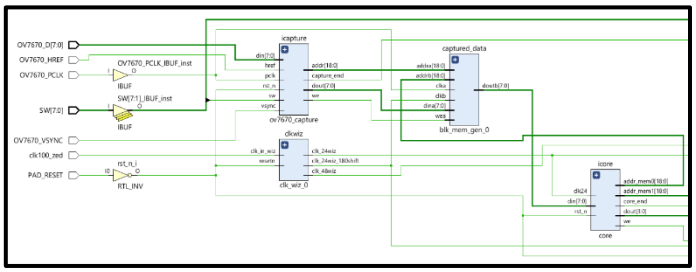

Ilustración 3.46 Etapa de captura de imagen

Las señales de salida del bloque Core van hacer procesadas directamente por el proessed_data_for_vga con su respectiva configuración por medio de bloque ip, otorgando una salida dout[3:0] de 4 bits, en el bloque VGA se obtiene salidas de color blue-green-red a la par de hsync y vsync así como las salidas correspondientes al module0V7670 las cuales son el PWDN-RESETSIOC-SIOD-XCLK; el bloque de configuración de cámara permitirá controlar los leds que pertenecen a la FPGA toda esta información se puede observar en la Ilustración 3.47.

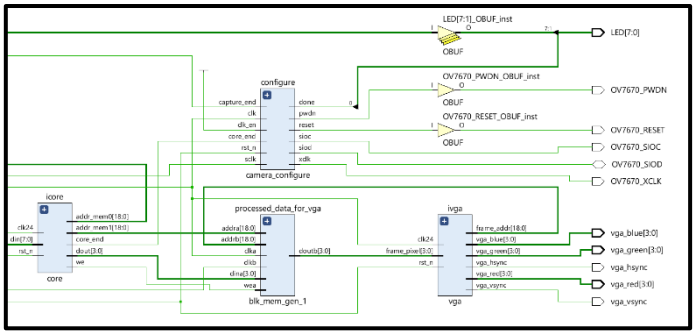

Ilustración 3.47 Etapa de video VGA

Se selecciona en el lado izquierdo la opción Generate Bitstream-Open Hardware Manager-Open Target el cual genera un archivo de compilación para cargar a la FPGA.

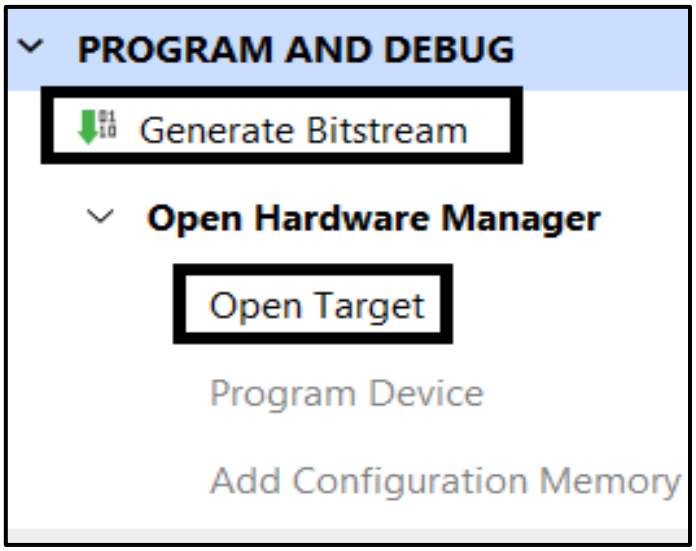

Ilustración 3.48 Generacion de Bitstream

Después de generar el Bitstream, se despliega la pestaña hardware y una etiqueta verde de Open target, como se puede apreciar en la Ilustración 3.49, localhost significa la dirección del dispositivo FPGA.

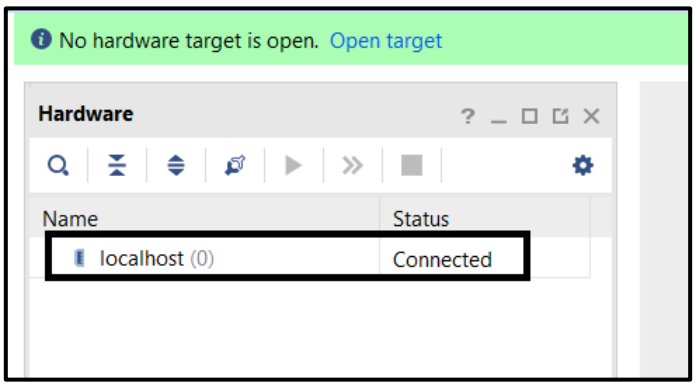

Ilustración 3.49 Open target

Al momento de encender la FPGA, se selecciona la opción Program device, donde se detecta el dispositivo esto se aprecia en la Ilustración 3.50.

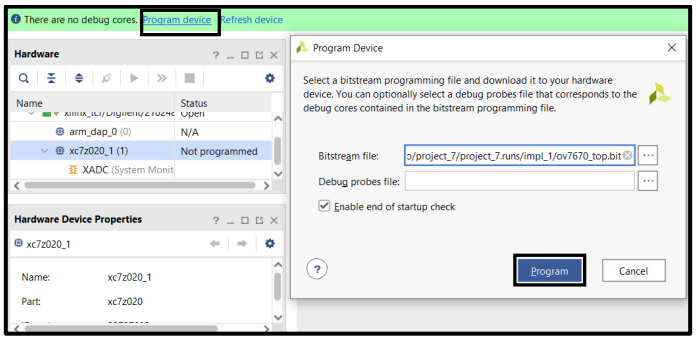

Ilustración 3.50 Programar dispositivo

En la pestaña Hardware de la Ilustración 3.51, se puede observar un mensaje que indica que la FPGA se encuentra programada.

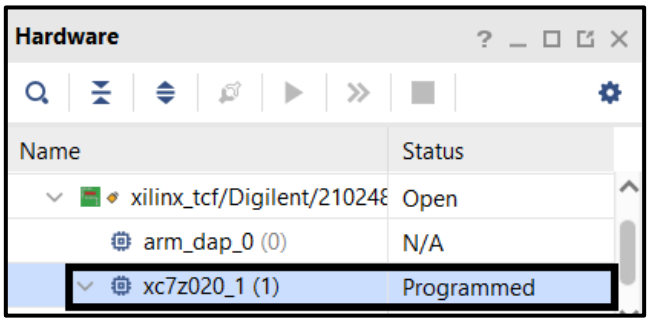

Ilustración 3.51 Dispositivo Programado.

3.2.3.3 Conexión FPGA - modulo 0V7670

En la Tabla 3.7 se visualizan los pines a conectar entre los Pmod JB-JC-JD de la FPGA y los pines del módulo OV7670. Mientras que las conexiones físicas realizadas se observan en la Ilustración 3.52.

MODULE OV7670	FPGA	MODULE OV7670	FPGA
3.3V	JB6	D7	JC3P
GND	JB5	D6	JC1P
SCL	JD4P	D5	JB10
SDA	JD2P	D4	JB4
VS	JD3P	D3	JB9
HS	JD1P	D2	JB3
PCLK	JC4P	D1	JB8
MCLK	JC2P	D0	JB2
D7	JC3P	RESET	JB7
D6	JC1P	PWDN	JB1

Tabla 3.7 Conexión FPGA-Module OV7670

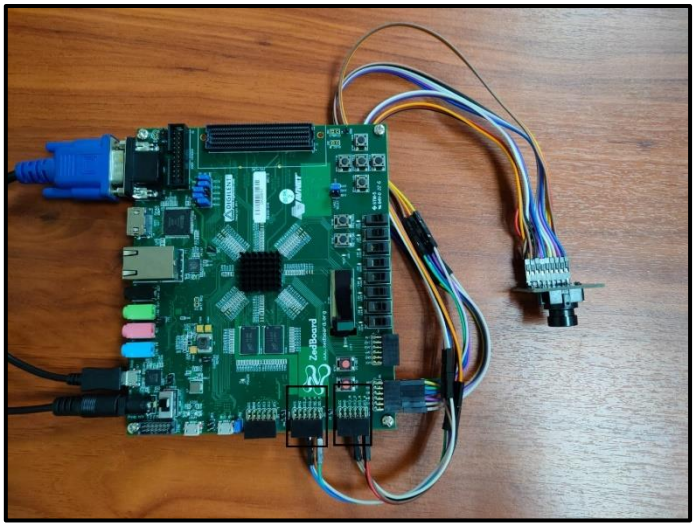

Ilustración 3.52 Conexión FPGA-Module OV7670

CAPITULO 4

4 ANALISIS DE RESULTADOS

En este capítulo se procederá al análisis de los resultados obtenidos en cada una de las prácticas realizadas, visualizando también el producto obtenido. A esto se añadirá la acogida obtenida por los estudiantes de la materia de Diseño de Aplicaciones en Telecomunicaciones del primer término 2022, provista mediante una encuesta.

4.1 Resultados de la práctica 1

La práctica de laboratorio 1 la cual consiste en integrar un módulo Arduino junto a una FPGA por medio del protocolo de comunicación I2C, se desarrolló aplicando los conceptos básicos de la comunicación especifica la cual es generar un dispositivo esclavo y un dispositivo maestro como se observa en la Ilustración 4.1.En donde la FPGA actúa como esclavo y el Arduino como maestro

Ilustración 4.1 Dispositivo eslavo y maestro

Al momento de compilar el código VHDL se crean los bloques I2c_Slave y data_to_master como se observa en la Ilustración 4.2, esto corrobora la correcta implementación de los códigos preestablecidos. Por otro lado, en la Ilustración 4.3 se observa la asignación de pines obtenida.

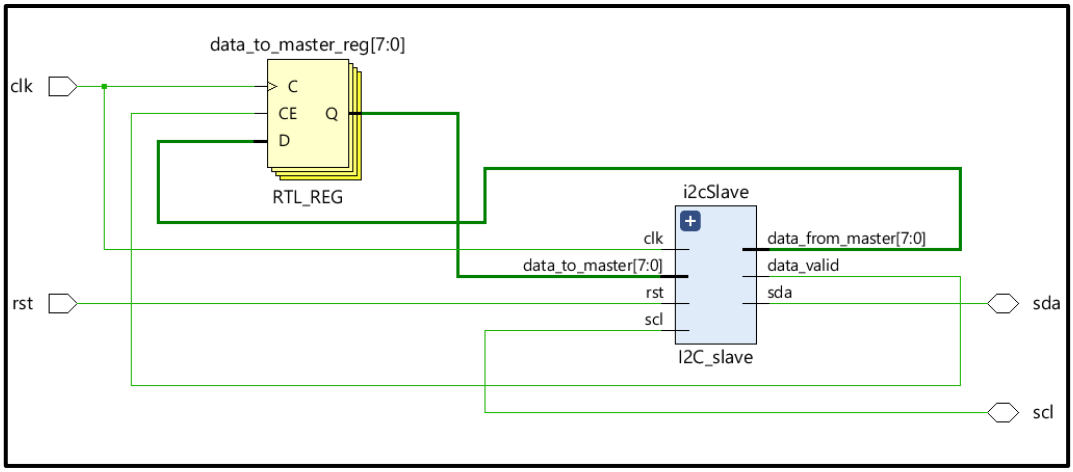

Ilustración 4.2 Generación bloques protocolo I2C

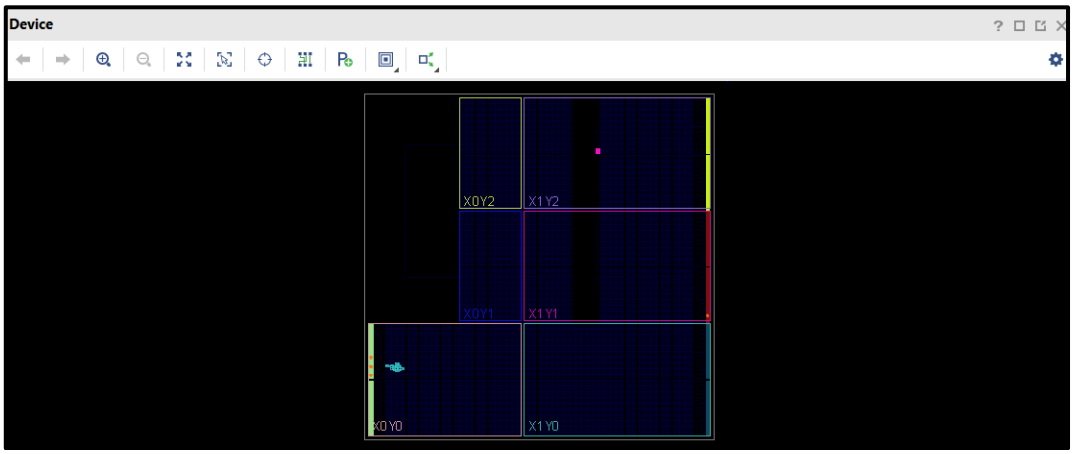

Ilustración 4.3 Generación del Bitstream para asignación de pines protocolo I2C

En la Ilustración 4.4 se observa el valor ingresado y que será enviado hacia la FPGA. A continuación, se muestra el valor recibido de la FPGA que en este caso le añadió 5 a cada valor enviado, obteniendo 11, 13 y 8 para las entradas 6, 8 y 3 correspondientemente.

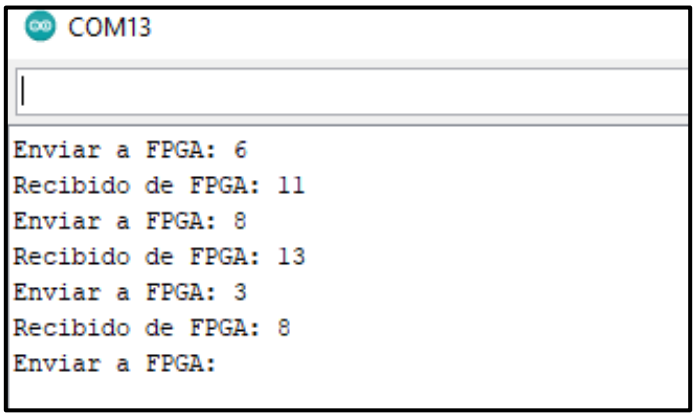

Ilustración 4.4 Resultado de suma en el protocolo I2C

En la Ilustración 4.5 se observan nuevamente los datos enviados y recibidos, con la diferencia de que ahora la FPGA realiza el proceso de sumar 2 a cada valor que recibe. Se envía 1, 4 y 6 obteniendo de la FPGA los valores 3, 6 y 8.

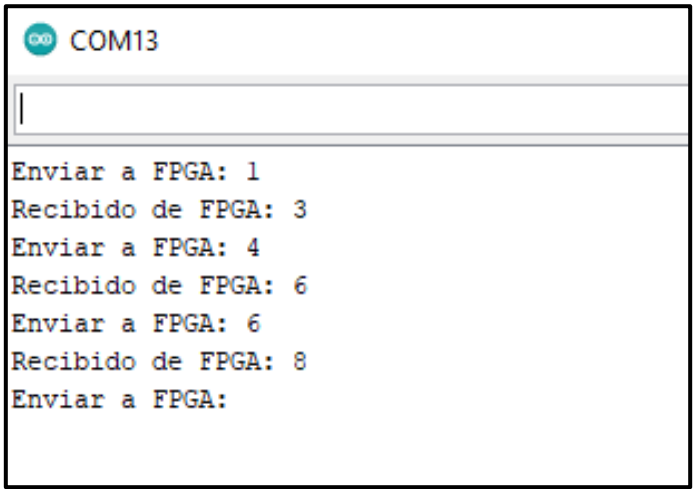

Ilustración 4.5 Resultado de suma en el Protocolo I2C

4.2 Resultados de la práctica 2

Para esta práctica el primer resultado que se obtuvo fue el correcto funcionamiento de los sensores diseñados. Así, en la Ilustración 4.6, se observa el funcionamiento del sensor de movimiento variando el voltaje en el emisor del fototransistor según se coloca o no un obstáculo entre el par emisor-receptor.

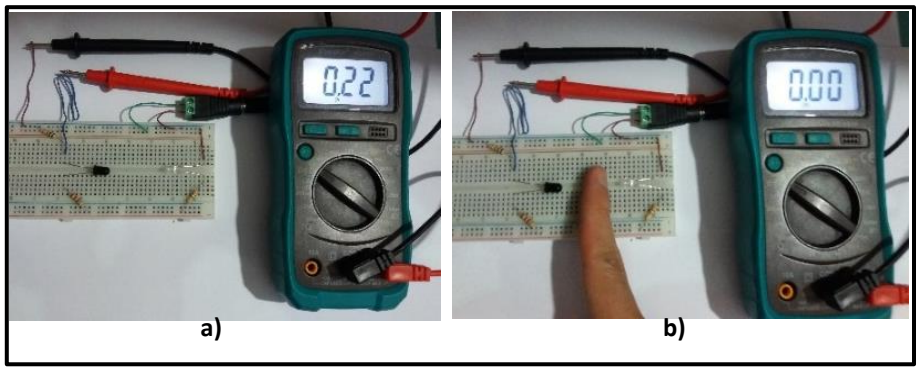

Ilustración 4.6 Voltaje en el emisor del fototransistor
a) sin obstáculo, b) con obstáculo.

De igual forma se comprobó el funcionamiento del sensor de temperatura. En la Ilustración 4.7 se observa el comportamiento del sensor tanto a temperatura ambiente, como en presencia de una fuente de calor que en este caso es un cautín.

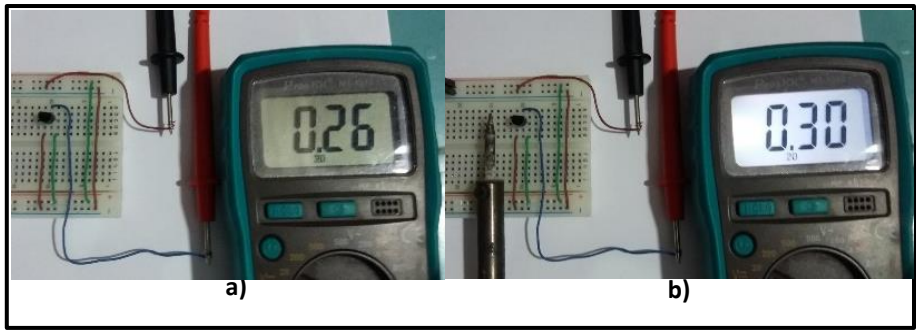

Ilustración 4.7 Comportamiento del sensor de temperatura
a) temperatura ambiente, b) temperatura en presencia de cautín.

Otro de los resultados fue el correcto funcionamiento de la FPGA, específicamente del conversor ADC y del procesador. De esta forma mediante un monitor serial en el software Vitis se pueden visualizar los valores medidos por los sensores en tiempo real, tal como se muestra en la Ilustración 4.8. El primer dato representa la señal del sensor de movimiento con niveles de voltaje por sobre los 100mV en presencia de luz, mientras que al colocar un obstáculo este valor cae hasta los 0mV. El segundo valor es la señal del sensor de temperatura.

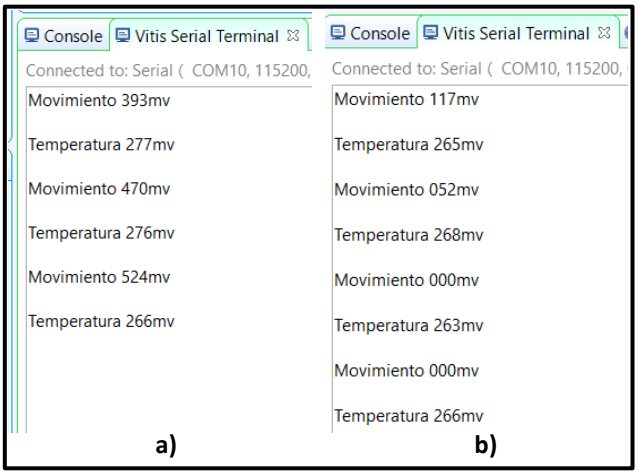

Ilustración 4.8 Terminal serial de Vitis a) sin obstáculos, b) con obstáculo.

A continuación, se corroboró el funcionamiento de la transmisión de los datos hacia los módulos Xbee mediante el protocolo UART y la comunicación inalámbrica entre ellos. Para visualizar los valores recibidos en el módulo conectado a la PC, se utilizó el monitor serial del software XCTU donde dicha información se presenta en color rojo. En la Ilustración 4.9 se observa de forma constante la recepción de los valores censados.

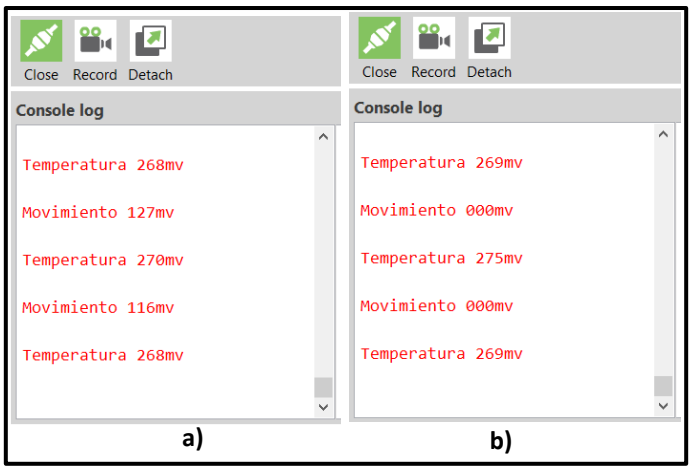

Ilustración 4.9 Monitor serial de XCTU; a) sin obstáculo, b) con obstáculo.

Finalmente, la última sección del sistema de la cual se obtuvieron resultados fue de la interfaz realizada en LabVIEW. Mediante dos indicadores, uno tipo metro para la temperatura y un led para el movimiento, se visualizan las señales censadas. En este caso no se presenta el valor del sensor de movimiento, más bien dicho voltaje al caer debajo de cierto nivel procederá a encender el led. En la Ilustración 4.10 se observa cómo se marca la temperatura del lugar, mientras el led se enciende o se apaga según se presentan obstáculos o no.

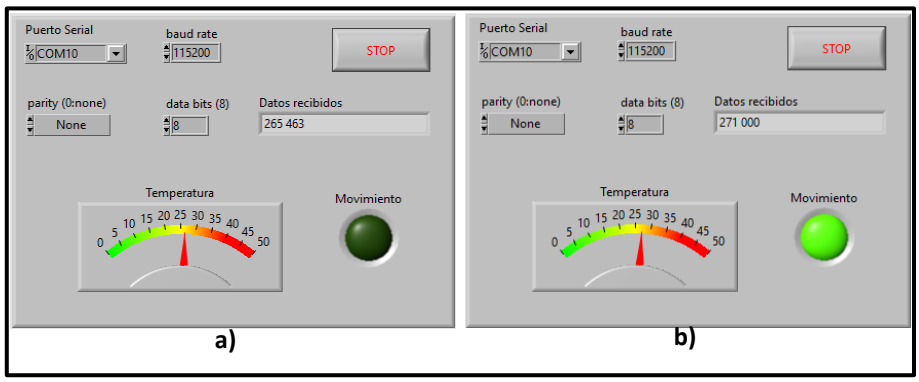

Ilustración 4.10 Interfaz en LabVIEW a) sin obstáculo, b) con obstáculo.

De esta forma se evidenció el correcto funcionamiento de cada una de las partes del sistema implementado, tanto en la generación de las señales como en el procesamiento, transmisión y visualización de estas.

4.3 Resultados de la práctica 3

La creación de bloques mediante RTL analysis que se visualiza en la Ilustración 4.11, es la implementación correcta del código con sus respectivas entradas y salidas de la FPGA. Cuando se genera dicho esquemático quiere decir que el código programado en Verilog y la unión de bloques son correctos y están trabajando paralelamente.

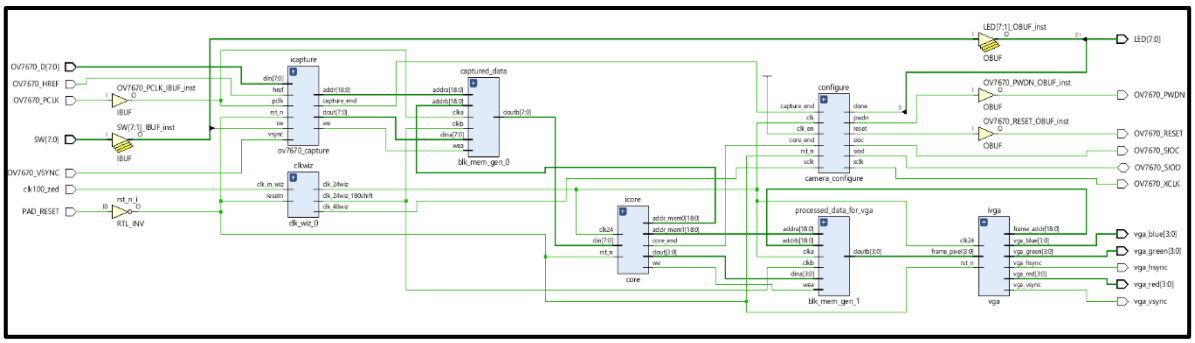

Ilustración 4.11 RTL analysis diagrama de bloques

El bitstream del dispositivo (FPGA), para que se genere este elemento se tuvo que compilar el run simulation y el RTL analysis de manera correcta para posterior asignar los pines IN Y OUT de la tarjeta como se observa en la Ilustración 4.12, este archivo es el que se programara en el dispositivo.

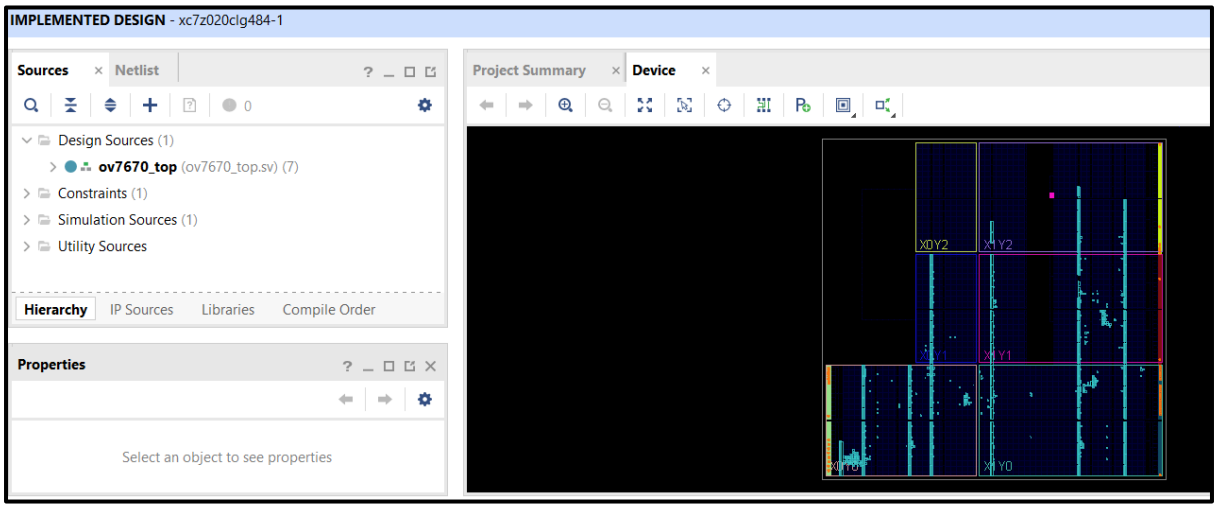

Ilustración 4.12 Bitstream Dispositivo

Cuando el led de la FPGA toma un color azul, significa que la información se está subiendo al dispositivo como se observa en la Ilustración 4.13.

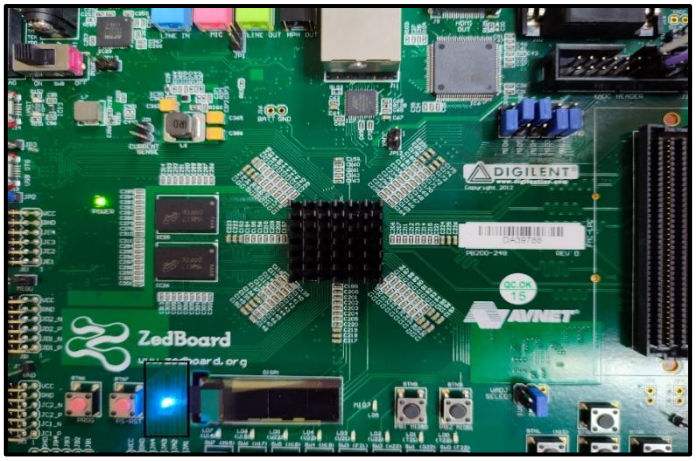

Ilustración 4.13 FPGA programada

En la Ilustración 4.14 - Ilustración 4.15 se puede observar, el funcionamiento de la cámara el cual se encuentra representado en una pantalla con entrada VGA, ya que la FPGA posee una entrada con esa característica para poder visualizar el resultado de video.

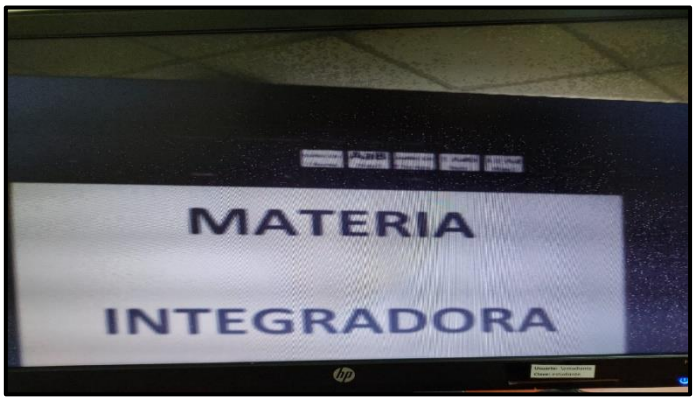

Ilustración 4.14 Imagen del Module OV7670

Ilustración 4.15 Imagen del Module OV7670

4.4 Resultados de las encuestas

Una vez obtenidos los resultados de las tres prácticas implementadas, se procedió a presentar dos de ellas a los estudiantes que se encuentran actualmente cursando la materia de Diseño de Aplicaciones en Telecomunicaciones. La finalidad de esta presentación es observar la acogida de los estudiantes hacia las prácticas diseñadas.

Las prácticas que se decidieron mostrar son la numero uno: SISTEMA DE SENSORES CON COMUNICACIÓN INALÁMBRICA, y el número dos: SISTEMA DE CÁMARA - MÓDULO OV7670. Se expuso el objetivo de la práctica, los materiales a utilizar y una pequeña descripción de las actividades a realizar. Al finalizar se realizó una encuesta donde los estudiantes podrían puntuar cada práctica; a continuación, se muestran los resultados de dicha encuesta.

En la Ilustración 4.16 se tiene la primera pregunta donde se evidencia que, de los estudiantes encuestados solo 6 (42.9%) de ellos han trabajado antes con una FPGA. A pesar de encontrarse en los últimos niveles de la carrera más de la mitad aún no a este tipo de equipos.

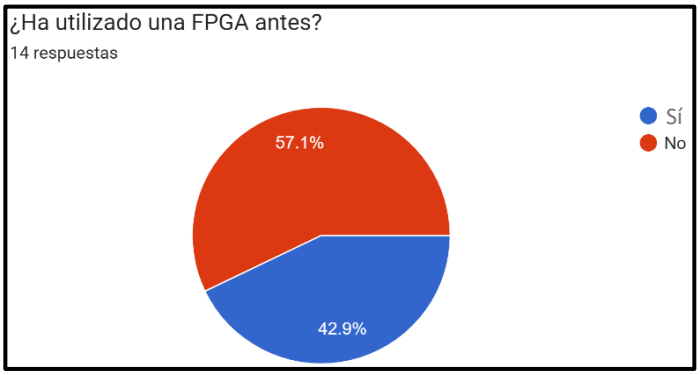

Ilustración 4.16 Resultados pregunta 1.

Para los que respondieron un "sí" a la primera pregunta, en la Ilustración 4.17 se observa que 4 (66.7%) han utilizado la marca Intel y solo 2 (33.3%) está familiarizado con la marca Xilinx. Se evidencia que de los encuestados solo el 14.3% ha utilizado esta marca de placas.

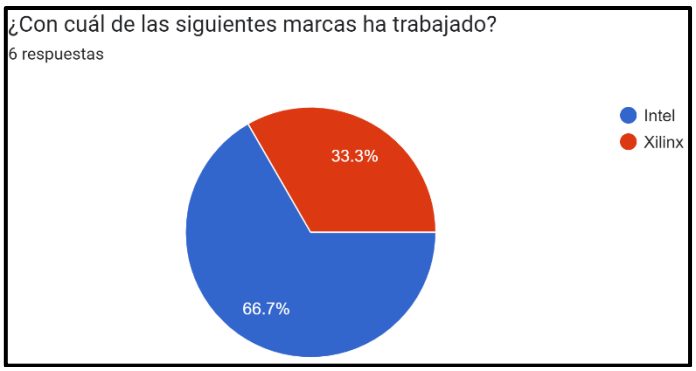

Ilustración 4.17 Resultados pregunta 2.

Para la tercera pregunta se decidió conocer si los estudiantes sabían que se podía incorporar componentes como el un Arduino, módulos Xbee u OV7670 junto con una FPGA. La mitad de ellos ignoraba que era posible integrar todos estos componentes, tal como lo indica la Ilustración 4.18.

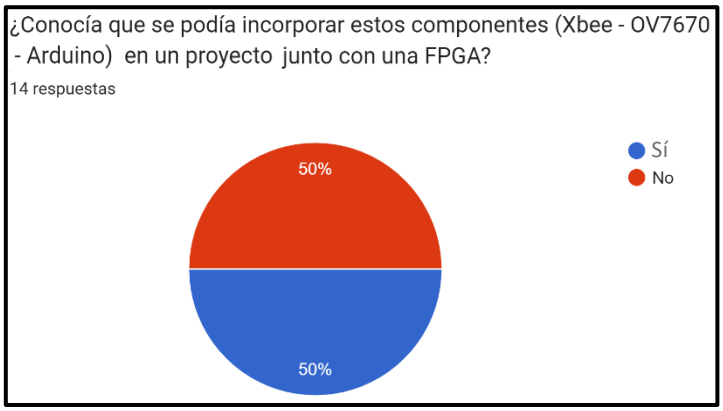

Ilustración 4.18 Resultados pregunta 3.

En las preguntas 4 y 5 se buscaba conocer si los estudiantes creían que les sería de ayuda saber utilizar una FPGA y si les interesa aprender a programarlas. En la Ilustración 4.19 y en la Ilustración 4.20 se muestra que 13 (92.9%) de ellos consideran que les será útil desarrollar habilidades sobre el uso y programación de estos equipos.

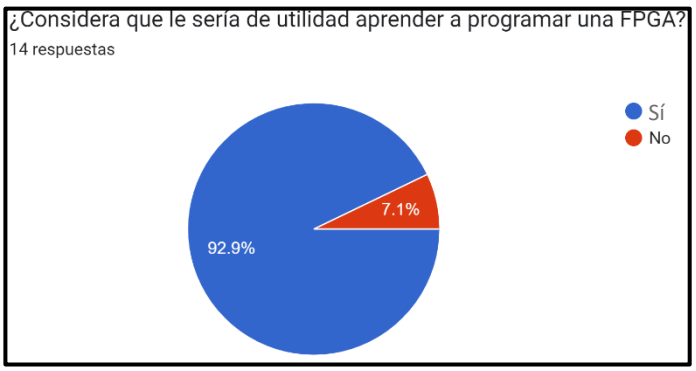

Ilustración 4.19 Resultados pregunta 4.

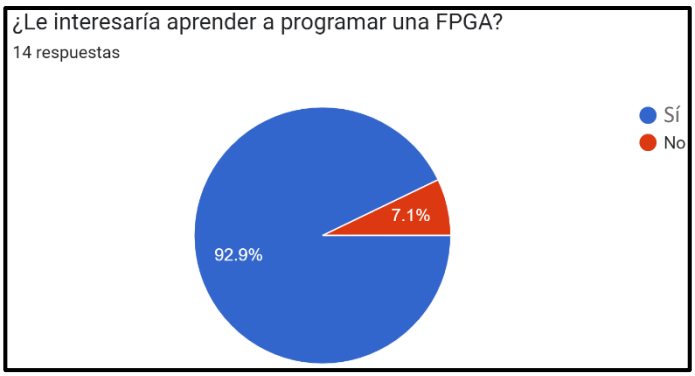

Ilustración 4.20 Resultados pregunta 5.

Finalmente se pidió a los estudiantes que puntúen las secciones presentadas de cada práctica, es decir los componentes utilizados, actividades a realizar, resultados y la práctica en general. En la todas las secciones el 92.8% de los encuestados calificó las prácticas con puntuaciones de 4 y 5, tal como se observa en la Ilustración 4.21 e Ilustración 4.22. En ambas preguntas se evidencia que como mínimo 8 estudiantes califican con 5 las prácticas presentadas.

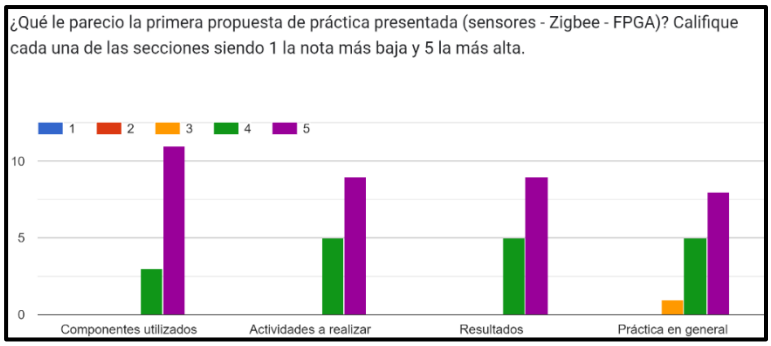

Ilustración 4.21 Resultados pregunta 6.

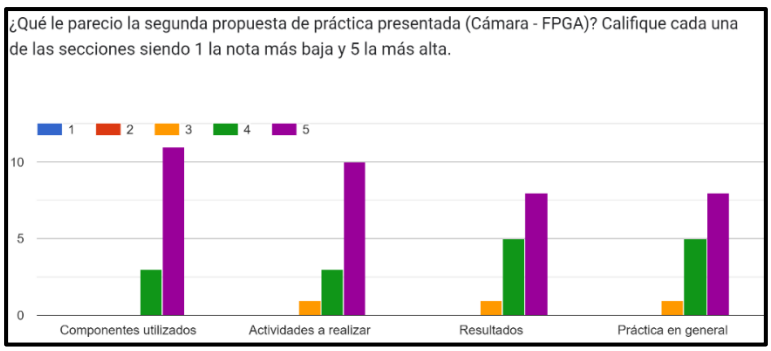

Ilustración 4.22 Resultados pregunta 7.

Habiendo conocido la acogida que tienen las prácticas, se procedió a realizar las correspondientes guías de laboratorio donde se establecerán los: objetivos, materiales a utilizar, actividades a realizar y los resultados esperados. Estas guías se encuentran dentro de la sección de Anexos.

CONCLUSIONES

Se evidenció la capacidad de ejecución de la FPGA, al poder realizar múltiples tareas de forma paralela; además de una alta versatilidad al permitir diseñar proyectos tanto en VHDL, Verilog así como incorporar bloques ya predefinidos en el software.

La implementación de las tres prácticas permitió trabajar en diferentes funciones de la FPGA, desarrollando proyectos completamente en VHDL o trabajando directamente con bloques predefinidos en el procesador. De esta forma los estudiantes evidenciaron la versatilidad del equipo y aprendieron las diferentes formas de llevar a cabo un proyecto.

En la práctica 1 se logró implementar el protocolo I2C mediante codificación en VHDL, permitiendo así que un Arduino envíe datos hacia la FPGA y que esta realice el correspondiente procesamiento. Así también se consiguió que la FPGA envié hacia el microprocesador los resultados, observándolos en el monitor serial de Arduino.

Con la implementación de la segunda práctica se obtuvo un sistema capaz de detectar señales, procesarlas y transmitirlas de forma inalámbrica. La electrónica utilizada para los sensores permitió adquirir las señales deseadas de temperatura y movimiento, así también la FPGA realizó la correspondiente conversión de los valores y su transmisión hacia los módulos de comunicación inalámbrica. Gracias a la configuración de los módulos Zigbee se consiguió establecer el enlace con la PC, en donde la interfaz realizada facilitó la visualización de los valores obtenidos.

En la práctica 3 se obtuvo el envío de información a tiempo real mediante la codificación de bloques en Verilog, logrando el correcto funcionamiento del módulo y la transmisión de los datos captados hacia una pantalla mediante la salida VGA. Los puertos Pmod (JB1-JC1-JD1) fueron ampliamente utilizados, tanto para las señales propias de las imágenes como para el protocolo de comunicación.

La presentación de las prácticas diseñadas con los estudiantes, así como la encuesta realizada permitió medir el conocimiento que ellos poseen acerca de la tecnología FPGA. Se observó que la mitad del curso desconocía la posibilidad de realizar una integración como la mostrada, así también se evidenció una buena acogida de las propuestas. Las respuestas obtenidas indican que la mayoría de ellos consideran que les será de utilidad tener conocimientos acerca de la programación de una FPGA, así como su integración con otros dispositivos.

RECOMENDACIONES

El conversor ADC de la FPGA solo trabaja con valores menores a 1 voltio, caso contrario puede ocasionar daños colaterales al equipo. Se recomienda realizar divisores de voltajes a las señales que se vayan a utilizar, permitiendo así el uso adecuado del dispositivo e incrementando su vida útil.

Se recomienda revisar el datasheet de los dispositivos a utilizar, ya que no todos trabajan con la misma energía, cómo fue el caso del puerto UART 1 de la FPGA el cual se alimenta con 3.3V y el módulo adaptador XBEE que se alimenta con 5V. Debido a esta diferencia de energía no se pudo conectar directamente los dispositivos, optando por conectar el adaptador a una salida de 5V provista por la FPGA y habilitar el puerto UART 0 en la FPGA.

Revisar con cautela la información proporcionada por el desarrollador y revisar la misma en el software ya que pueda tener información mal ingresada, como fue el caso de la practica número 3 en la cual Tx y Rx, se encontraba al revés en el puerto FIFO del software VIVADO.

En el caso del uso de Arduino uno tener precaución ya que el dispositivo presenta una salida TX y RX, pero para el desarrollo del protocolo I2C no se necesita estos pines ya que se requiere de SDA con el pin A4 Y SCL con el pin A5, solo 2 líneas las cuales irán conectadas a la FPGA.

Al momento de asignar pines para entradas o salidas tener precaución, ya que la FPGA presenta una función específica para ciertos pines o puertos. Este es el caso del bloque ADC el cual solo acepta señales analógicas y las convierte en digitales, en cambio Pmod JA1-JB1 acepta señales digitales y de energización en cambio los Pmod JC1-JD1 son para crear interfaz para los protocolos

Las versiones más recientes de Vivado y Vitis utilizan una gran cantidad de almacenamiento y de memoria RAM, por ello no se recomienda descargarlas en computadoras antiguas o de bajos recursos. Se aconseja utilizar la versión 2019.2 ya que representa una menor carga para el equipo.

BIBLIOGRAFÍA

[1] Ethernity Networks, «Ethernity Networks,» 18 11 2021. [En línea]. Available: https://ethernitynet.com/advantage-using-fpgas-telecom-edge-networking/#:~:text=FPGAs%20in%20Networking%20Applications&text=The%20parallel%20processing%20trait%20of,be%20reprogrammed%20in%20the%20field. [Último acceso: 31 05 2022].

[2] F. Pineda, «Logica Programable,» 30 10 2015. [En línea]. Available: https://sites.google.com/site/logicaprogramable/vhdl/lenguaje-vhdl. [Último acceso: 15 06 2022].

[3] Ministerio de Salud Pública, 25 05 2022. [En línea]. Available: https://www.salud.gob.ec/por-varias-semanas-ecuador-presenta-indicadores-estables-de-covid-19/#. [Último acceso: 2022 05 26].

[4] Ministerio de Salud Pública, 15 03 2022. [En línea]. Available: https://www.salud.gob.ec/ecuador-es-referente-en-el-control-de-la-pandemia-de-la-covid-19/. [Último acceso: 26 05 2022].

[5] Ministerio de Educación, 21 02 2022. [En línea]. Available: https://educacion.gob.ec/el-proceso-de-retorno-a-la-educacion-presencial-en-todo-el-pais-esta-previsto-para-mayo-de-2022/#. [Último acceso: 26 05 2022].

[6] ESPOL, 04 04 2022. [En línea]. Available: https://www.espol.edu.ec/noticias/espol-capacita-profesores-en-el-dictado-de-clases-en-modalidad-hibrida#:~:text=Como%20parte%20del%20retorno%20progresivo,E%2Dlearning%20y%20Virtual%20emergente. [Último acceso: 26 05 2022].

[7] FIEC, ESPOL, 05 09 2021. [En línea]. Available: https://www.fiec.espol.edu.ec/sites/fiec.espol.edu.ec/files/PROTOCOLO%20RETORNO%20PROGRESIVO%20FINAL%20ACADEMICO%20Y%20ADMINISTRATIVO%20APROBADO.pdf. [Último acceso: 26 05 2022].

[8] República del Ecuador, «VII Régimen del buen vivir,» *Constitución de la República del Ecuador,* pp. 169, 186.

[9] A. M. M. P. A. M. Joaquín Olivares, «Laboratorio virtual para la programación de FPGAs,» Leiria, Portugal, 2005.

[10] A. B. A. Steven, «Herramienta FPGA para el diseño en bloques y programación VHDL en la asignatura de Sistemas Digitales I,» Universidad Católica de Santiago de Guayaquil, Guayaquil, 2012.

[11] B. M. L. P. I. M. Wilfredo Falcón Urquiaga, «PROTOTIPADO RÁPIDO PARA APLICACIONES SOBRE FPGAs DE XILINX,» *ResearchGate,* vol. 1, nº 1, p. 6, 2010.

[12] DIEECTQAI(Laboratorios remotos), «Ieectqai,» 13 12 2016. [En línea]. Available: http://www.ieectqai.uned.es/labs_remotos.html. [Último acceso: 31 05 2022].

[13] E. Boemo Scalvinoni, «ESTADO DEL ARTE DE LA TECNOLOGÍA FPGA,» Buenos Aires, 2005.

[14] E. Cruz Miguel, J. Rodríguez Reséndiz, J. García Martínez, K. Camarillo Gómez y G. Pérez Soto, «Field-programmable gate array-based laboratory orientedto control theory courses,» *Computer Applications in Engineering Education,* vol. 27, nº 5, p. 1253–1266, 07 2019.

[15] M. Abdulwahed y W. Balid, «A novel FPGA educational paradigm using the next generation programming languages case of an embedded FPGA system course,» de *2013 IEEE Global Engineering Education Conference (EDUCON)*, Berlin, 2013.

[16] F. Shakith, A. Kumar y R. C. Panicer, «Project-Based Learning in Embedded Systems Education Using an FPGA Platform,» *IEEE Transactions on Education,* vol. 56, nº 4, pp. 407 - 415, 11 2013.

[17] G. E. Blanco Silva, J. Rodriguez Resendiz, E. Gorrostieta Hurtado, J. C. Pedraza Ortega y J. M. Ramos Arreguin, «Didactic Platform for Image Processing,» *IEEE LATIN AMERICA TRANSACTIONS,* vol. 13, nº 10, pp. 3398 - 3404, 10 2015.

[18] Q. Shi, L. Xiang, T. Chen y W. Hu, «FPGA-Based Embedded System Education,» de *2009 First International Workshop on Education Technology and Computer Science*, Wuhan, 2009.

[19] S. A. Edwards, «Experiences Teaching an FPGA-based,» de *In Proceedings of the Workshop on Embedded Systems Education (WESE)*, New York, 2005.

[20] J. O. Hamblen y T. S. Hall, «System-on-a-programmable-chip development platforms in the classroom,» *IEEE Transactions on Education,* vol. 47, nº 4, pp. 502 - 507, 11 2004.

[21] Yong-Kyu Jung, «Work in Progress - A Rapid Design Methodology for FPGA-based Processor Platform Design Education,» de *Proceedings Frontiers in Education 35th Annual Conference*, Indianapolis, 2005.

[22] L. Sousa, S. Antao y J. Germano, «A Lab Project on the Design and Implementation of Programmable and Configurable Embedded Systems,» *IEEE Transactions on Education,* vol. 56, nº 3, pp. 322 - 328, 08 2013.

[23] R. K. M. GORGON´, «FPGA IMPLEMENTATION OF DECISION TREES AND TREE ENSEMBLES FOR CHARACTER RECOGNITION IN VIVADO HLS,» *Image Processing & Communication,* vol. 19, nº 2-3, p. 12, 2015.

[24] Ó. O.-P. J. C.-R. J. A. S.-D. R. N.-G. A. D.-M. J. A. Botín-Córdoba, «Laboratorio Remoto de Electrónica Digital,» Universidad de Malaga, Malaga, 2020.

[25] E. M. C. M. R. V. D. H. E. B. R. C. M. B. D. O. S. A. L. Jiménez, «Prototipo de cámara IMAX+ como recurso docente para calibración y tratamiento de imágenes en FPGA,» *XIV Congreso de Tecnologías Aplicadas a la Enseñanza de la Electrónica,* p. 7, 2020.

[26] Digi-Key América del Norte, «Digi Key,» Diji Key, 14 02 2019. [En línea]. Available: https://www.digikey.com/es/articles/why-how-to-use-serial-peripheral-interface-simplify-connections-between-multiple-devices#:~:text=El%20SPI%20permite%20la%20comunicaci%C3%B3n,permite%20la%20comunicaci%C3%B3n%20entre%20microcontroladores.. [Último acceso: 01 06 2022].

[27] A. Balean, «Aplicación de la placa FPGA IceZUM Alhambra en prácticas de laboratorio de automatización de sistemas,» Universidad Jaume I, Castellom, 2018.

[28] D. J. Castillo Castillo y R. E. Osorio Salcedo, «ANÁLISIS, DISEÑO E IMPLEMENTACIÓN DE UN PROTOTIPO PARA LA,» Universidad Politécnica Salesiana, Quito, 2011.

[29] «FPGA 4 Student,» [En línea]. Available: https://www.fpga4student.com/2018/08/basys-3-fpga-ov7670-camera.html. [Último acceso: 26 05 2022].

[30] National Instruments, «ni.com,» 08 2018. [En línea]. Available: https://www.ni.com/es-cr/innovations/white-papers/08/fpga-fundamentals.html. [Último acceso: 07 2022].

[31] C. Sisterna, «Departamento de Electrónica y Automática,» 2010. [En línea]. Available: http://dea.unsj.edu.ar/sisdig2/Field%20Programmable%20Gate%20Arrays_A.pdf. [Último acceso: 12 07 2022].

[32] S. Noriega, «Facultad de Ingeniería Universidad Nacional de La Plata,» 2020. [En línea]. Available: https://catedra.ing.unlp.edu.ar/electrotecnia/islyd/apuntes/Tema%2012%20a%20y%20b%20Logica%20Programable%20Dispositivos%202020.pdf. [Último acceso: 07 2022].

[33] B. Gomez, «Profesional review,» 27 10 2020. [En línea]. Available: https://www.profesionalreview.com/2020/10/27/amd-compra-xilinx/. [Último acceso: 12 07 2022].

[34] HETPRO, «Het pro store,» [En línea]. Available: https://hetpro-store.com/altera-fpga-cyclone-cpld/. [Último acceso: 12 07 2022].

[35] M. Sánchez-Élez, «INTRODUCCIÓN A LA PROGRAMACIÓN EN VHDL,» Facultad de Informática Universidad Complutense de Madrid, Madrid, 2014.

[36] M. Sánchez Élez, «Universidad Complutense Madrid,» 11 2014. [En línea]. Available: https://eprints.ucm.es/id/eprint/26200/1/intro_VHDL.pdf. [Último acceso: 07 2022].

[37] P. De Clavijo Vásquez, «dte.us.es,» 2012. [En línea]. Available: https://www.dte.us.es/Members/paulino/Verilog-Intro.pdf. [Último acceso: 07 2022].

[38] Jithin, «interServer,» 04 08 2016. [En línea]. Available: https://www.interserver.net/tips/kb/common-network-protocols-ports/. [Último acceso: 19 07 2022].

[39] R. Concepción, «rjconcepcion,» 08 10 2020. [En línea]. Available: https://www.rjconcepcion.com/podcast/protocolos-de-comunicacion-usados-en-microcontroladores/#:~:text=Protocolo%20de%20comunicaci%C3%B3n%20en%20paralelo&text=La%20comunicaci%C3%B3n%20en%20paralelo%20consiste,para%20la%20referencia%20o%20tierra.. [Último acceso: 19 07 2022].

[40] INCIBE, «incibe-cert,» 08 11 2018. [En línea]. Available: https://www.incibe-cert.es/blog/el-protocolo-serie-entiendelo-y-protegelo. [Último acceso: 19 07 2022].

[41] B. W, Serial and Parallel Communications, 6 ed., Boston: Elsevier, 2015, pp. 91-94.

[42] New York University, «NYU,» 14 07 2019. [En línea]. Available: https://itp.nyu.edu/physcomp/lessons/serial-communication-the-basics/. [Último acceso: 19 07 2022].

[43] A. Pandit, «Circuit Digest,» 29 04 2019. [En línea]. Available: https://circuitdigest.com/tutorial/serial-communication-protocols. [Último acceso: 19 07 2022].

[44] A. Fernández Caparros, «Universidad de Córdoba,» 04 2004. [En línea]. Available: chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/http://www.uco.es/~el1mofer/Docs/IntPerif/Bus%20I2C.pdf. [Último acceso: 19 07 2022].

[45] Mosaic Industries, «mosaic-industries,» 21 05 2011. [En línea]. Available: http://www.mosaic-industries.com/embedded-systems/sbc-single-board-computers/freescale-hcs12-9s12-c-language/instrument-control/i2c-bus-specifications. [Último acceso: 19 07 2022].

[46] NXP Semiconductors, «NXP,» 1 10 2021. [En línea]. Available: chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://www.nxp.com/docs/en/user-guide/UM10204.pdf. [Último acceso: 19 07 2022].

[47] L. Llamas, «Luis Llamas,» 18 05 2016. [En línea]. Available: https://www.luisllamas.es/arduino-i2c/. [Último acceso: 19 07 2022].

[48] SONY, «Sony latin,» 02 05 2022. [En línea]. Available: https://www.sony-latin.com/es/electronics/support/storage-recording-media/articles/00030769. [Último acceso: 19 07 2022].

[49] SONY, «Sony Latin,» 02 05 2022. [En línea]. Available: https://www.sony-latin.com/es/electronics/support/storage-recording-media/articles/00024733. [Último acceso: 19 07 2022].

[50] R. Sole, «Profesional review,» 5 06 2021. [En línea]. Available: https://www.profesionalreview.com/2021/06/05/bluetooth-5-2-vs-bluetooth-5-1/. [Último acceso: 20 07 2022].

[51] VERIZON, «Verizon español,» [En línea]. Available: https://espanol.verizon.com/info/definitions/wifi/#:~:text=Wi%2DFi%20es%20la%20tecnolog%C3%ADa,que%20puedes%20ver%20y%20usar.. [Último acceso: 19 07 2022].

[52] Y. Fernandez, «Xataka Basics,» 18 03 2018. [En línea]. Available: https://www.xataka.com/basics/wifi-2-4g-y-5g-cuales-son-las-diferencias-y-cual-elegir. [Último acceso: 19 07 2022].

[53] Redes inalambricas, «Redes inalambricas,» [En línea]. Available: https://www.redesinalambricas.es/redes-wifi/. [Último acceso: 20 07 2022].

[54] Newark, «Newark Mexico,» [En línea]. Available: https://mexico.newark.com/wireless-lora-technology#:~:text=LoRa%20es%20una%20tecnolog%C3%ADa%20inal%C3%A1mbrica,del%20Internet%20de%20las%20cosas.. [Último acceso: 19 07 2022].

[55] Aprendiendo arduino, «Aprendiendo arduino,» [En línea]. [Último acceso: 20 07 2022].

[56] VENCO, «vencoel,» [En línea]. Available: https://www.vencoel.com/que-es-zigbee-como-funciona-y-caracteristicas-principales/. [Último acceso: 19 07 2022].

[57] A. Rodriguez, «comunicaciones inalambricas hoy,» 22 07 2015. [En línea]. [Último acceso: 20 07 2022].

[58] MCI electronics, «xbee cl,» [En línea]. Available: https://xbee.cl/que-es-xbee/. [Último acceso: 20 07 2022].

[59] T. Siep, I. Gifford, R. Braley y R. Heile, «Paving the way for personal area network standards: an overview of the IEEE P802.15 Working Group for Wireless Personal Area Networks,» *IEEE Personal Communications,* vol. 7, nº 1, pp. 37 - 43, 2000.

[60] W. Stallings, «Redes de transmisión de datos,» de *COMUNICACIONES Y REDES*, Madrid, Pearson, 2004, pp. 14-17.

[61] Eficiencia, [En línea]. Available: https://e-ficiencia.com/domotica-que-es-y-como-funciona/. [Último acceso: 20 07 2022].

[62] S&P, «solerpalau,» 03 04 2017. [En línea]. Available: https://www.solerpalau.com/es-es/blog/sensor-temperatura/#:~:text=Los%20sensores%20temperatura%20son%20dispositivos,agua%20caliente%20sanitaria%20(ACS).. [Último acceso: 20 07 2022].

[63] S&P, «solerpalau,» 15 10 2018. [En línea]. Available: https://www.solerpalau.com/es-es/blog/sensores-movimiento/. [Último acceso: 20 07 2022].

[64] Netatmo, [En línea]. Available: https://www.netatmo.com/es-es/glosario/sensor-de-humo. [Último acceso: 20 07 2022].

[65] Instituto Tecnológico de Querétaro, «itq.edu.mx,» 09 2016. [En línea]. [Último acceso: 07 2022].

ANEXOS

PRÁCTICA # 1

PROTOCOLO DE COMUNICACIÓN I2C

OBJETIVOS:

- Implementar un protocolo de comunicación I2C entre FPGA y Arduino
- Crear el archivo constraints con las respectivas entradas y salidas que requiera el protocolo I2C.
- Realizar las conexiones entre la FPGA y el microcontrolador Arduino para generar la comunicación I2C.

MATERIALES Y HERRAMIENTAS:

Software Vivado 19.2	FPGA Zedboard
Software Arduino	LM386L
Arduino UNO	Fuente 12V
Cable USB – micro USB	Jumpers

BREVE EXPLICACIÓN DE LA PRÁCTICA:

Para la práctica de "Protocolo de comunicación I2C" se trabajó con el diagrama de bloques que se observa en la Ilustración 1. Se compone de dos bloques que funcionan en la FPGA y dos bloques en el Arduino. En la FPGA se realiza la programación del I2C esclavo, y se define un programa para realizar el procesamiento de los datos. En el Arduino se realiza la programación del I2C maestro, el ingreso de los datos y visualización de los resultados luego del procesamiento en la FPGA.

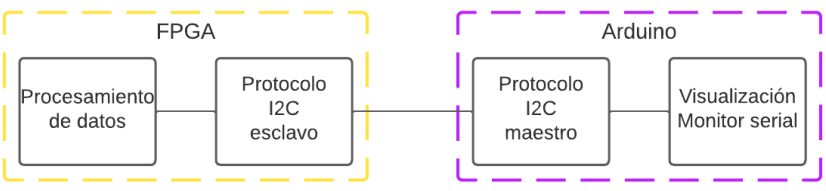

Ilustración 1 Diagrama de bloques general del sistema.

MARCO TEÓRICO:

El protocolo I2C (Inter Integrated Circuit) se compone de dos líneas, SDA y SCL, así como de una referencia. A los dispositivos conectados se los define como maestro o esclavo, permitiendo tener redes de: un maestro y un esclavo, un maestro y varios esclavos, o varios maestros y varios esclavos, tal como se observa en la Ilustración 2.

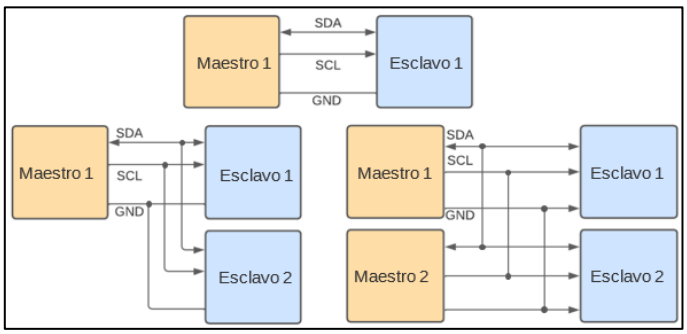

Ilustración 2 Tipos de redes I2C.

Dependiendo de la aplicación necesitada la configuración de maestro-esclavo puede permitir que el maestro trabaje como emisor y el esclavo como receptor, o para que ambos sean emisor y receptor en diferentes tiempos. Cada dispositivo esclavo cuenta con una única dirección, de esta forma el maestro indicará con qué dispositivo desea comunicarse.

En la se observa el proceso de comunicación y los distintos valores que tomarán las señales SDA y SCL según el estado de la transmisión. En la secuencia de comunicación se tiene una condición de inicio donde la línea SDA cambia a bajo estando SCL, 7 bits de dirección, 1 bit que indica escritura o lectura, 8 bits de datos y finalmente una condición de parada que se genera cuando la línea SDA cambia a alto estando SCL en alto.

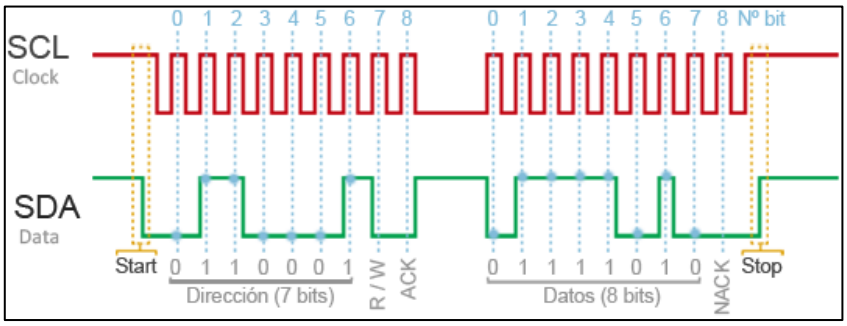

Ilustración 3 Secuencia de comunicación I2C.

INSTRUCCIONES DE LA PRÁCTICA:

1. Descargue la carpeta proporcionada por el docente Protocolo_I2C, de dicha carpeta extraiga los archivos.vhdl, cree una carpeta con el nombre Practica1 en donde almacenara todos los elementos obtenidos anteriormente
2. Proceda abrir el software VIVADO 2019.2 de XILINX, se aplasta la opción Create Project luego se despliega una pestaña donde se selecciona la opción Next, como se observa en la Ilustración 4.

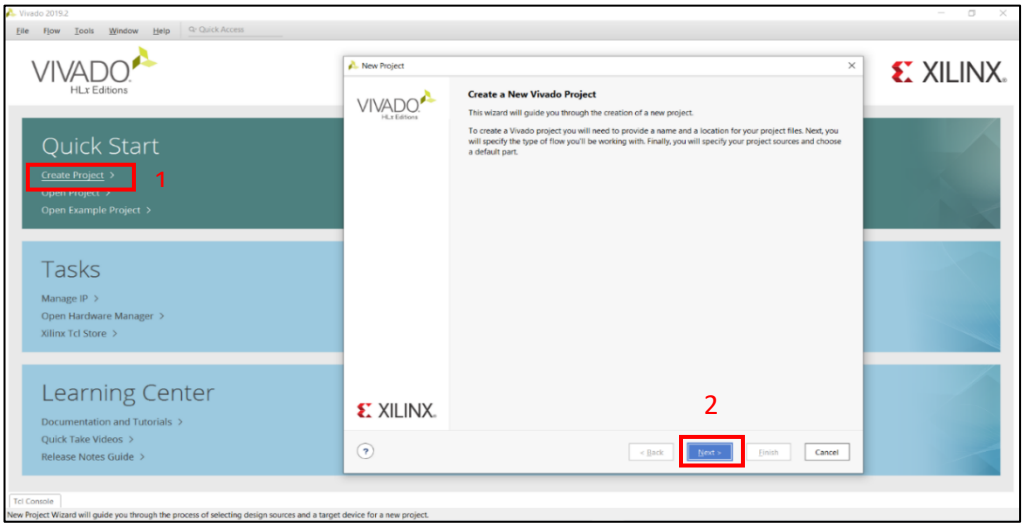

Ilustración 4 Interfaz VIVADO: Create Project

3. Se despliega una pestaña para seleccionar la dirección de la carpeta antes creada, para ubicar el proyecto como se visualiza en la Ilustración 5.

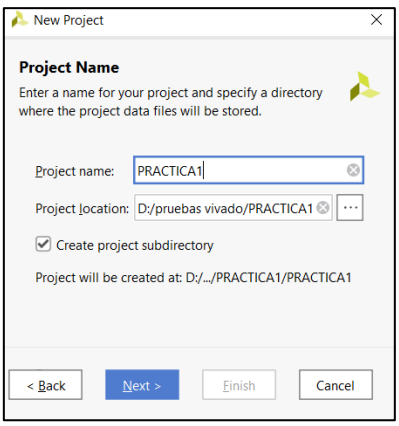

Ilustración 5 Project Name

4. Se procede a escoger la opción RTL Project y luego Next como se visualiza en la Ilustración 6.

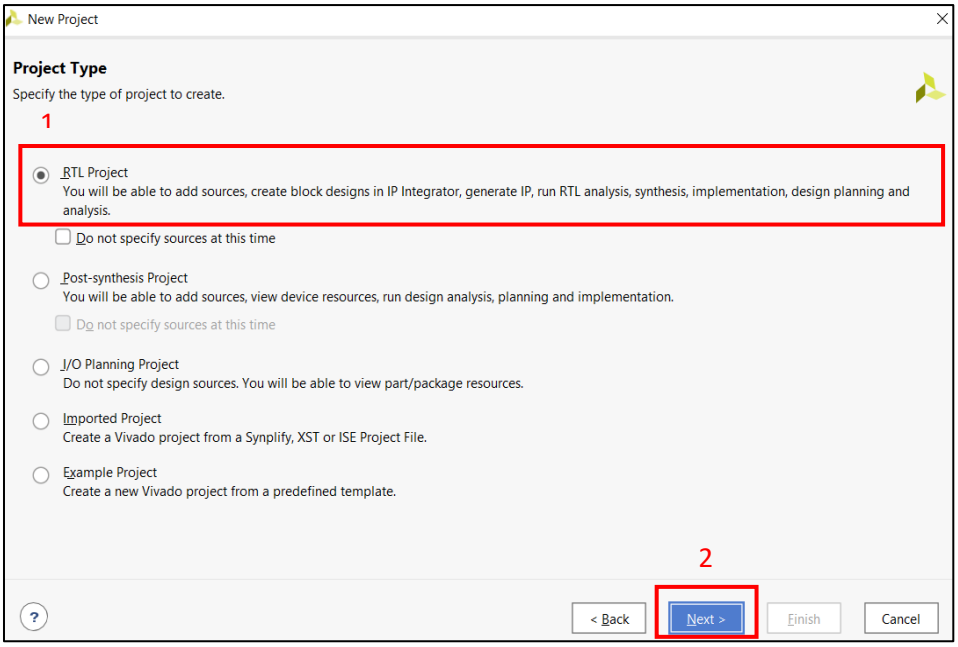

Ilustración 6 Project Type-RTL Project

5. Se despliega la pestaña Add Sources en donde se procede a seleccionar la opción Add Files y posterior Next como se aprecia en la Ilustración 7.

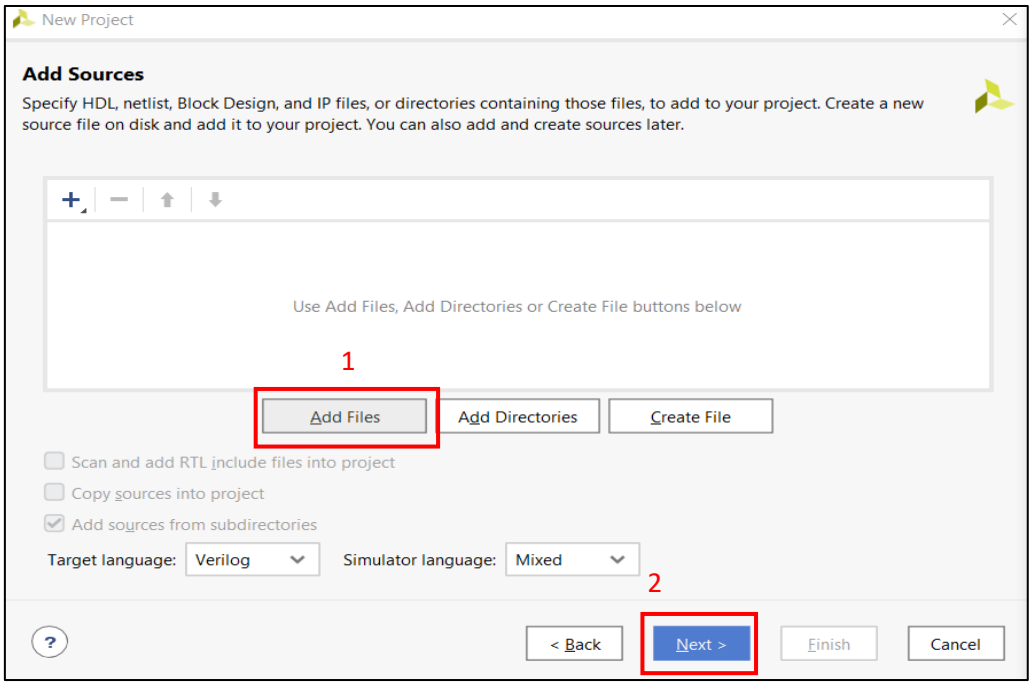

Ilustración 7 Adición de archivos

6. Se procede a cargar los archivos de la carpeta Protocolo_I2C proporcionada por el docente como se observa en la Ilustración 8.

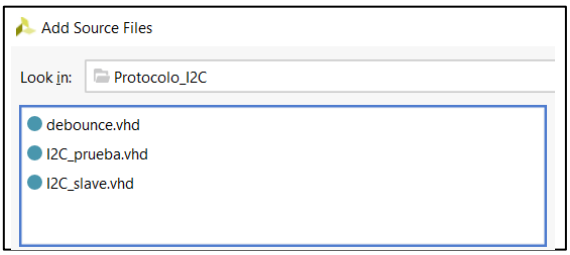

Ilustración 8 Selección de archivos ".vhd".

7. Después que los archivos.vhd se encuentren correctamente cargados dar la opción next, como se observa en la Ilustración 9.

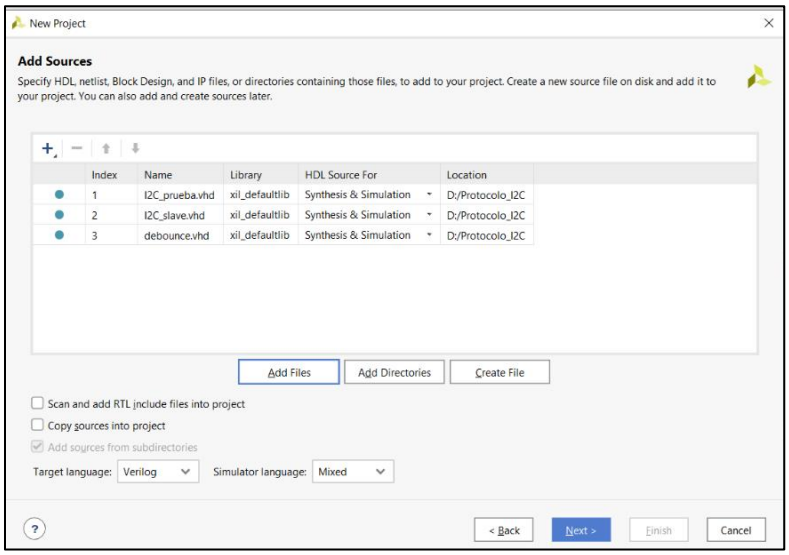

Ilustración 9 Archivos seleccionados en Add sources

8. Se despliega la pestaña Defaul Part como se observa en la Ilustración 10, se selecciona la familia Zylinq-7000, el modelo Xc7z020clg484-1 y se procede a dar Next.

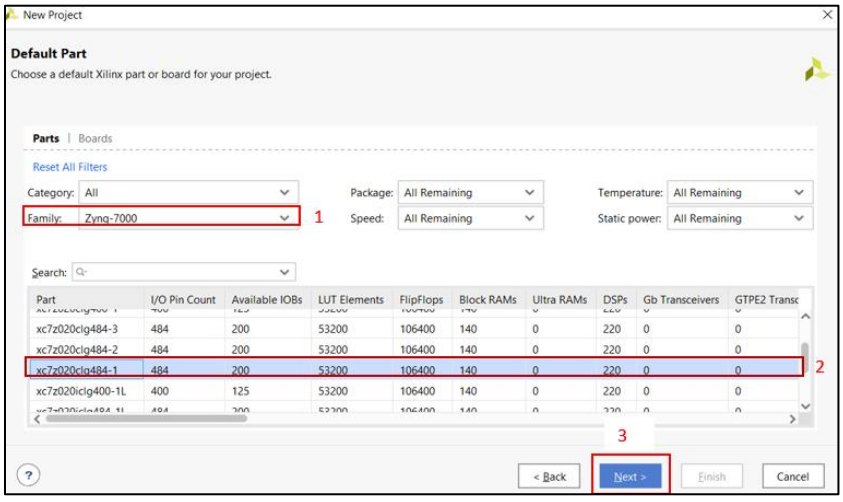

Ilustración 10 Eleccion de placa FPGA

9. Se despliega la interfaz PROJECT MANAGER-PRACTICA 1, como se observa en la Ilustración 11 donde se aprecia los archivos.vhd que se subió en la Ilustración 8.

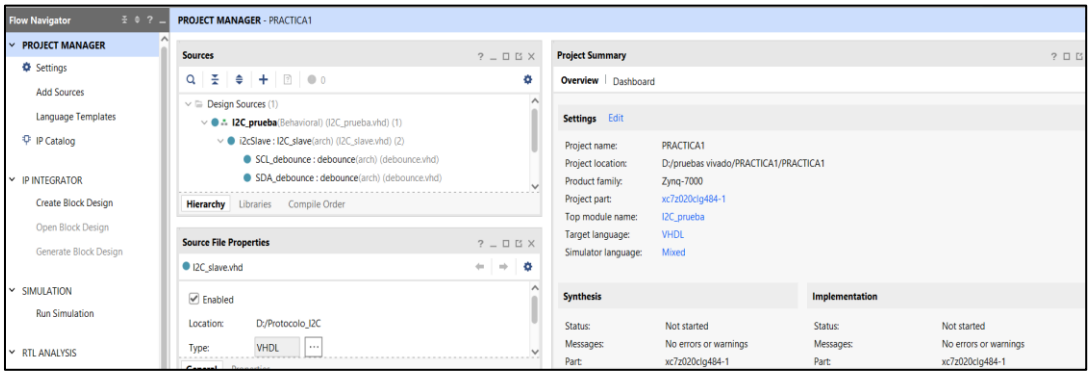

Ilustración 11 Interfaz PROJECT MANAGER de VIVADO

10. Se procede abrir el archivo I2c_Slave en vivado en donde se procede a buscar la parte de bits_processed_reg + y se coloca el valor numérico de 5 como en la Ilustración 12.

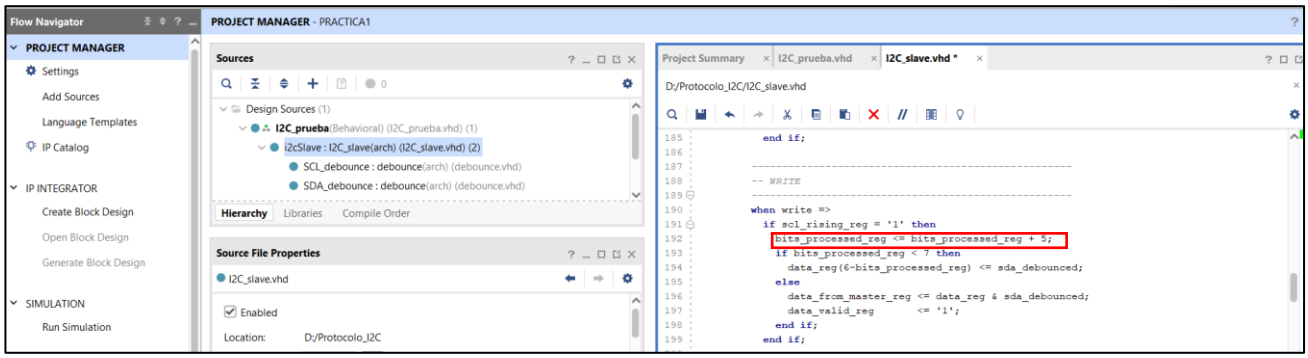

Ilustración 12 I2C_Slave bits procesados

11. Se procede a escoger la opción Add Sources y posterior, Add or create constrains, como se observa en la Ilustración 13.

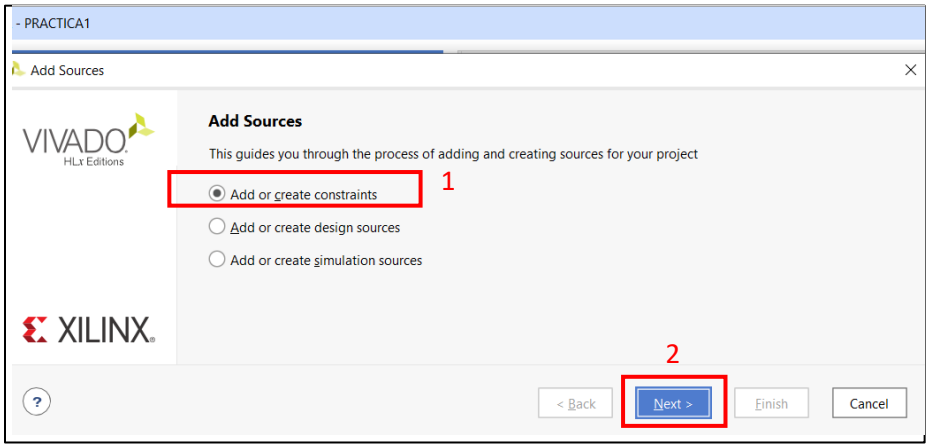

Ilustración 13 Create constraints

12. Se selecciona Create File en donde se despliega la pestaña Add or Create Constraints-Create Constrains file-file name y se escribe cualquier nombre en este caso i2C como se observa en la Ilustración 14.

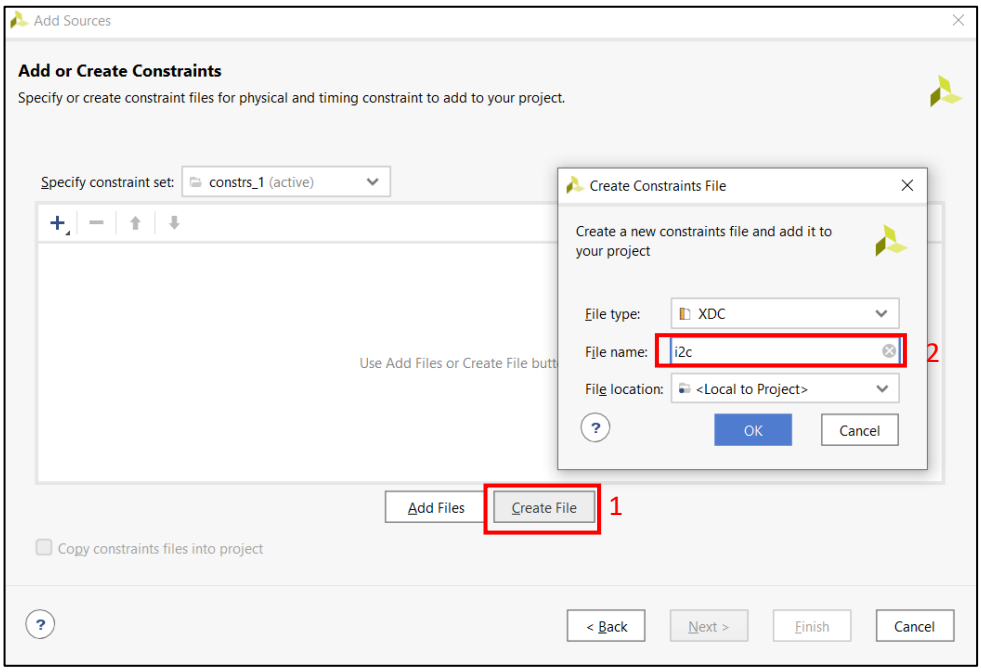

Ilustración 14 Creación del constraints i2c

13. En la Ilustración 15 se encuentra el archivo i2c.xdc respectivamente creado.

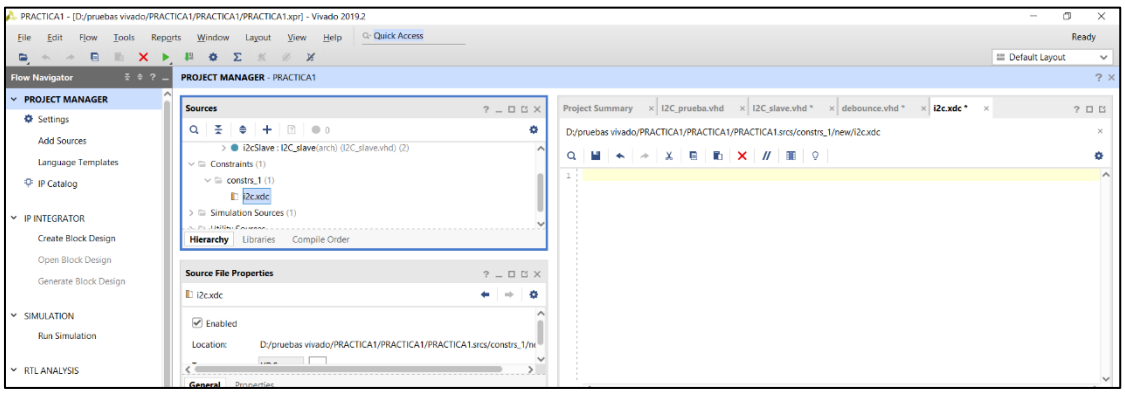

Ilustración 15 Generación de archivo constrainst xdc

14. En base a la Ilustración 16 se procede a escribir el archivo constraints con referencia al archivo.txt proporcionado en la carpeta Protocolo_I2c, donde se puede seleccionar cualquiera de los pines del bloque JA1 y JB1 para la transferencia de datos de la comunicación I2C. Para ser más precisos y seguir el formato de los Pmod, se sugiere utilizar como guía la Ilustración 17.

Pmod	Signal Name	Zynq pin	Pmod	Signal Name	Zynq pin
JA1	JA1	Y11	JB1	JB1	W12
	JA2	AA11		JB2	W11
	JA3	Y10		JB3	V10
	JA4	AA9		JB4	W8
	JA7	AB11		JB7	V12
	JA8	AB10		JB8	W10
	JA9	AB9		JB9	V9
	JA10	AA8		JB10	V8

Pmod	Signal Name	Zynq pin	Pmod	Signal Name	Zynq pin
JC1 Differential	JC1_N	AB6	JD1 Differential	JD1_N	W7
	JC1_P	AB7		JD1_P	V7
	JC2_N	AA4		JD2_N	V4
	JC2_P	Y4		JD2_P	V5
	JC3_N	T6		JD3_N	W5
	JC3_P	R6		JD3_P	W6
	JC4_N	U4		JD4_N	U5
	JC4_P	T4		JD4_P	U6

Pmod	Signal Name	Zynq pin	MIO
JE1 MIO Pmod	JE1	A6	MIO13
	JE2	G7	MIO10
	JE3	B4	MIO11
	JE4	C5	MIO12
	JE7	G6	MIO0
	JE8	C4	MIO9
	JE9	B6	MIO14
	JE10	E6	MIO15

Ilustración 16 Periféricos (Pmod) de la FPGA

Pin #	SPI	I²C	UART	GPIO
1	Chip Select (CS)	Serial Data (SDA)	CTS[1]	I/O[2]
2	Master-Out-Slave-In (MOSI)	Serial Clock (SCK)	Transmit Data (TXD)	I/O[2]
3	Master-In-Slave-Out (MISO)	N/A[3]	Receive Data (RXD)	I/O[2]
4	Serial Clock (SCK)	N/A[3]	RTS[1]	I/O[2]
5	Ground	Ground[4]	Ground	Ground
6	V_{cc}	V_{cc}[5]	V_{cc}	V_{cc}

Ilustración 17 Guía de pines para los protocolos de comunicación

15. De la carpeta proporcionada por el docente Protocolo_I2C, proceda a copiar el formato de asignación de pines como se observa en la Ilustración 18, las entradas y salidas correspondiente al protocolo I2C se escribirán en base a la Ilustración 16 e Ilustración 17.

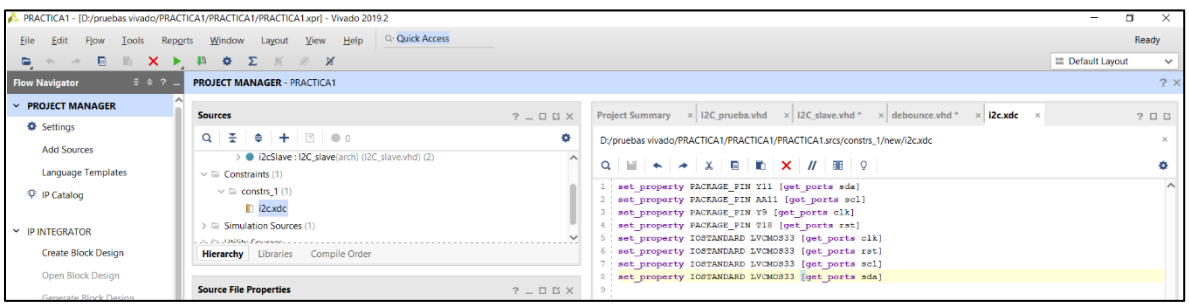

Ilustración 18 Generación del archivo i2c.xdc

16. Ahora se procederá a trabajar con Arduino. Abra el programa Arduino y coloque el código provisto en la sección de Anexos.

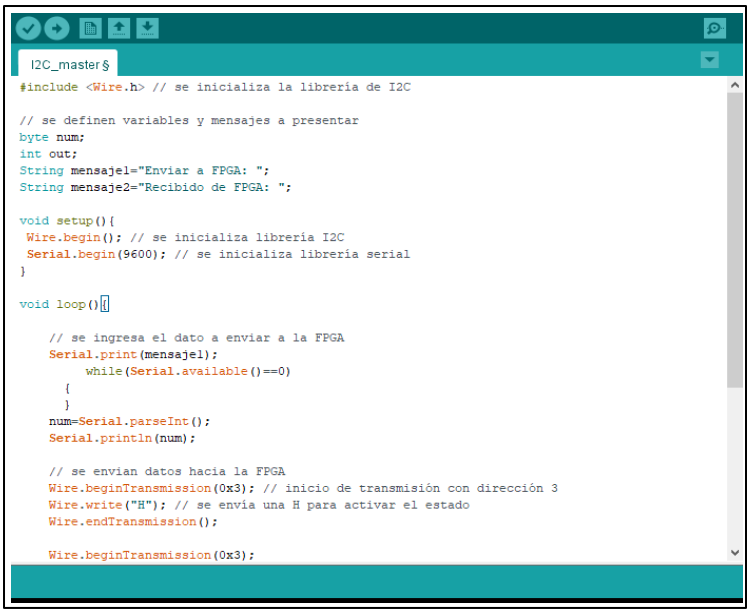

Ilustración 19 Código de Arduino.

17. Seleccione la opción de verificar para realizar la compilación del código y luego proceda a subir el programa a la placa Arduino, como se observa en la Ilustración 20.

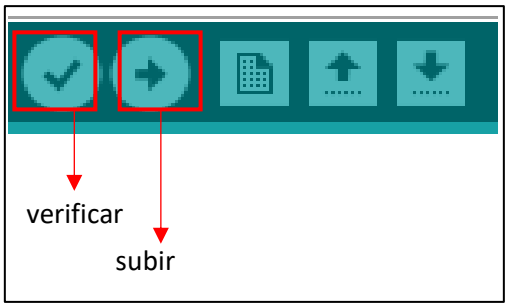

Ilustración 20 Opciones de verificar y subir.

18. Después de realizar todos los pasos anteriores se procede a seleccionar la opción PROGRAM AND DEBUG-Generate Bitstream-Open Hardware Manager-Open Target como se observa en la Ilustración 21.

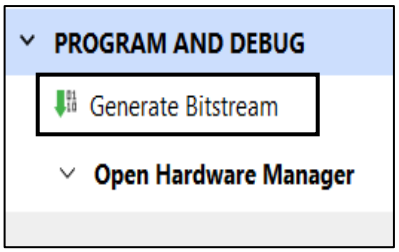

Ilustración 21 Generate Bitstream

19. En la Ilustración 22.Se aprecia que la FPGA se encuentra conectada, pero no programada.

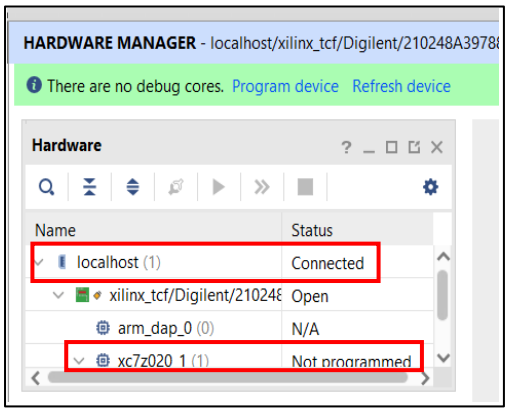

Ilustración 22 FPGA conectada pero no programada

20. Se procede a programar la FPGA para lo cual se selecciona la opción Program device-Program dicha acción produce que la FPGA se encuentre debidamente programada como se observa en la Ilustración 23.

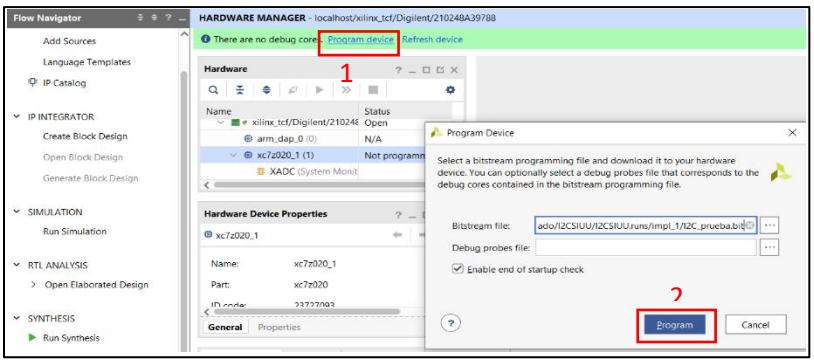

Ilustración 23 Programando FPGA

21. Después de realizar todos los pasos previos, se puede observar que la FPGA se encuentra debidamente programada, tal como lo indica la Ilustración 24.

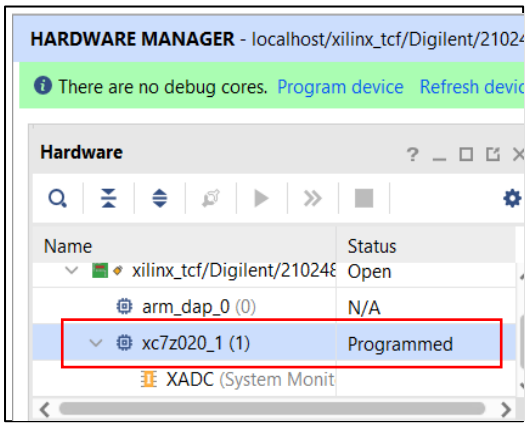

Ilustración 24 FPGA programada

BIBLIOGRAFÍA

[1] Digilent, «Digilent,» 27 01 2014. [En línea]. Available: https://digilent.com/reference/programmable-logic/zedboard/reference-manual. [Último acce 15 06 2022].

[2] A. Pandit, «Circuit Digest,» 29 04 2019. [En línea]. Available: https://circuitdigest.com/tutorial/serial-communication-protocols. [Último acceso: 03 07 2022

ANEXOS

Código de Arduino

```
// se ingresa el dato a enviar a la FPGA
  Serial.print(mensaje1);
     while(Serial.available()==0)
   {
   }
  num=Serial.parseInt();
  Serial.println(num);

  // se envian datos hacia la FPGA
  Wire.beginTransmission(0x3); // inicio de transmisión con dirección 3
  Wire.write("H"); // se envía una H para activar el estado
  Wire.endTransmission();

  Wire.beginTransmission(0x3);
  Wire.write(num); // se envía el número ingresado
  Wire.endTransmission();

  delay(1);

  // se reciven datos de la FPGA
  Wire.requestFrom(3,1); // se piden datos al slave con dirección 3
  while(!Wire.available()){;}
  out=Wire.read(); // se lee el dato enviado por la FPGA
```

```
  // impresión del número ingresado (num) + 5
  Serial.print(mensaje2);
  Serial.println(out);
  delay(1000);
}
```

Conexiones de los dispositivos

FPGA	ARDUINO
SDA PIN JA1	SDA PIN A4
SCL PIN JA2	SCL PIN A5
GROUND	GROUND

Tabla 1 Conexiones FPGA - Arduino.

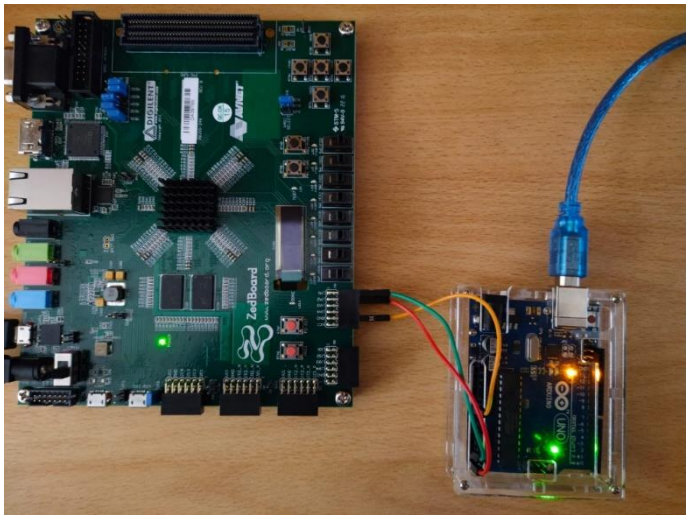

Ilustración 25 Conexiones FPGA - Arduino.

3.3V	GND	3	2	1	0
3.3V	GND	7	6	5	4

Ilustración 26 Formato del Pmod.

PRÁCTICA # 2

SISTEMA DE SENSORES CON COMUNICACIÓN INALÁMBRICA

OBJETIVOS:

- Programar en Vivado el ADC de la FPGA para que realice la conversión de las señales obtenidas de los sensores de temperatura y movimiento.
- Programar en Vivado el procesador de la FPGA para que transmita, mediante el protocolo UART, los datos del ADC hacia el módulo Xbee.
- Generar un código en Vitis que lea los datos sensados y los transmita hacia el módulo Xbee.
- Configurar los módulos Xbee para que proporcionen un medio de comunicación inalámbrica entre la FPGA y una PC.

MATERIALES Y HERRAMIENTAS:

Software Vivado y Vitis 2019.2	FPGA Zedboard
Cables jumpers	Fuente de alimentación 12 V
Sensor de temperatura	Fuente de alimentación 5 V
Sensor de movimiento	Módulo Xbee (2)
Cable USB a micro USB (2)	Módulo adaptador Xbee Explorer (2)

BREVE EXPLICACIÓN DE LA PRÁCTICA:

Para la implementación de la práctica “Sistema de sensores con comunicación inalámbrica” se trabajó con el diagrama de bloques que se observa en la Ilustración 1. La primera parte se compone de la adquisición de datos realizada por los sensores de movimiento y temperatura. La segunda etapa está conformada por la configuración de los bloques: procesador y XADC de la FPGA. La siguiente etapa consiste en la configuración de los módulos Xbee para que permitan obtener una comunicación

inalámbrica. Finalmente se tiene la visualización de los datos, donde se trabaja con una interfaz diseñada en LabVIEW.

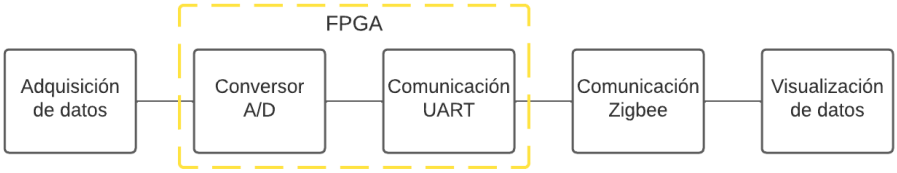

Ilustración 1 Diagrama de bloques general del sistema.

MARCO TEÓRICO:

Un sistema de sensores se compone por dispositivos encargados de recolectar datos en un entorno o área específica. Cada componente realiza la adquisición de distintos valores de interés, para luego transmitirlos hacia un dispositivo central de mayor complejidad que se encargará de su posterior procesamiento. Debido a que muchas de las veces estos sistemas se observan en ambientes exteriores o rurales, es necesario diseñar una red propia para la comunicación de los datos para lo cual se tienen distintas opciones.

El protocolo Zigbee tiene como base el estándar IEEE 802.15.4, que se utiliza para redes inalámbricas. Entre sus características esta que: presenta baja latencia, rango medio en su ancho de banda, utiliza las bandas ISM, bajo consumo y alcance medio.

Este protocolo maneja tres tipos de dispositivos: coordinador, router, dispositivo final. Dentro de una red Zigbee debe haber un solo coordinador, este se encarga de iniciar la red. El router se encargará de generar el enrutamiento, para redes que tienen más de un salto; se pude conectar a un coordinador, a otro router o a un dispositivo final. El dispositivo final es el que posee la menor jerarquía, se puede conectar con un ruter o directamente al coordinador. En la Ilustración 2 Ejemplo de red Zigbee. se observa un ejemplo de red Zigbee.

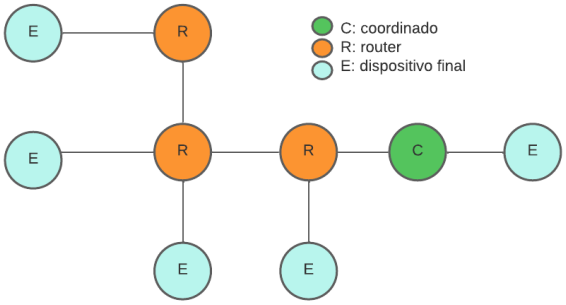

Ilustración 2 Ejemplo de red Zigbee.

El módulo que incorpora este protocolo es el módulo Xbee, para su programación se trabaja con el software XCTU donde se define el modo de trabajo de cada elemento de la red. Generalmente junto al módulo se utiliza el Explorer, ya que este permite conectar con mayor facilidad el módulo al proveer un puerto mini USB. De esta forma mediante un solo cable se puede conectar el módulo a la PC y realizar su configuración.

Ilustración 3 Módulo Xbee y explorer.

INSTRUCCIONES DE LA PRÁCTICA:

LABORATORIO:

1. Proceda a abrir el programa Vivado. Seleccione la opción **Create Project**, luego seleccione **Next** en la nueva ventana desplegada (Ilustración 4).

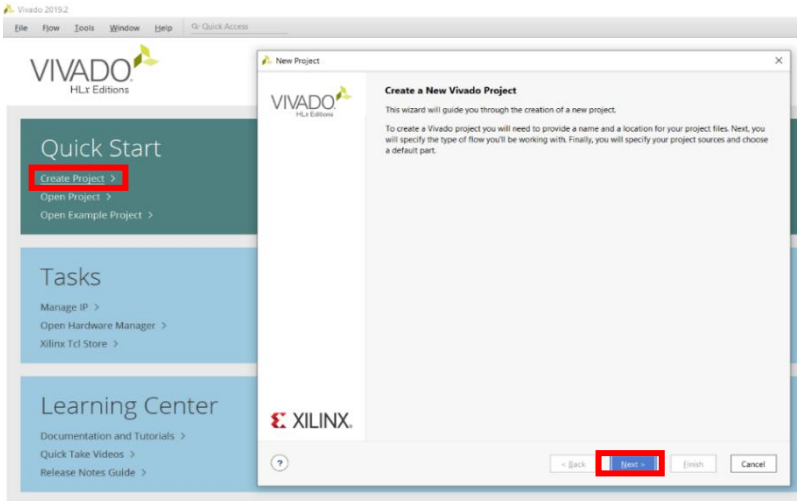

Ilustración 4 Creación del proyecto.

2. Coloque un nombre al proyecto y elija la ruta donde lo vaya a crear (Ilustración 5), de preferencia elija una ruta que no tenga espacios en blanco o caracteres especiales. Luego seleccione **Next**.

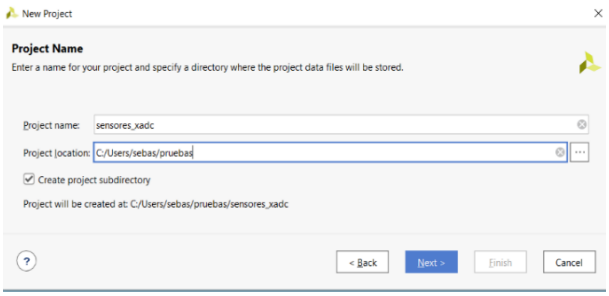

Ilustración 5 Nombre del proyecto.

3. Seleccione la opción **RTL Project** (Ilustración 6) y marque la casilla que se encuentra por debajo.

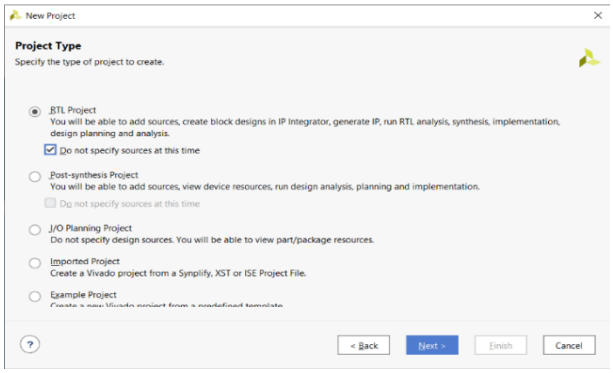

Ilustración 6 Tipo de proyecto.

4. Seleccionar la placa a trabajar, especifique la familia de la FPGA (Ilustración 7) así como sus características. Luego seleccione **Next**.

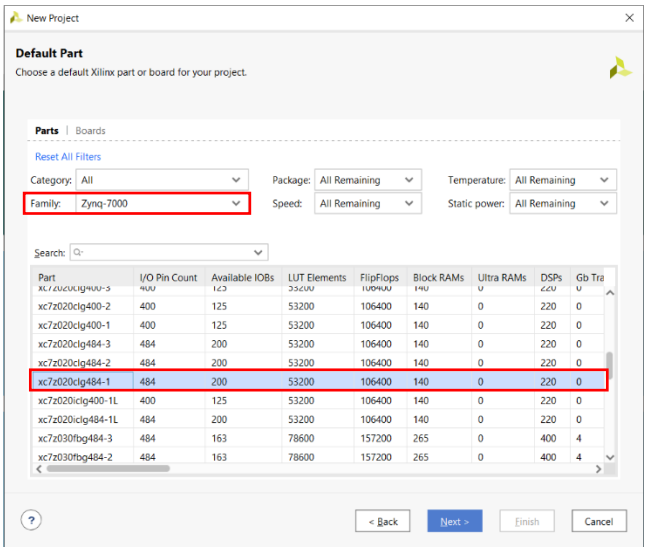

Ilustración 7 Selección de placa.

5. Finalmente, para crear el proyecto selecciones **Finish** (Ilustración 8).

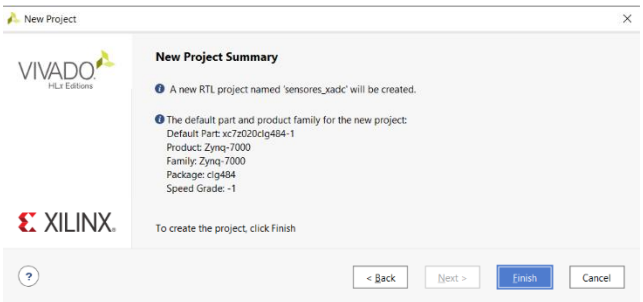

Ilustración 8 Resumen del proyeto.

6. Una vez creado el proyecto se desplegará la ventana principal de Vivado (Ilustración 9). Aquí se realiza toda la programación y configuración de la FPGA. En el panel de la izquierda (**Navegador de Flujo**), en las opciones de **IP Integrator** seleccione **Create Block Desing**.

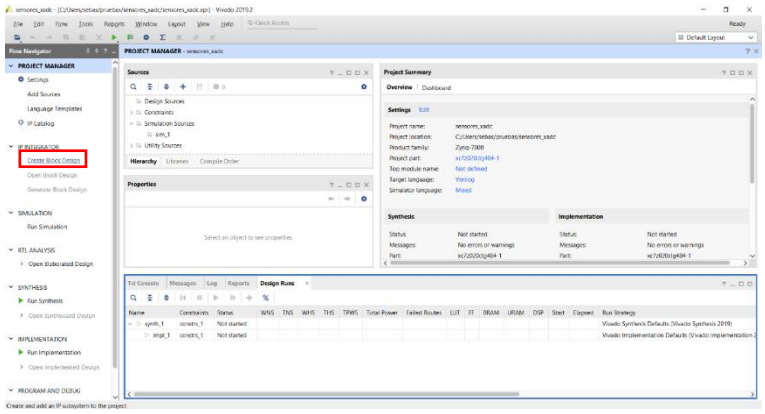

Ilustración 9 Ventana principal.

7. En la ventana desplegada coloque el nombre del diseño (Ilustración 10), las demás opciones puede dejarlas por defecto. Luego seleccione **OK**.

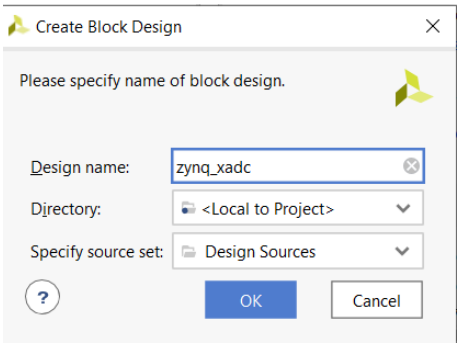

Ilustración 10 Nombre del diseño de bloques.

8. En la ventana principal se observará una nueva sección de **Diagram** (Ilustración 11), aquí se realizará el diseño por boques. Entre las distintas opciones de esta sección, elija + para insertar los bloques deseados. Para maximizar la ventana del diagrama seleccione  .

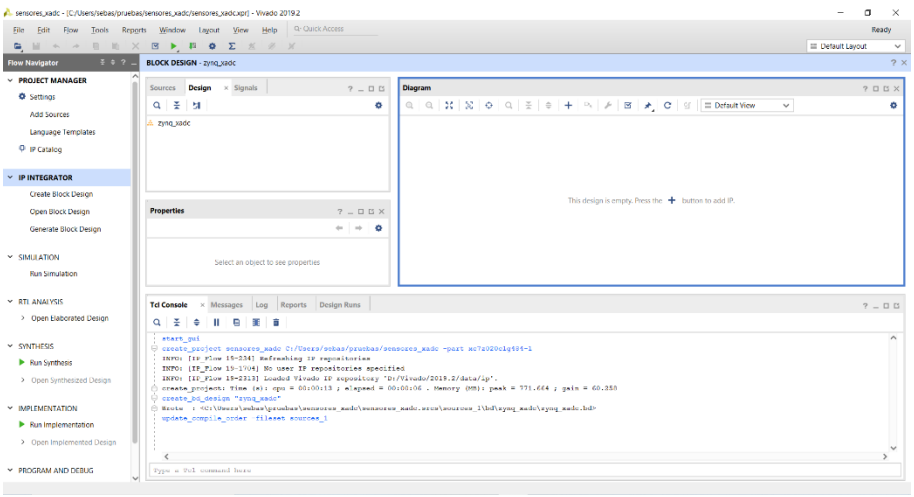

Ilustración 11 Ventana de diagramas.

9. En la ventana desplegada coloque el nombre del bloque a colocar (Ilustración 12), en este caso se utilizará el procesador **ZYNQ** y el convertidor **XADC**.

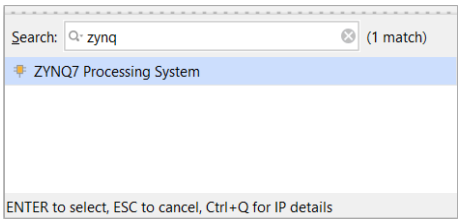

Ilustración 12 Buscador de bloques.

10. En la sección de diagramas aparecerán los dos bloques ingresados (Ilustración 13), al dar doble click sobre ellos puede acceder a las configuraciones de estos.

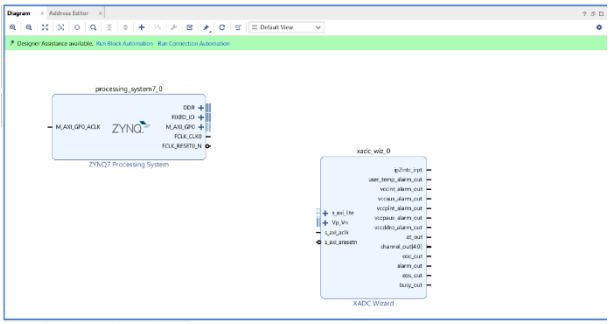

Ilustración 13 Bloques del procesador y XADC.

11. Dar doble clic sobre el bloque **ZYNQ**, en la primera pestaña de la ventana desplegada se observan las características disponibles a configurar del procesador (Ilustración 14).

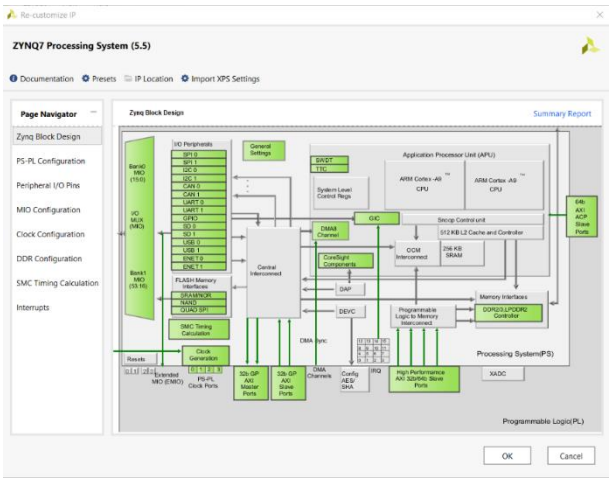

Ilustración 14 Ventana principal del procesador.

12. En la pestaña **MIO Configurations** (Ilustración 15), activar la casilla de UART 0 y definir los pines MIO 10 y 11. Por defecto define a Tx como el pin MIO 11 y Rx como MIO 10.

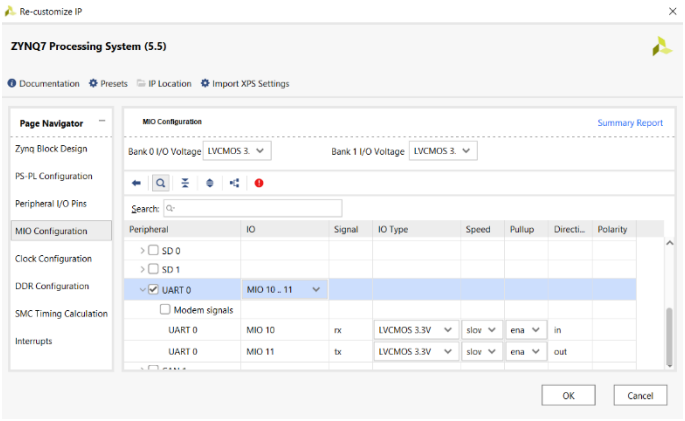

Ilustración 15 Configuración de puertos MIO.

13. En la pestaña de **PS-PL Configuration** (Ilustración 16) se puede definir el Baud Rate a trabajar, en este caso se dejará el valor de 115200 por defecto. Finalmente, dar clic en OK.

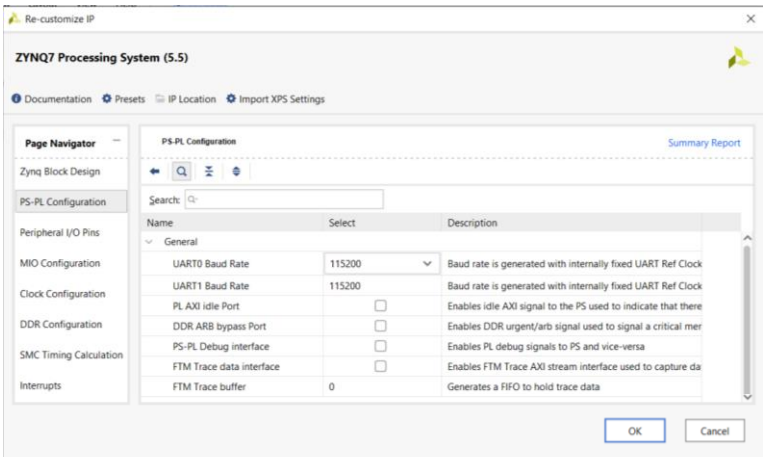

Ilustración 16 Configuración de baud rate.

14. Para configurar el XADC dar doble clic sobre él, se desplegará una nueva ventana. En la pestaña de **Basic** (Ilustración 17) seleccionar la opción **Channel Sequencer**.

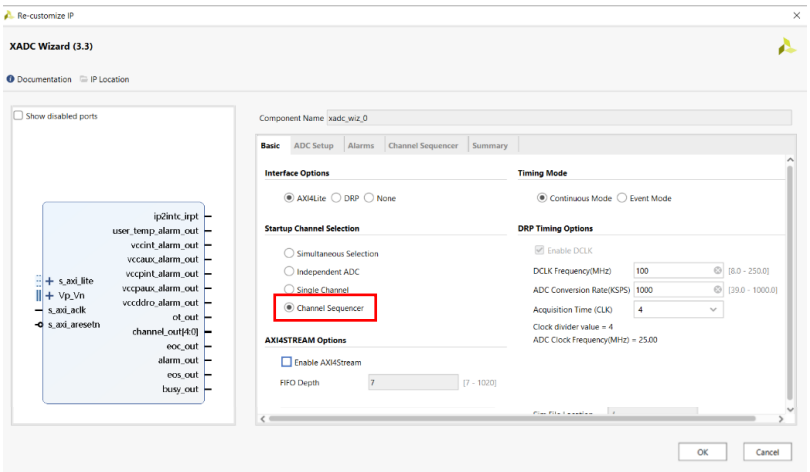

Ilustración 17 Ventana principal del XADC.

15. En la pestaña de **ADC Setup** (Ilustración 18) desmarcar todas las opciones de calibraciones y sensores, definir un **Channel averaging** de 16.

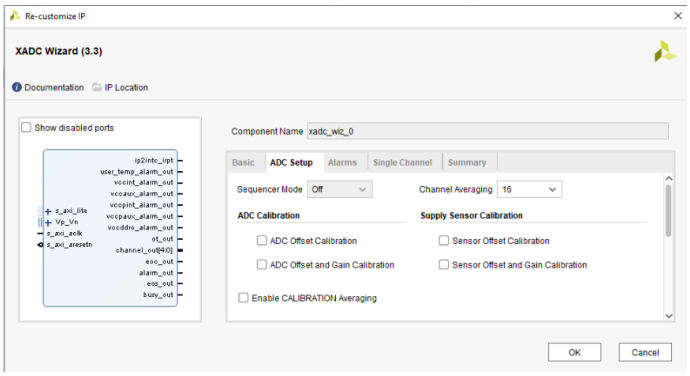

Ilustración 18 ADC setup.

16. En la pestaña de **Alarms** (Ilustración 18) desmarcar todas las alarmas disponibles.

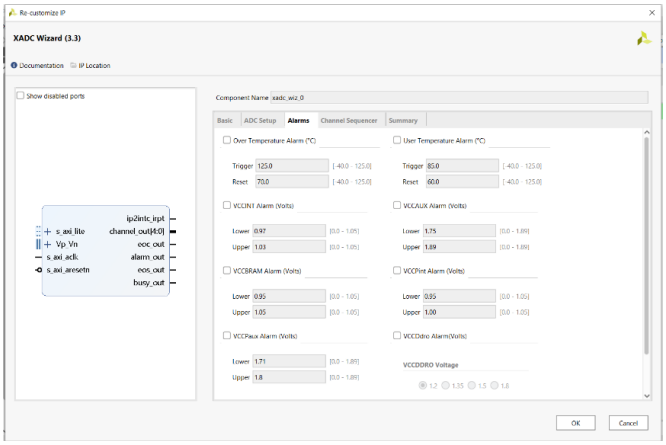

Ilustración 19 Alarmas del XADC.

17. En la pestaña de **Channel Sequencer**, habilitar los canales a utilizar con su promediado. En la Ilustración 20 se habilitó el canal 0, habilité el otro canal disponible guiándose con la descripción del XADC en la Ilustración 56 en Anexos.

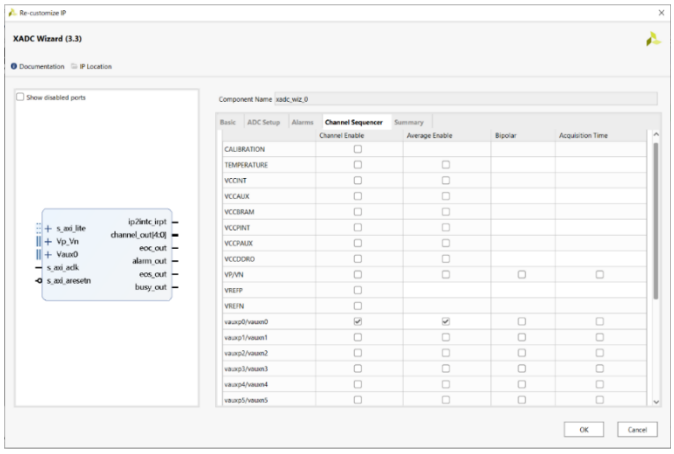

Ilustración 20 Canales del XADC.

18. En la pestaña **Summary** se observan las configuraciones realizadas (Ilustración 21), finalmente de click en OK.

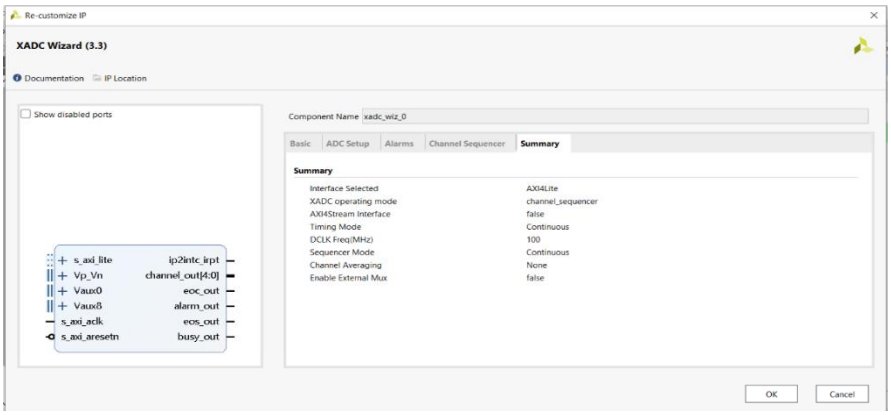

Ilustración 21 Resumen de configuraciones.

19. Una vez configurados ambos bloques, en el XADC se harán externas las entradas correspondientes a los canales habilitados. Teniendo presionado la tecla “ctrl” se selecciona el vauxn0 y vauxp0, se pondrán de color naranja (Ilustración 22); de igual forma seleccionar el otro segundo canal que habilitó.

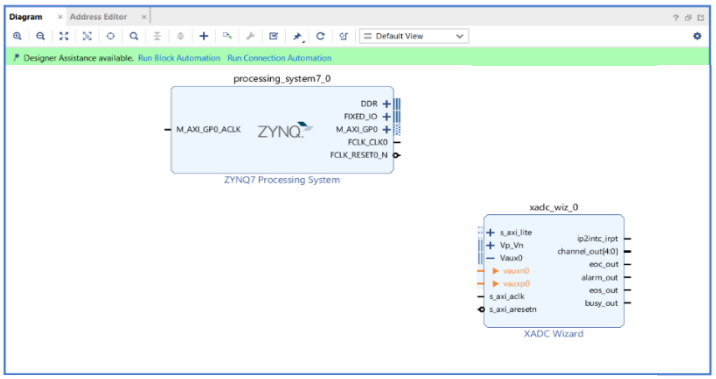

Ilustración 22 Bloques ya configurados.

20. Dar clic derecho sobre los canales seleccionados y elegir **make external** (Ilustración 23).

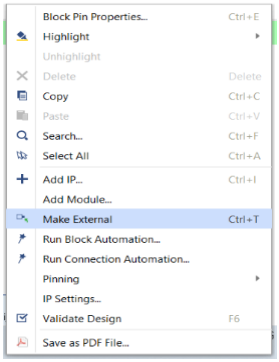

Ilustración 23 Definiendo entradas como externas.

21. Los bloques configurados deben observar cómo se presenta en la Ilustración 24, así como se observan las entradas del canal 0 deber visualizarse las entradas del segundo canal que habilitó. Seleccionar **Run Connection Automation** que se encuentra sobre la franja verde.

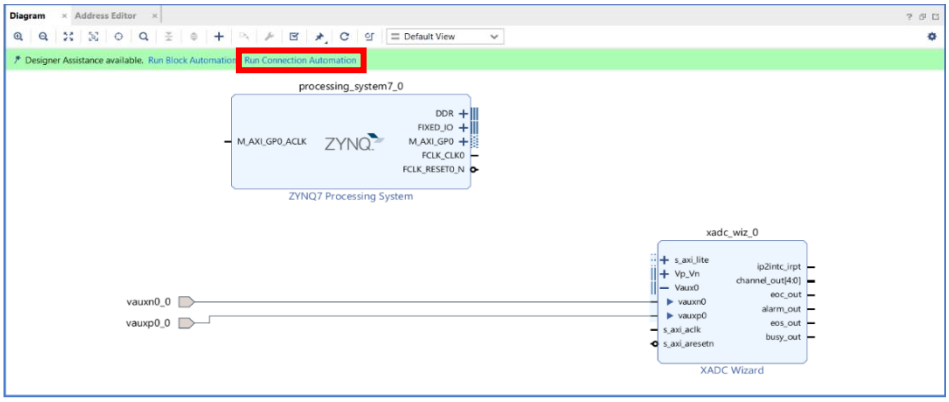

Ilustración 24 Realizar conexiones automáticamente.

22. En la ventana desplegada (Ilustración 25) comprobar que se encuentren marcadas todas las casillas de la izquierda. Dar clic en OK.

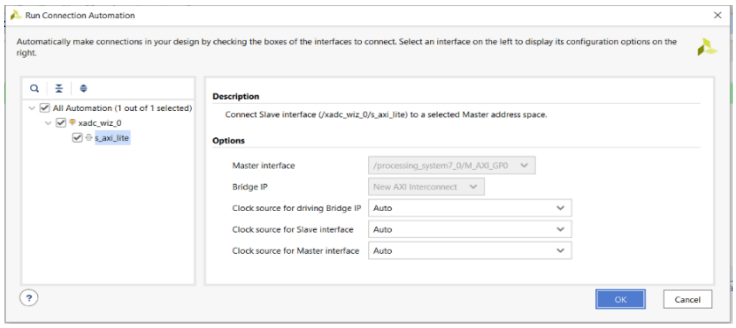

Ilustración 25 Ventana de conexiones.

23. Se agregarán dos nuevos bloques y de forma automática se realizarán las conexiones entre todos ellos, como se observa en la Ilustración 26. Seleccionar **Run Block Automation.**

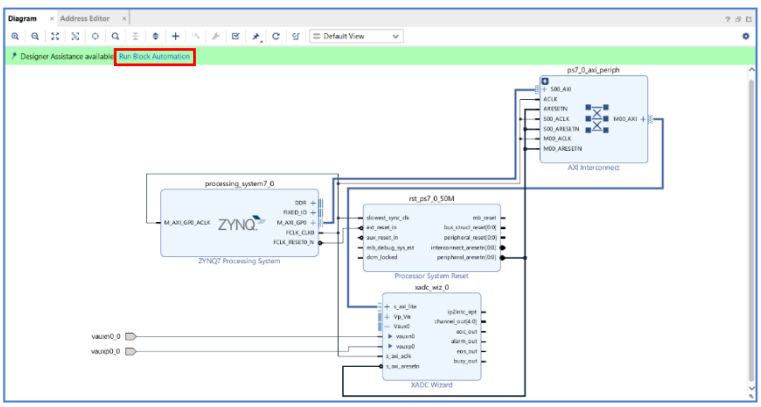

Ilustración 26 Conexiones realizadas.

24. En la ventana desplegada (Ilustración 27) marcar todas las casillas de la izquierda, y seleccionar OK.

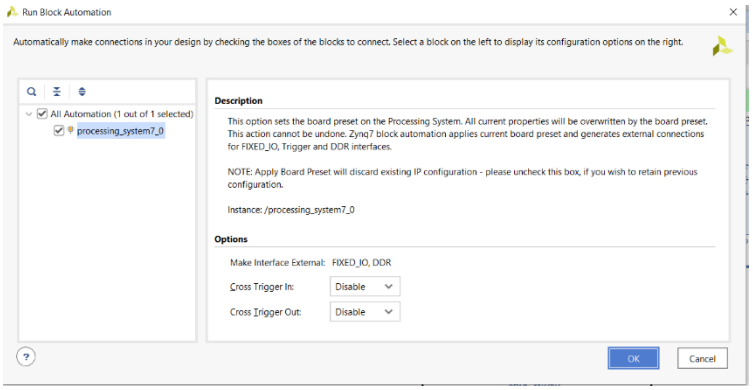

Ilustración 27 Automatización de los bloques.

25. La franja verde de la parte superior habrá desaparecido, para finalizar el diagrama seleccionar **Validate Desing** (Ilustración 28). Si la validación se realizó correctamente le aparecerá un mensaje de "Validation Successful".

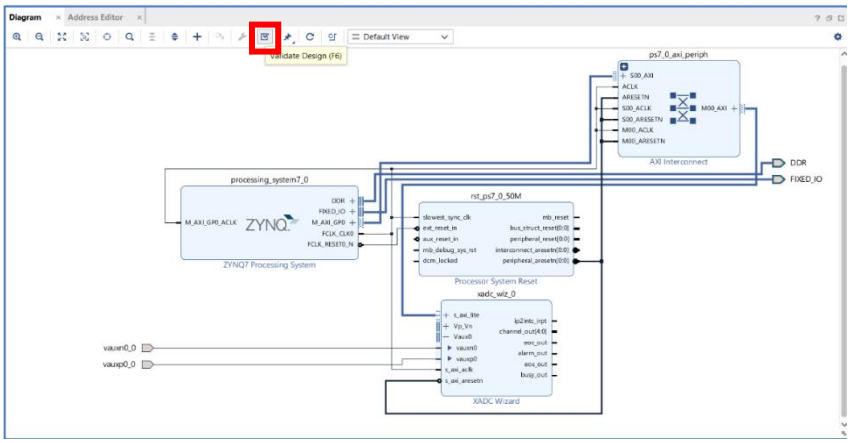

Ilustración 28 Validación del diseño.

26. Finalizado el diagrama se debe crear un HDL Wrapper, para ello en la sección **Sources**, se despliega la pestaña de **Desing Sourcesn**. Aquí aparecerá el archivo del diagrama con el nombre que le fue asignado, dar click derecho sobre este y seleccionar "Create HDL Wrapper" (Ilustración 29).

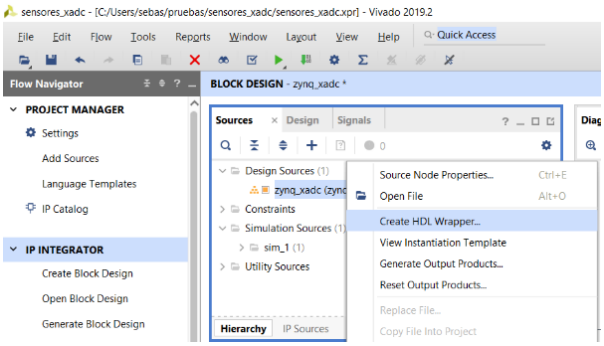

Ilustración 29 Creación de HDL wrapper.

27. Se mostrará una nueva ventana (Ilustración 30), marcar la segunda opción y seleccionar OK.

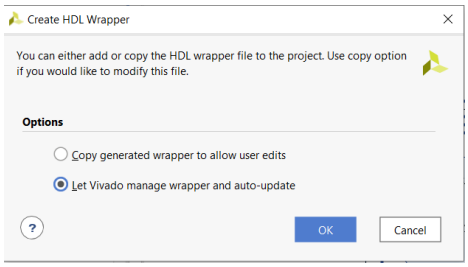

Ilustración 30 Configuraciones de wrapper.

28. En la sección de **Sources→Desing Sources** se generan dos nuevos archivos (Ilustración 31), en estos se definen las configuraciones realizadas en los bloques, pero mediante lenguaje VHDL.

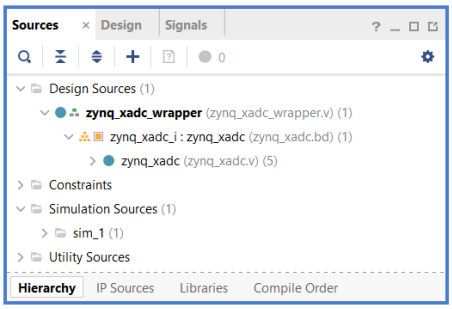

Ilustración 31 Archivos generados.

29. A este punto todas las configuraciones en Vivado han terminado, solo queda generar el Bitstream. Seleccionar la opción de **Generate Bitstream**  **,** que se encuentra sobre la sección de Sources.

30. Se mostrará un mensaje el cual indica que aún no se ha realizado un síntesis e implementación (Ilustración 32), al seleccionar **YES** estos dos procesos se realizarán automáticamente.

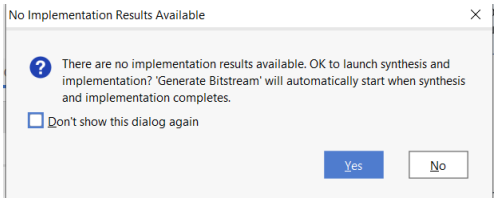

Ilustración 32 Ejecución de síntesis e implementación.

31. Se desplegará una ventana en la que se pueden indicar el número de "trabajos" (Ilustración 33), dejar los valores predeterminados y seleccionar OK.

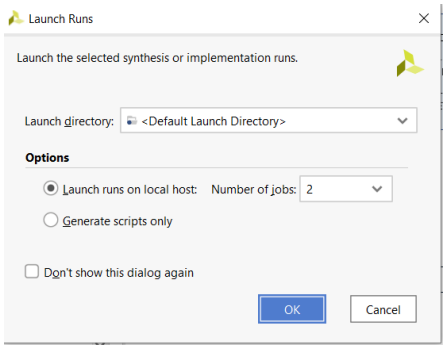

Ilustración 33 Ventana launch runs.

32. En la parte superior derecha de la ventana se observarán todos los procesos que se generan, al finalizar se mostrará una ventana que indica que el bitstream se generó exitosamente (Ilustración 34). Si desea ver a detalle las características que se utilizarán de la FPGA puede seleccionar **OK**, caso contrario seleccione **Cancel.**

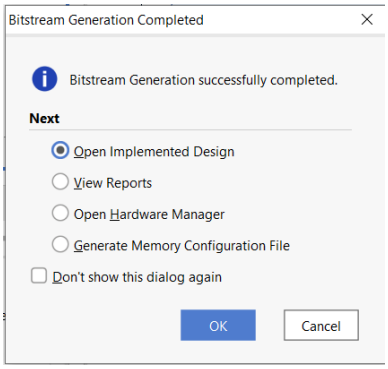

Ilustración 34 Confirmación de bitstream generado.

33. Ya que se está trabajando con bloques, es necesario crear una aplicación que los utilice. Para ello se procederá a exportar el hardware (Ilustración 35) en **File → Export → Export Hardware.**

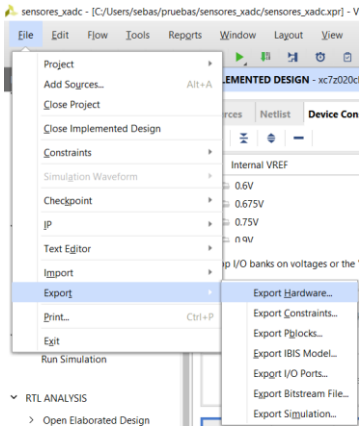

Ilustración 35 Exportar hardware.

34. En la ventana desplegada marcar la casilla de incluir bitstream (Ilustración 36), definir un nombre para el archivo XSA a generar y la ruta donde se guardará. De preferencia defina la misma ruta que para todos los archivos anteriormente trabajados.

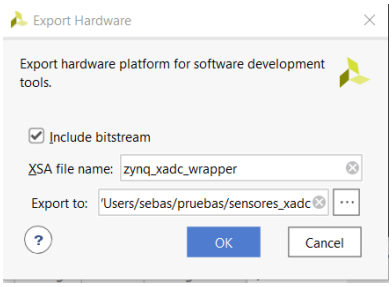

Ilustración 36 Incluir bitstream.

35. Ahora se realizará la programación en Vitis. Al abrir Vitis se desplegará una ventana en la que le pide seleccionar un área de trabajo (Ilustración 37), es importante que la ruta sea la misma donde se encuentra el archivo XSA.

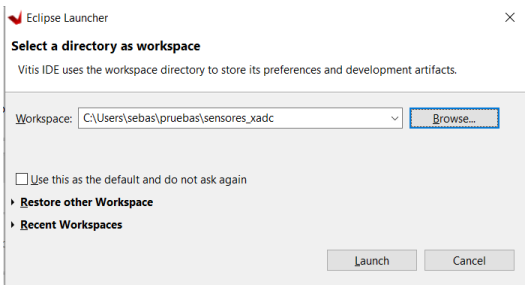

Ilustración 37 Workspace a utilizar.

36. Se desplegará una nueva ventana (Ilustración 38), seleccionar **Create Application Project.**

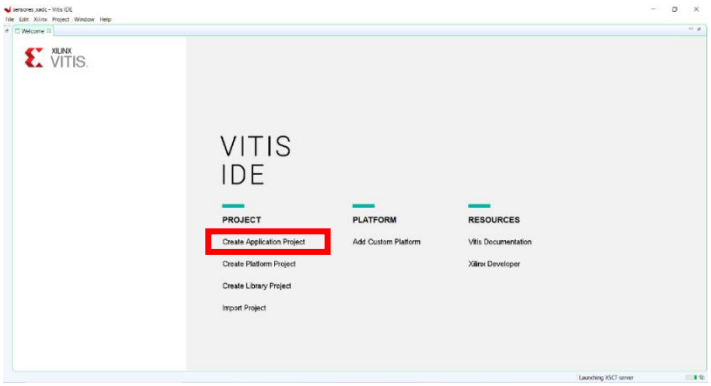

Ilustración 38 Creación del proyecto en Vitis.

37. Colocar un nombre al proyecto.

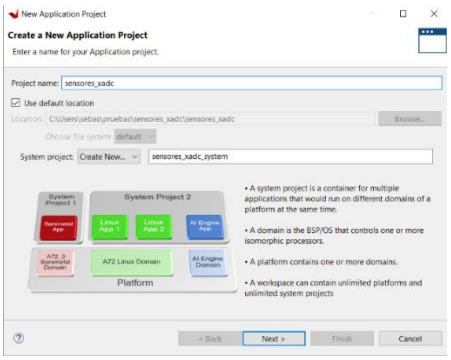

Ilustración 39 Nombre del proyecto.

38. Ubicarse en la pestaña **Create a new platform**. Esta plataforma se basará en el archivo XSA que se generó anteriormente. Para agregar un nuevo archivo (Ilustración 40) a la lista seleccione . Debe buscar el archivo y seleccionarlo.

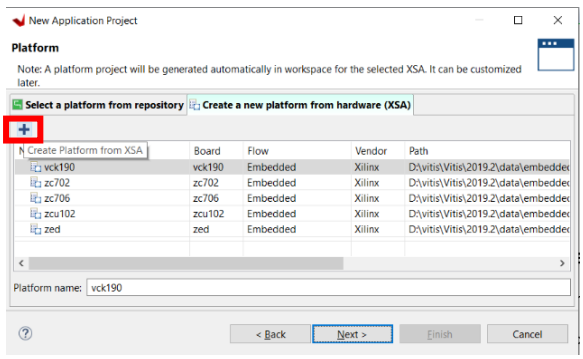

Ilustración 40 Agregar archivo XSA.

39. Una vez seleccionado el archivo de click en **Next**.

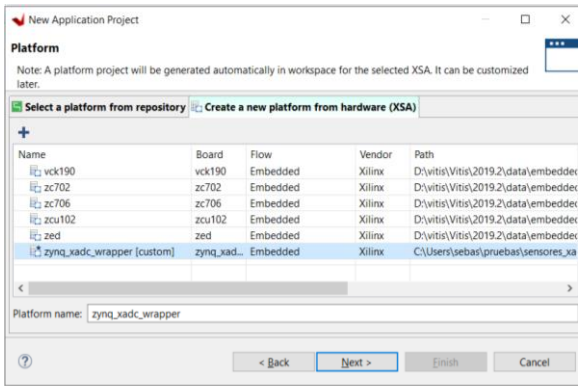

Ilustración 41 Archivo agregado.

40. En la siguiente ventana permite elegir el CPU, OS y lenguaje de programación a utilizar (Ilustración 42). Verificar que el lenguaje sea C y los demás campos dejarlos en sus valores por defecto; clic en **Next.**

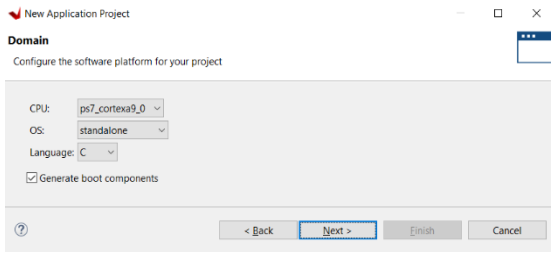

Ilustración 42 Lenguaje a utilizar en el proyecto.

41. Se muestran algunas plantillas de programas en C a utilizar, seleccionar la opción “Hello world” (Ilustración 43) y clic en **Finish**.

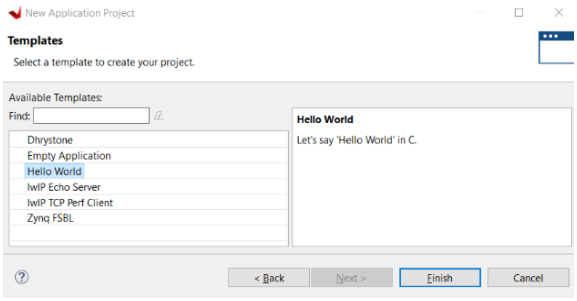

Ilustración 43 Plantilla a utilizar.

42. Se muestra la ventana principal de Vitis donde se escribirá el código (Ilustración 44), también se realizará el debug y ejecución del código.

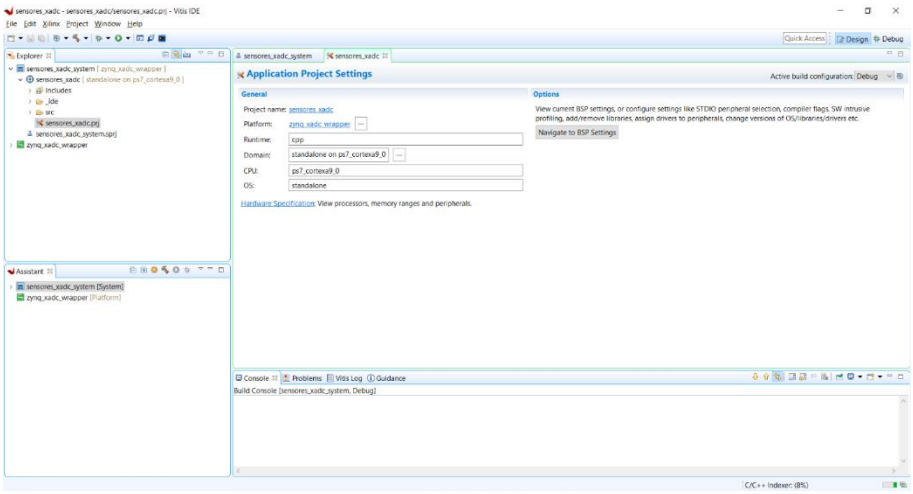

Ilustración 44 Ventana principal de Vitis.

43. A la izquierda se tiene le explorador, aquí se encuentran todos los archivos con los que cuenta el proyecto. El archivo “Hello world” se encuentra en la carpeta **src→hello world** (Ilustración 45). Ya que este código solo imprime un mensaje por pantalla, se realizó un código que utilice los bloques definidos en Vivado.

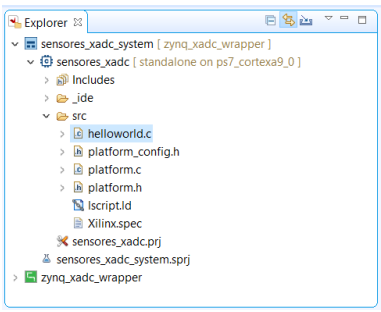

Ilustración 45 Ubicación de la plantilla generada.

44. En la Ilustración 46 se presenta el código, este se encargará de leer los valores obtenidos mediante el canal 0 y escribirlos en el registro del XADC. Luego se realiza la conversión de los valores para que se muestren en mV, para finalmente hacer un print de ellos. En este caso en lugar de presentarse por pantalla, al hacer print los datos se envían hacia los pines de salida del protocolo UART 0. Debe agregar el código para escribir, convertir y presentar los datos para el segundo canal que habilitó. Cada canal tiene un offset que se debe agregar a C_BASEADDR para poder ingresar los valores, utilice la Ilustración 57 en Anexos para guiarse según el canal que haya habilitado.

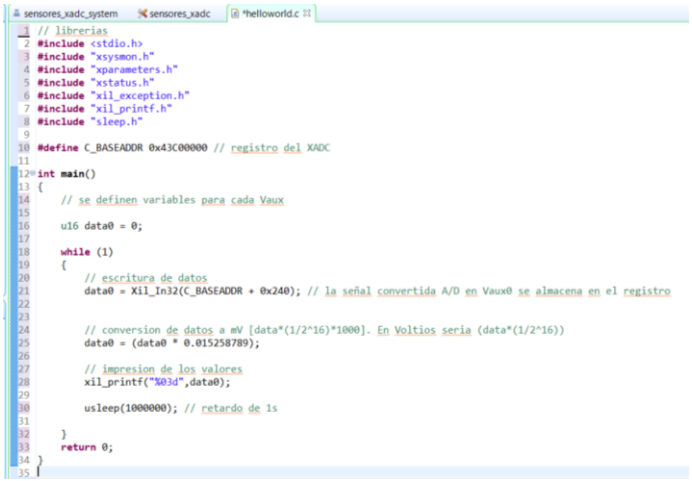

Ilustración 46 Código de escritura y envío de datos sensados.

Al realizar el print se recomienda escribir los valores de acuerdo con el siguiente formato, ya que así la interfaz diseña trabajará adecuadamente:

xil_printf(“%03d %03d”,variable1,variable2);

El valor %03d indica que se trabajará con 3 decimales. Para un correcto funcionamiento de la interfaz en LabVIEW la primera variable debe contener los valores de temperatura.

45. Una vez finalizado el código proceda a realizar el debug en la opción que se encuentra sobre la sección del explorador. Al finalizar se mostrará un mensaje (Ilustración 47) en la consola.

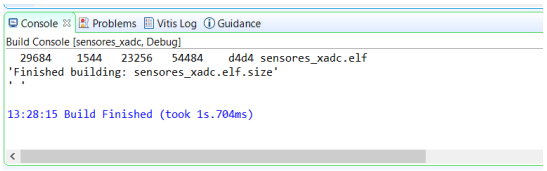

Ilustración 47 Mensaje de debug finalizado.

El código en Vitis estáa listo para ejecutarse, ahora se deben configurar los módulos Xbee.

46. Abrir el programa XCTU. Para detectar un nuevo módulo seleccione la opción **Discover** marcada en la Ilustración 48.

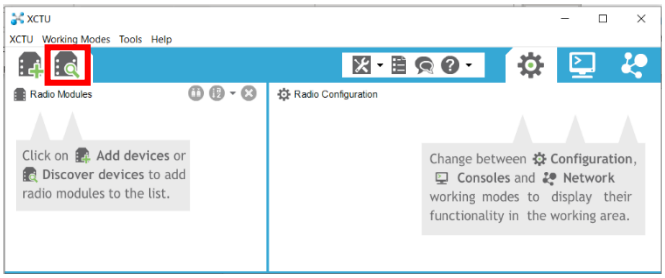

Ilustración 48 Búsqueda de dispositivos en XCTU.

47. En la ventana desplegada seleccione el puerto COM (Ilustración 49) al cual se encuentra conectado el módulo, luego de click **Next**.

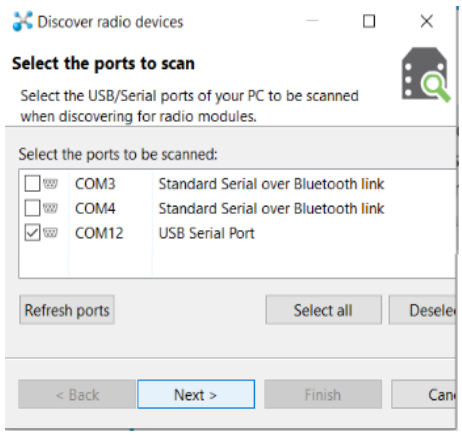

Ilustración 49 Puerto de conexión con el módulo.

48. Seleccionar los parámetros del puerto (Ilustración 50), asegurarse que el baud rate concuerde con el definido en el procesador de la FPGA que en este caso es de 115200. En las demás casillas dejar los valores por defecto, luego selecciones **Finish**.

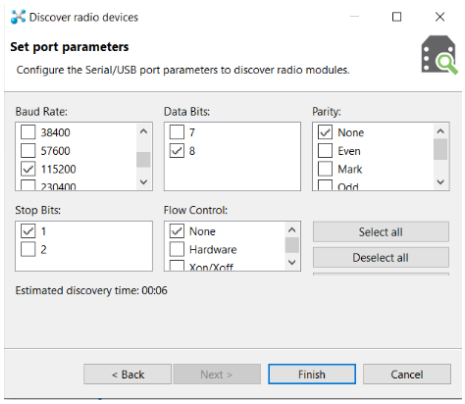

Ilustración 50 Características del puerto.

49. Se observará el proceso de detección de los módulos conectados.

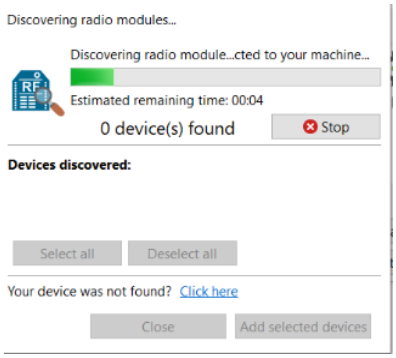

Ilustración 51 Buscando módulos.

50. En la ventana desplegada aparecerá el módulo detectado, marcar la casilla junto al módulo y dar click en **Add selected devices.**

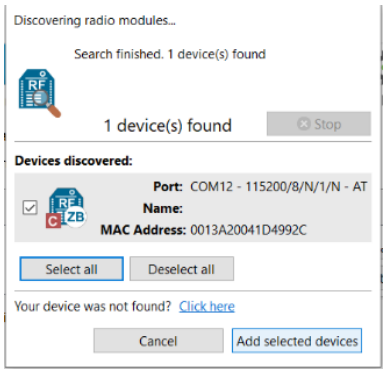

Ilustración 52 Módulos encontrados.

51. En la ventana principal de XCTU aparece a la izquierda el módulo detectado y a la derecha los parámetros a configurar.

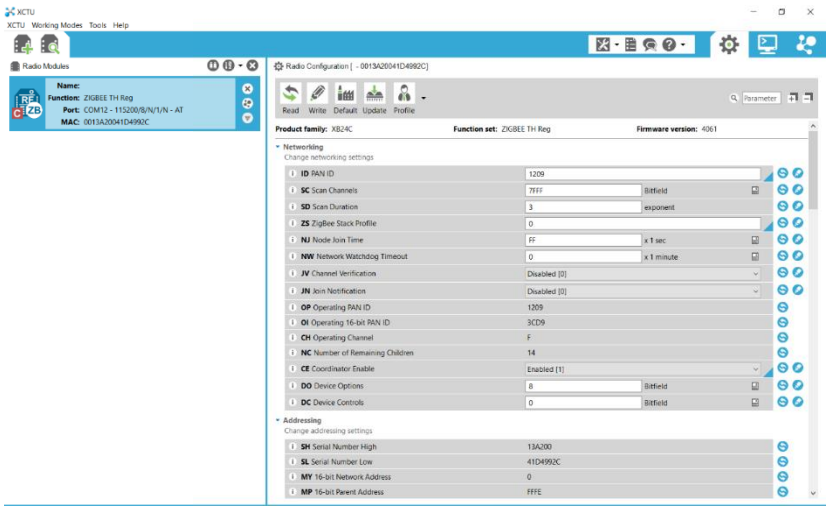

Ilustración 53 Configuraciones disponibles.

52. Los módulos deben configurarse como coordinador y como router. El router estará conectado a la FPGA y el coordinador a la PC. En base a la Tabla 1 en Anexos configure los módulos.

53. Seleccione la opción **Write**  para guardar todos los cambios realizados.

54. Realice el cableado de los componentes como se indica en la Tabla 2 y Tabla 3 en la sección de Anexos. Antes de conectar la alimentación revisar detenidamente las conexiones, ya que el XADC solo pude recibir voltajes menores a 1V.

55. Conecte y encienda la FPGA, asimismo conecte el cable de programación. Abra la interfaz de LabVIEW provista y conecte el Xbee coordinador a la PC.

56. En Vitis de clic derecho sobre el archivo del proyecto y seleccione **Debug As→Launch on Hardware**.

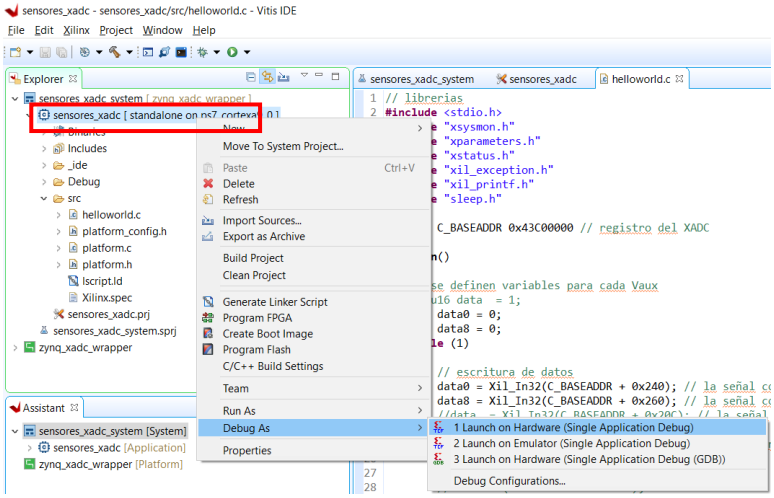

Ilustración 54 Ejecución del código y programación de la FPGA.

57. Sobre el código en C aparecerá una franja verde indicando que hasta ese punto se ha ejecutado, para continuar con la ejecución presione F8. Los datos se observarán en la interfaz de LabVIEW.

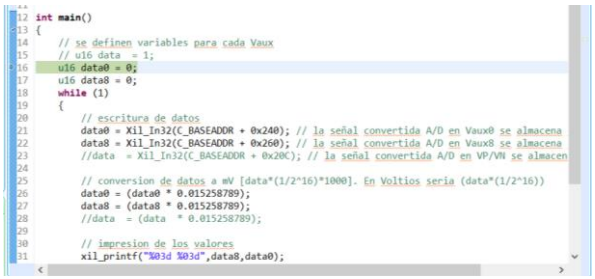

Ilustración 55 En espera para continuar la ejecución.

BIBLIOGRAFÍA

[1] Xilinx, «Xilinx,» 05 10 2016. [En línea]. Available: https://docs.xilinx.com/v/u/en-US/pg091-xadc-wiz. [Último acceso: 22 07 2022].

[2] Digilent, «Digilent,» 27 01 2014. [En línea]. Available: https://digilent.com/reference/programmable-logic/zedboard/reference-manual. [Último acceso: 15 06 2022].

[3] VENCO, «VENCO,» 12 03 2020. [En línea]. Available: https://www.vencoel.com/que-es-zigbee-como-funciona-y-caracteristicas-principales/. [Último acceso: 30 06 2022].

ANEXOS

Descripción del XADC

En la Ilustración 56 se observan los pines disponibles en el conversor, así como su correspondiente función. Mientras que, en la Ilustración 57 se presentan los valores de Offset a agregar al Base Address para la escritura de los datos correspondientes a cada uno de los canales del conversor.

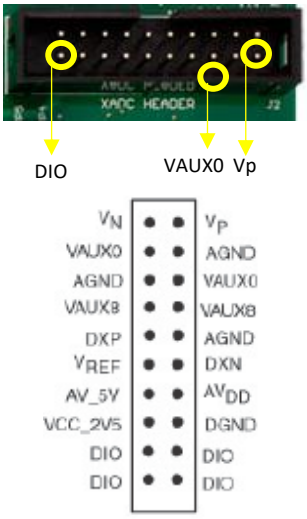

Ilustración 56 Pines disponibles del XADC.

Base Address + Offset (hex)	Register Name	Reset Value (hex)	Access Type	Description
C_BASEADDR + 0x240	$V_{AUXP}[0]$/ $V_{AUXN}[0]$	0x0	R[6]	The 12-bit MSB justified result of A/D conversion on the auxiliary analog input 0 is stored in this register.
C_BASEADDR + 0x244	$V_{AUXP}[1]$/ $V_{AUXN}[1]$	0x0	R[6]	The 12-bit MSB justified result of A/D conversion on the auxiliary analog input 1 is stored in this register.
C_BASEADDR + 0x248	$V_{AUXP}[2]$/ $V_{AUXN}[2]$	0x0	R[6]	The 12-bit MSB justified result of A/D conversion on the auxiliary analog input 2 is stored in this register.
C_BASEADDR + 0x24C	$V_{AUXP}[3]$/ $V_{AUXN}[3]$	0x0	R[6]	The 12-bit MSB justified result of A/D conversion on the auxiliary analog input 3 is stored in this register.
C_BASEADDR + 0x250	$V_{AUXP}[4]$/ $V_{AUXN}[4]$	0x0	R[6]	The 12-bit MSB justified result of A/D conversion on the auxiliary analog input 4 is stored in this register.
C_BASEADDR + 0x254	$V_{AUXP}[5]$/ $V_{AUXN}[5]$	0x0	R[6]	The 12-bit MSB justified result of A/D conversion on the auxiliary analog input 5 is stored in this register.
C_BASEADDR + 0x258	$V_{AUXP}[6]$/ $V_{AUXN}[6]$	0x0	R[6]	The 12-bit MSB justified result of A/D conversion on the auxiliary analog input 6 is stored in this register.
C_BASEADDR + 0x25C	$V_{AUXP}[7]$/ $V_{AUXN}[7]$	0x0	R[6]	The 12-bit MSB justified result of A/D conversion on the auxiliary analog input 7 is stored in this register.
C_BASEADDR + 0x260	$V_{AUXP}[8]$/ $V_{AUXN}[8]$	0x0	R[6]	The 12-bit MSB justified result of A/D conversion on the auxiliary analog input 8 is stored in this register.
C_BASEADDR + 0x264	$V_{AUXP}[9]$/ $V_{AUXN}[9]$	0x0	R[6]	The 12-bit MSB justified result of A/D conversion on the auxiliary analog input 9 is stored in this register.

Ilustración 57 Offset de los canales del XADC.

Configuración de módulos Xbee

Los principales parámetros para configurar se detallarán a continuación:

Pestaña	Parámetros	Descripción
Networking	**ID** PAN ID	Es un número cualquiera que representa al "nombre" de la red. Por ello, todos los módulos que se encuentren dentro de una misma red tendrán el mismo valor en este parámetro.
	JV Channel verification	Puede estar deshabilitado (0) o habilitado (1). Cuando un módulo NO trabaja como coordinador es requerido que se habilite.
	CE Coordinator enable	Al habilitarlo (1) el módulo trabajará como coordinador.
Addressing	**DH** Destination Address High	Este valor debe ser exactamente igual al **SH Serial number high,** en la pestaña Addressing, del mismo módulo**.**
	DL Destination Address Low	Este valor debe ser exactamente igual al **SL Serial number low,** pero del módulo con el que se quiere tener comunicación.
Sleep Modes	**SM** Sleep Mode	Aquí se define si el módulo trabajara como router (0) o como dispositivo final (1). Si el módulo trabajará como coordinador dejar el valor por defecto.

Tabla 1 Principales configuraciones de los módulos.

Código en C:

```
// librerias
#include <stdio.h>
#include "xsysmon.h"
#include "xparameters.h"
#include "xstatus.h"
#include "xil_exception.h"
#include "xil_printf.h"
#include "sleep.h"

#define C_BASEADDR 0x43C00000 // registro del XADC

int main()
{
	// se definen variables para cada Vaux

	u16 data0 = 0;

	while (1)
	{
		// escritura de datos
		data0 = Xil_In32(C_BASEADDR + 0x240); // la señal convertida A/D en Vaux0 se almacena en el registro


		// conversion de datos a mV [data*(1/2^16)*1000]. En Voltios seria (data*(1/2^16))
		data0 = (data0 * 0.015258789);

		// impresion de los valores
		xil_printf("%03d",data0);

		usleep(1000000); // retardo de 1s

	}
	return 0;
}
```

Conexión de los componentes

FPGA - JE	Xbee	Detalle
JE2	Rx	Cable blanco
JE3	Tx	Cable cafe
5 V	Vcc	Cable amarillo
GND	GND	Cable verde

Tabla 2 Conexiones FPGA - Xbee router.

a) b) c)

Ilustración 58 Conexiones a) FPGA-Xbee router, b) alimentación Xbee, c) Formato de los Pmod.

FPGA – Heder XADC	Sensores	Detalle
3	Emisor del fototransistor	Vaux0P, cable naranja
6	GND	Vaux0N, cable blanco
7	GND	Vaux8N, cable blanco
8	Salida del LM35	Vaux8P, cable rojo

Tabla 3 Conexiones Sensores - FPGA.

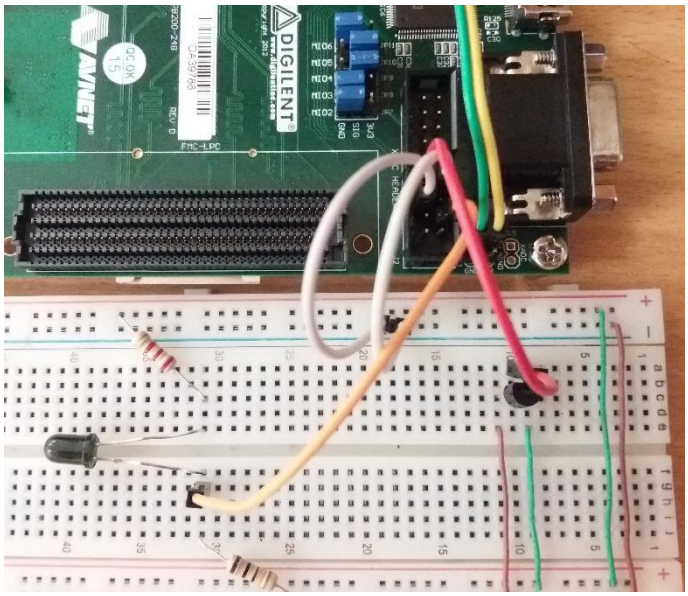

Ilustración 59 Conexiones sensores - FPGA.

PRACTICA #3

SISTEMA DE CÁMARA FPGA UTILIZANDO EL MÓDULO OV7670

OBJETIVOS

- Implementar un sistema de captura de imagen y salida VGA para visualización mediante el software VIVADO
- Crear bloques IP para poder controlar, el procesamiento de captura y transmisión VGA
- Conectar la FPGA mediante jumpers con el Módulo OV7670 a la par con una pantalla que contenga entrada VGA para el video.

MATERIALES Y HERRAMIENTAS:

Software Vivado 2019.2	18 jumpers
Tarjeta de Desarrollo ZYNQ 7000	1 cable VGA
Pantalla con entrada VGA	Fuente de poder 12V
Modulo OV7670	

Tabla 1 Materiales de la práctica

BREVE EXPLICACIÓN DE LA PRÁCTICA:

Para la práctica SISTEMA DE CÁMARA FPGA UTILIZANDO EL MÓDULO OV7670 se utilizan los archivos provistos configurados en Verilog, para crear un sistema de video-captura de imagen y transmisión de señal por puerto VGA. Esto mediante una FPGA Xilinx Zynq 7000, la cual procesa la información de forma paralela trabajando a la par con el módulo OV760 el cual es una cámara con presencia de 18 pines los mismos que irán conectados a las estructuras periféricas de la FPGA.

MARCO TEÓRICO:

Modulo OV7670

Es un módulo de cámara muy económico para el mercado por esta razón se utiliza de manera educativa, y con proyectos económicos, con presencia de 18 pines, trabaja por medio de la interfaz SCCB ya que presenta una señal de reloj y una de datos funcionando como un protocolo I2C, a la par una señal para controlar la información vertical y horizontal, con presencia de 8 pines digitales, el cual se alimenta con 3.3 V, debido a su entrada de 18 pines. [1]

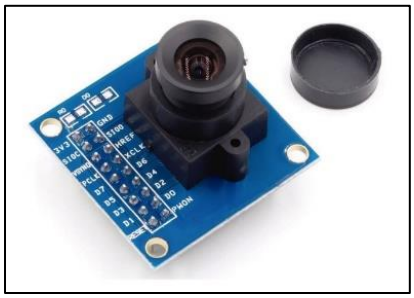

Ilustración 1 Modulo OV7670 [1]

FPGA

Una FPGA (Field Programmable Get Array), como su nombre lo indica es un arreglo programable de compuertas lógicas. Se caracteriza por realizar operaciones matriciales utilizando la lógica matemática programable mediante hardware. Su funcionamiento se da de forma paralela por lo que cada tarea a realizar se asigna a una parte del chip, permitiendo realizar varias tareas de forma simultánea [1].

Ilustración 2 FPGA

VERILOG

Es un lenguaje de programación en hardware, se utiliza en gran parte para la programación en FPGA (Field program gate arrage), creando circuitos lógicos y sus respectivas funciones, presenta características similares al lenguaje de programación en C, se utiliza las funciones For-While-if, en gran escala lo cual lleva una grata experiencia al trabajar y asimilar con otros lenguajes. [3]

VGA

Es una interfaz gráfica estándar que funciona con señales analógicas por lo cual su resolución de imagen no es de alta calidad, se conecta mediante une entrada física para visualizar la información como son el caso de pantallas-monitores-proyectores etc, dicha salida se puede ver tanto en color como en blanco y negro. Trabaja con una interfaz de 640*680 de resolución, en la actualidad sigue vigente, pero presenta una gran competencia como es el caso de la interfaz HDMI-DVI entre otras. [3]

Ilustración 3 Cable VGA [4]

INSTRUCCIONES DE LA PRÁCTICA:

1. Descargue la carpeta ov7670_to_vga-main proporcionada por el docente, de dicha carpeta extraiga todos los archivos.sv, cree una carpeta con el nombre PRACTICA3 en donde almacenara todos los elementos obtenidos anteriormente.
2. Proceda abrir el software VIVADO 2019.2 de XILINX, se aplasta la opción Create Project luego se despliega una pestaña donde se selecciona la opción Next, como se observa en la Ilustración 4.

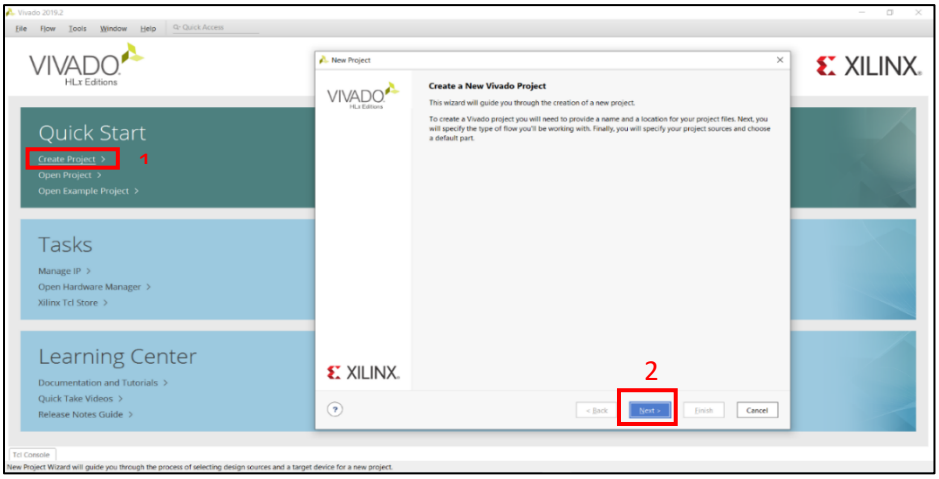

Ilustración 4 Interfaz inicio VIVADO

3. Se despliega una pestaña para seleccionar la dirección de la carpeta antes creada para ubicar el proyecto como se visualiza en la Ilustración 5.

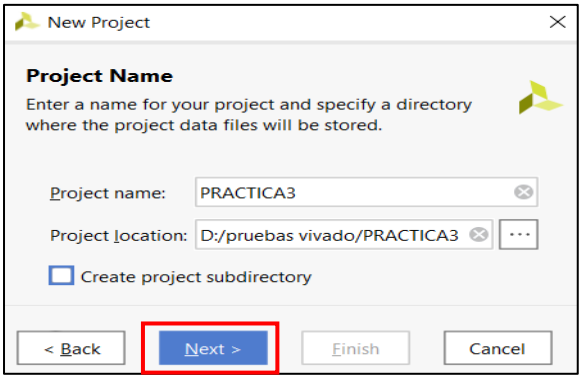

Ilustración 5 Project Name

4. Se procede a escoger la opción RTL Project y next en la Ilustración 6.

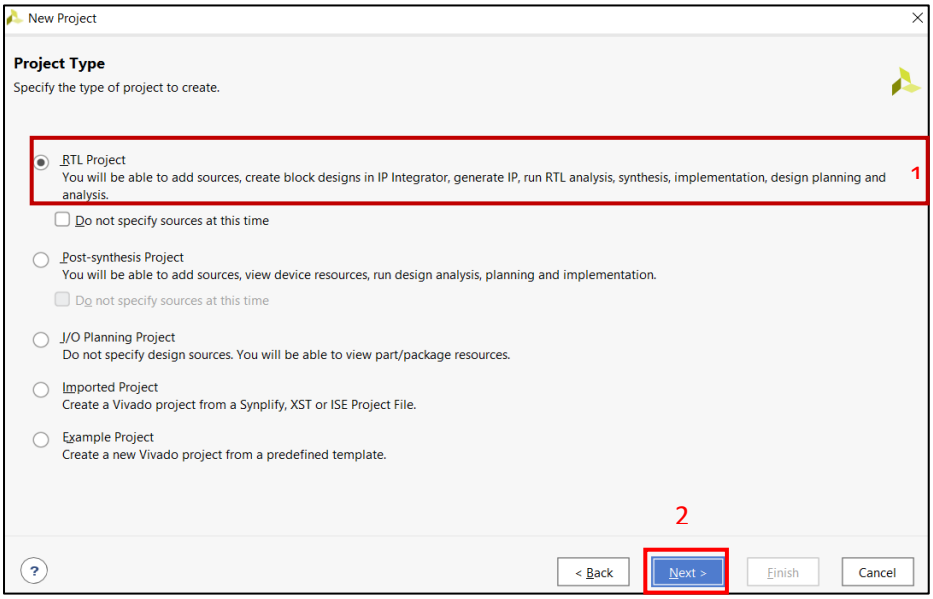

Ilustración 6 Selección de proyecto

5. Se despliega la pestaña Add Sources en donde se procede a seleccionar la opción Add Files y posterior Next como se aprecia en la Ilustración 7.

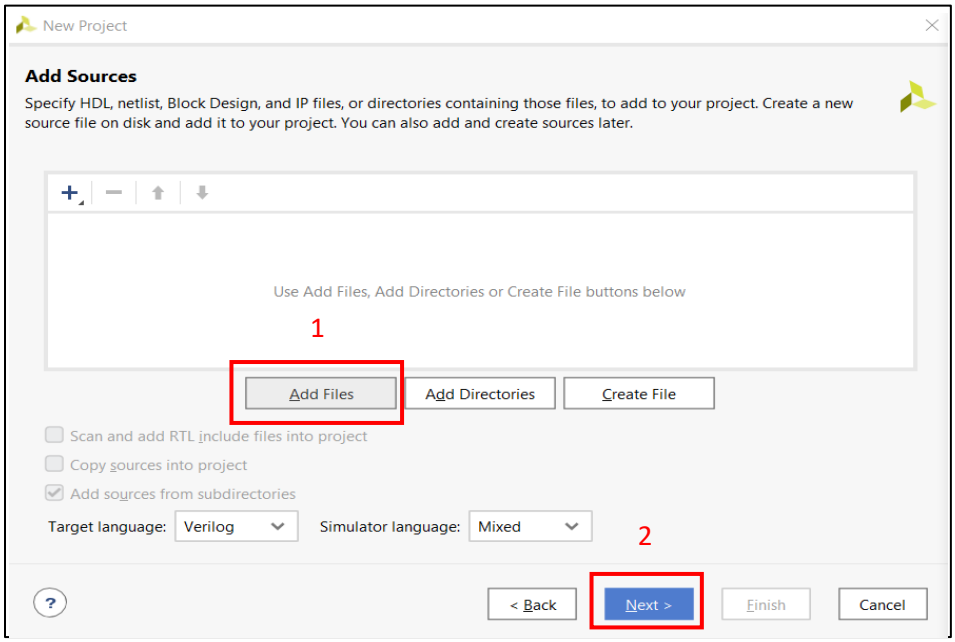

Ilustración 7 Adición de archivos

6. Se cargan los archivos.sv de la carpeta ov7670_to_vga-main proporcionada por el docente como se observa en la Ilustración 8.

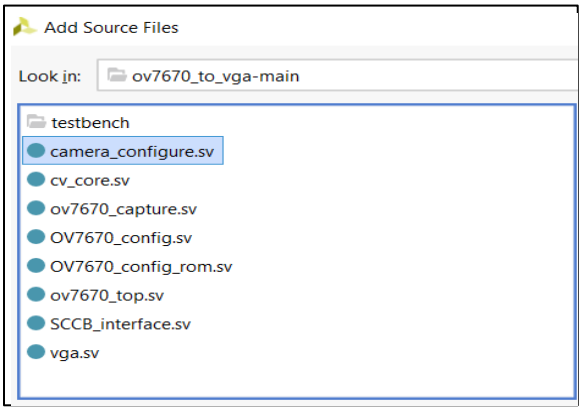

Ilustración 8 Selección de archivos.sv

7. Después que los archivos.sv se encuentren debidamente cargados se procede a dar Next en la Ilustración 9.

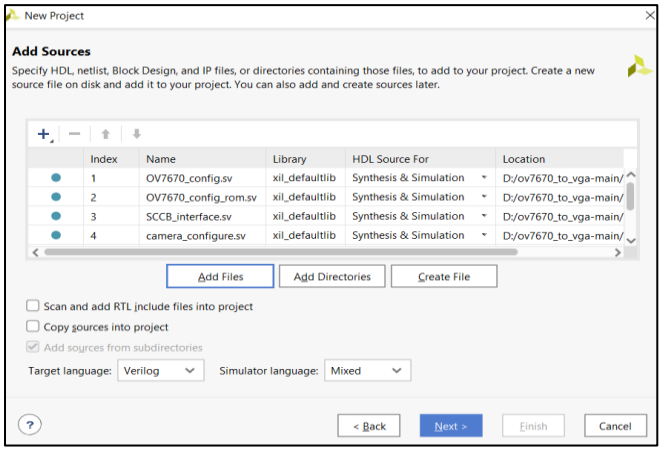

Ilustración 9 Archivos seleccionados en Add sources

8. Se despliega la pestaña Defaul Part como se observa en la Ilustración 10, se selecciona la familia Zylinq-7000, el modelo Xc7z020clg484-1 y se procede a dar Next.

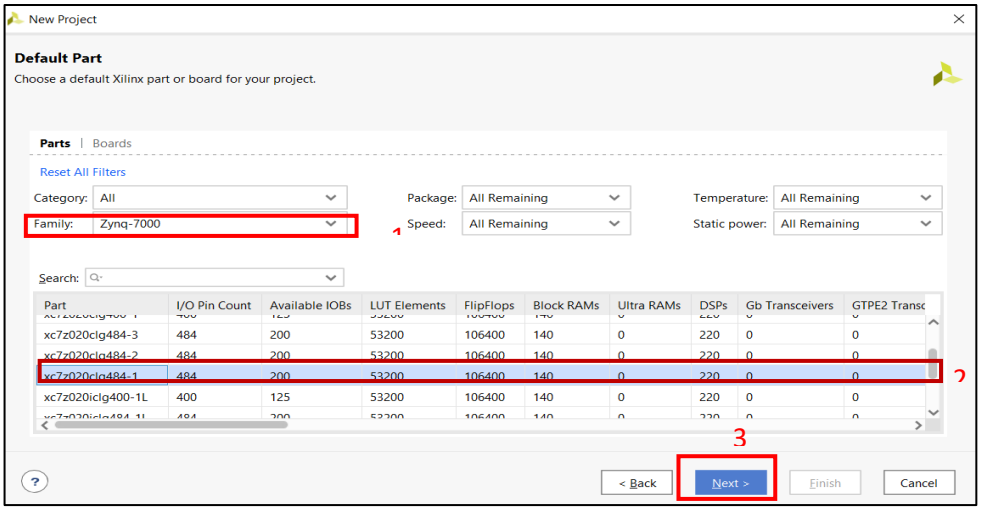

Ilustración 10 Elección de la placa FPGA

9. Se despliega la interfaz PROJECT MANAGER-PRACTICA 3 como se observa en la Ilustración 11 donde se aprecia los archivos.sv que se subió en la Ilustración 8.

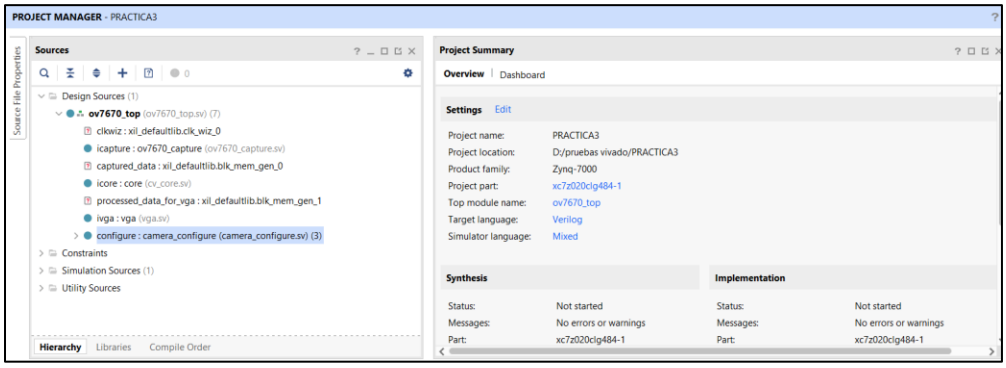

Ilustración 11 Interfaz PROJECT MANAGER de VIVADO

10. En la interfaz de vivado se procede a seleccionar IP Catalog y escoger Clocking Wizard como se observa en la Ilustración 12.

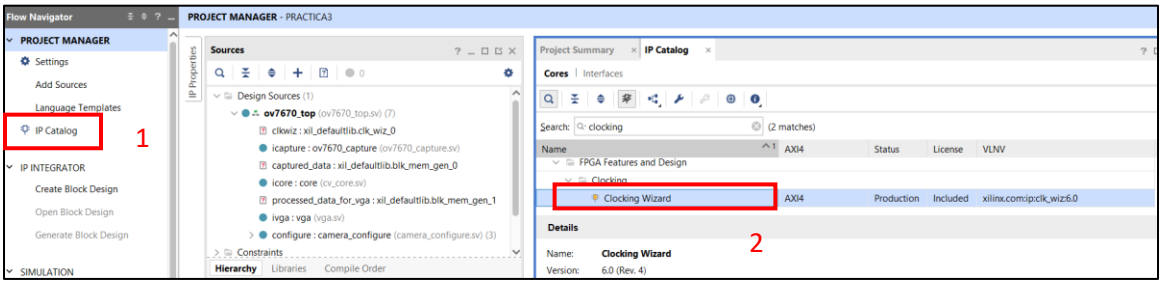

Ilustración 12 Creación del Bloque IP Clocking Wizard

11. Se despliega la pestaña Clocking Wizard (6.0) y se procede a seleccionar los cuadros con un visto como en la Ilustración 13.

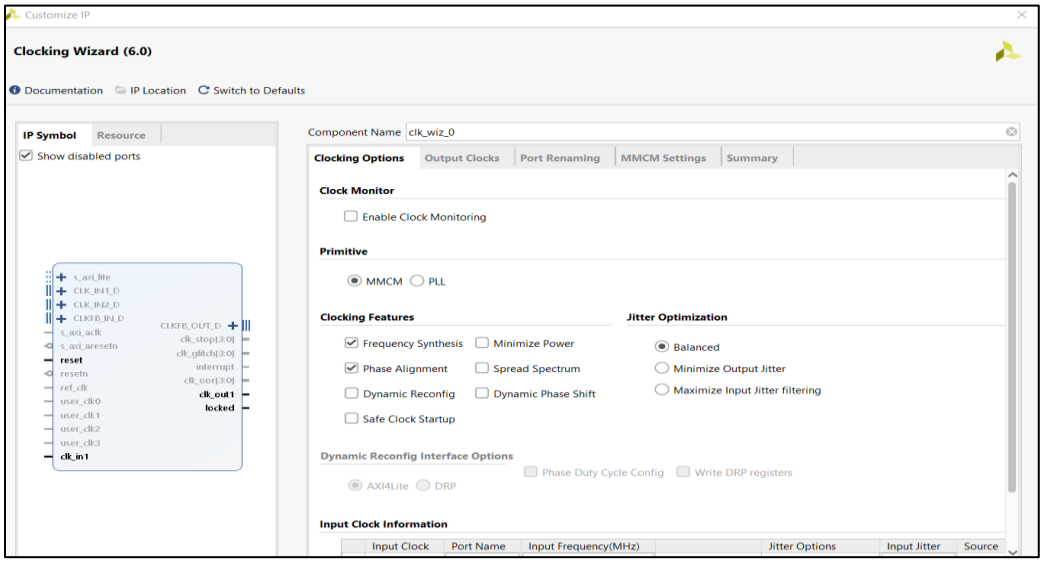

Ilustración 13 Selección del reloj

12. Se procede a ingresar las entradas y salidas de acuerdo con la Ilustración 14.

```
- Input Clock : name - clk_in_wiz        / frequency - 100MHz
- Output Clock1 : name - clk_48wiz       / frequency - 48MHz
- Output Clock2 : name - clk_48wiz_180shift     / frequency - 48MHz     / Phase - 180
- Output Clock3 : name - clk_24wiz       / frequency - 24MHz
- Output Clock4 : name - clk_24wiz_180shift     / frequency - 24MHz     / Phase - 180
```

Ilustración 14 Entradas y salidas del bloque IP clocking wizard

13. Como se observa en la Ilustración 15 se coloca el nombre de la señal y su respectiva frecuencia.

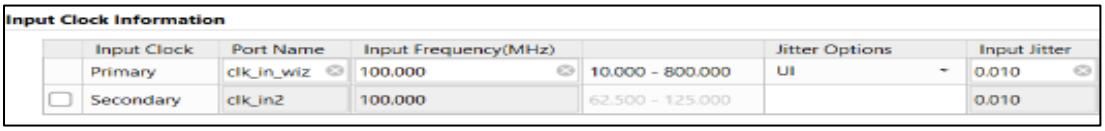

Ilustración 15 Señal de entrada (input clock): name clk_in_wiz

14. Proceda a implementar las señales de salida en base a la Ilustración 14 con su respectiva frecuencia y giro de fase si requiere, como se aprecia en la Ilustración 16.

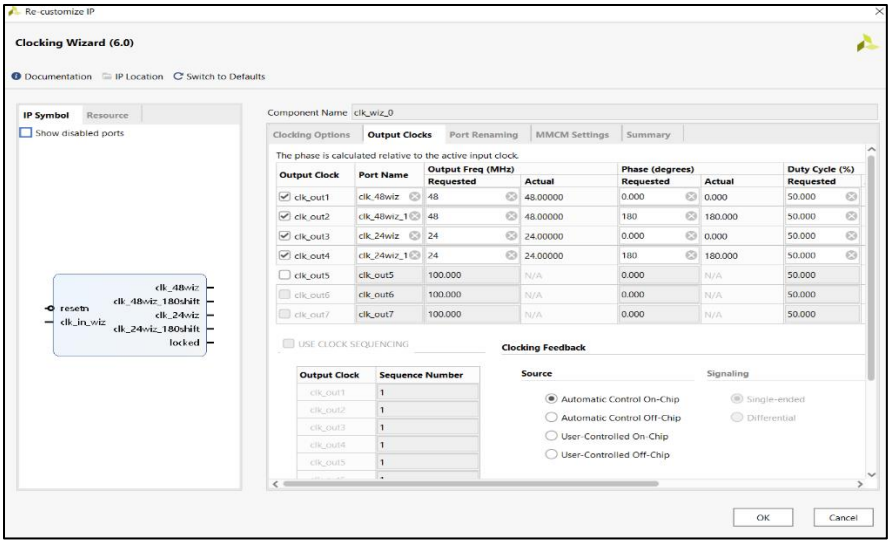

Ilustración 16 Señales de salida (Output Clocks)

15. Se debe escoger la opción Reset Type y seleccionar Active Low como se observa en la Ilustración 17.

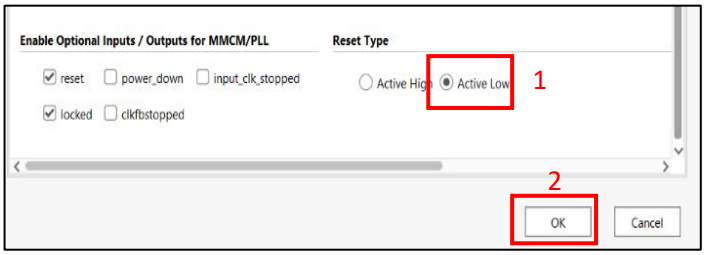

Ilustración 17 Ultimo paso para la creación del bloque IP

16. Se despliega la pestaña output products como se observa en la Ilustración 18 en donde se selecciona generate.

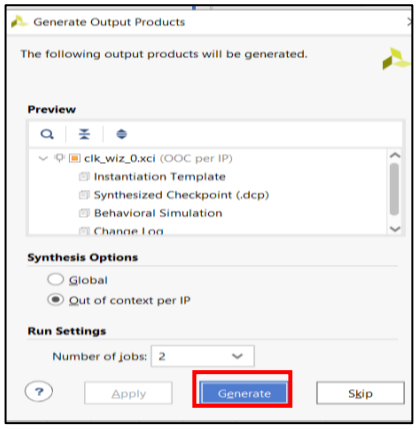

Ilustración 18 Generate Output Products

17. En la interfaz VIVADO se selecciona IP Catalog, y se utiliza Block Memory Generator como se observa en la Ilustración 19.

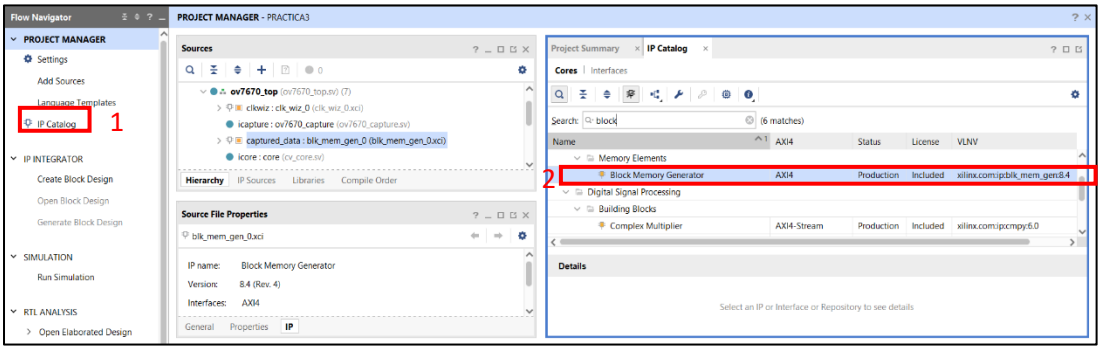

Ilustración 19 selección del bloque IP Block Memory Generator

18. Se despliega la pestaña Block Memory Generator (8:4). En memory Type se escoge la opción Simple Dual Port Ram como se aprecia en la Ilustración 20.

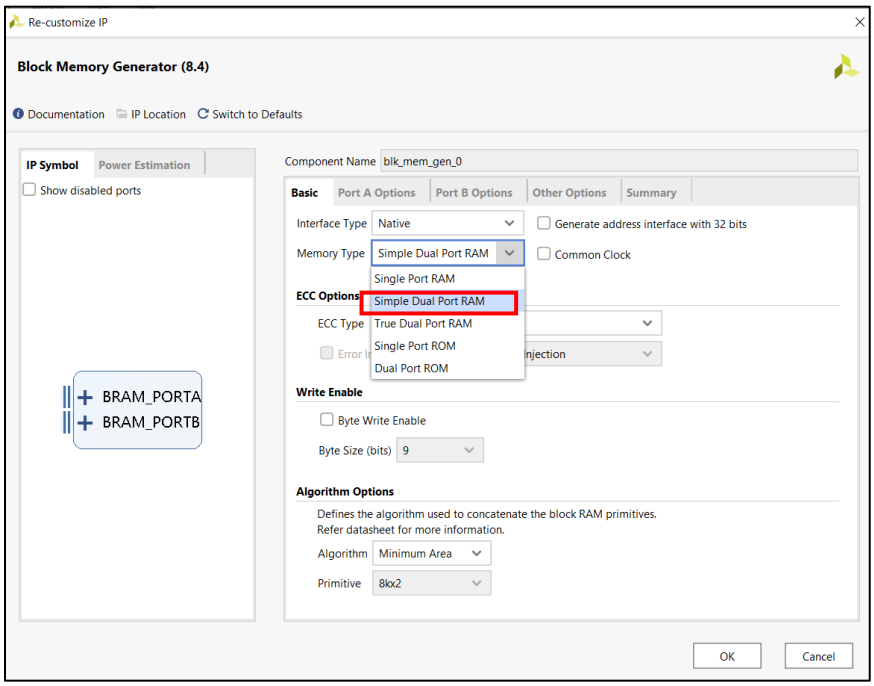

Ilustración 20 Simple Dual Port RAM

```
- Port A Options : Port A Width - 8 / Port A Depth - 307200 / Enable Port Type - Always Enabled
- Port B Options : Port B Width - 8 / Port B Depth - 307200 / Enable Port Type - Always Enabled
```

Ilustración 21 Port A Options y Port B options

19. En port A Width el rango de bits es 8, y en Port a Depth 307200 como se observa en la Ilustración 22 el enable port se debe escoger en modo Always Enabled.

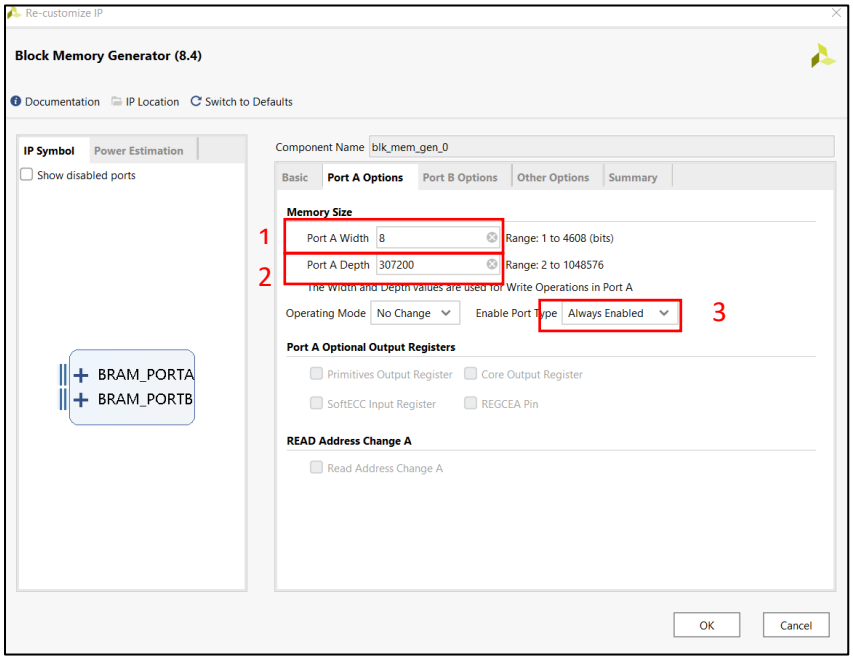

Ilustración 22 Port A options

20. Para el Port B options se realiza lo mismo que en el paso anterior, el primitives output registers no debe estar seleccionado, el enable port se debe escoger en modo always enabled como se aprecia en la Ilustración 23.

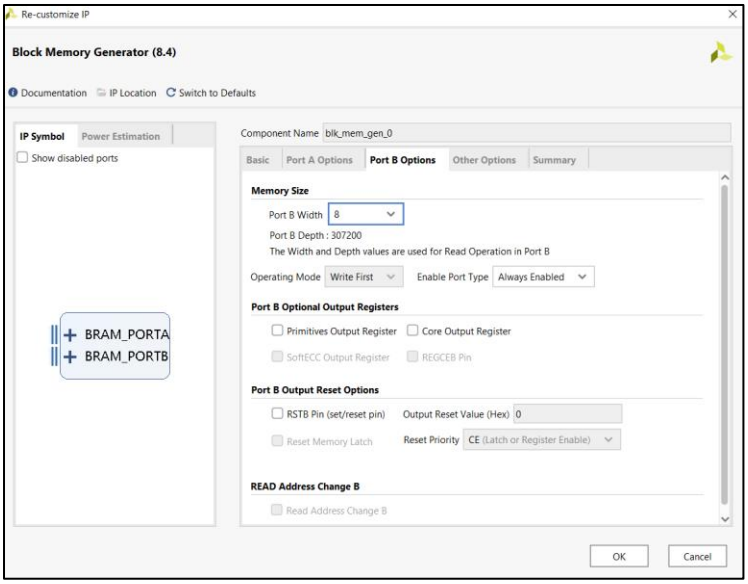

Ilustración 23 Port B options

Repita los pasos 17-18-19 y 20 en donde se crea un nuevo block memory generator (8:4) en este caso con otras características las cuales se aprecian en la Ilustración 24.

```
- Port A Options : Port A Width - 4 / Port A Depth - 307200 / Enable Port Type - Always Enabled
- Port B Options : Port B Width - 4 / Port B Depth - 307200 / Enable Port Type - Always Enabled
```

Ilustración 24 Port A Options y Port B Options

21. Se ce procede a escoger la opción Add Sources y posterior, Add or create constrains, como se observa en la Ilustración 25.

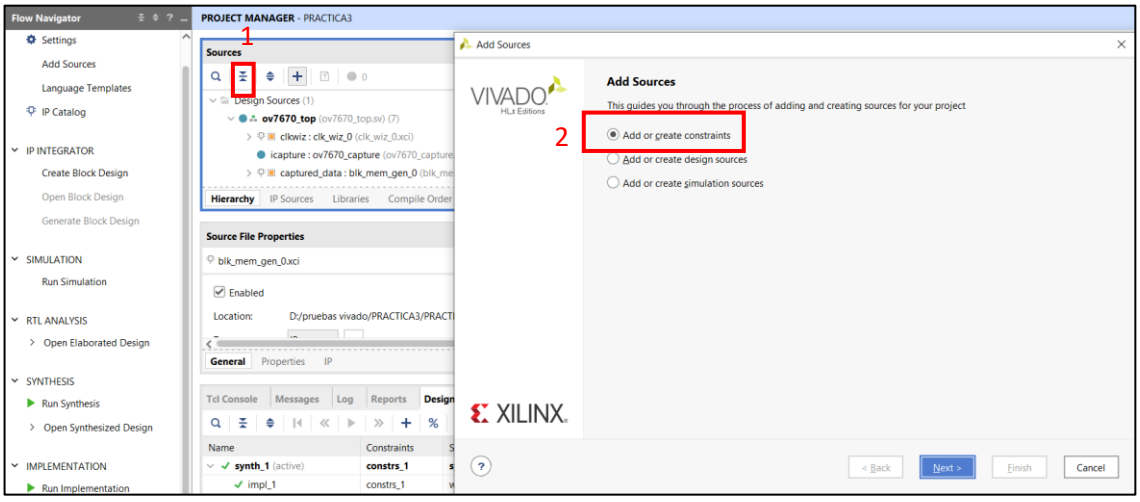

Ilustración 25 Creación de archivo Constraints

22. Se selecciona la opción Create File en donde se despliega la pestaña Add or Create Constraints-Create Constrains file-file name y se escribe cualquier nombre en este caso ov7670 como se observa en la Ilustración 26.

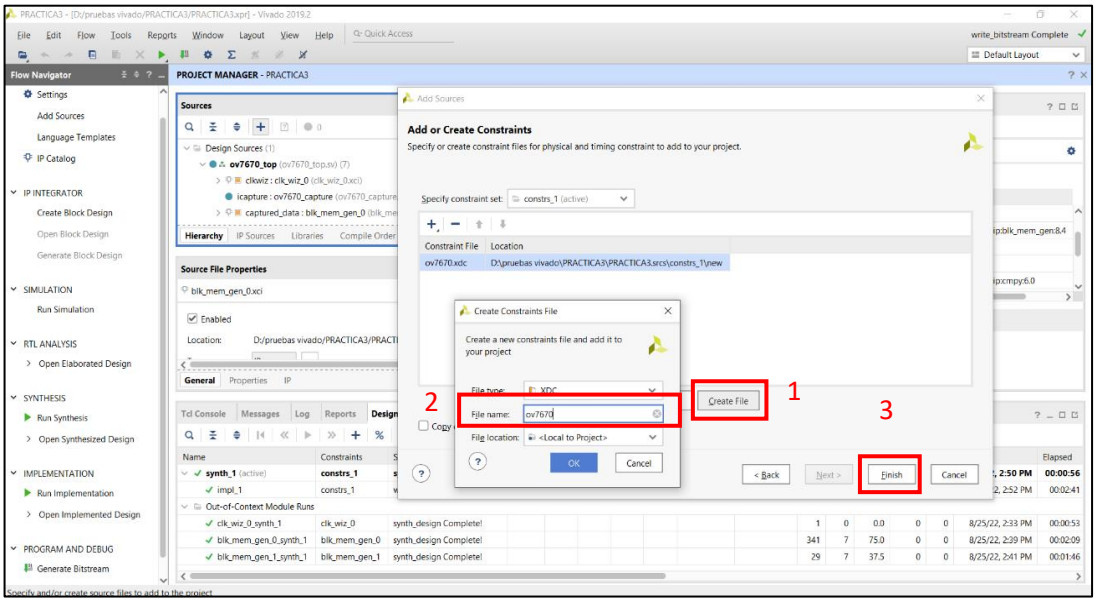

Ilustración 26 File Name OV7670

23. Se despliega el menú principal de vivado PROJECT MANAGER-PRACTICA3 en archivos constrains en donde ya está creado ov7670.xdc como se observa en la Ilustración 27.

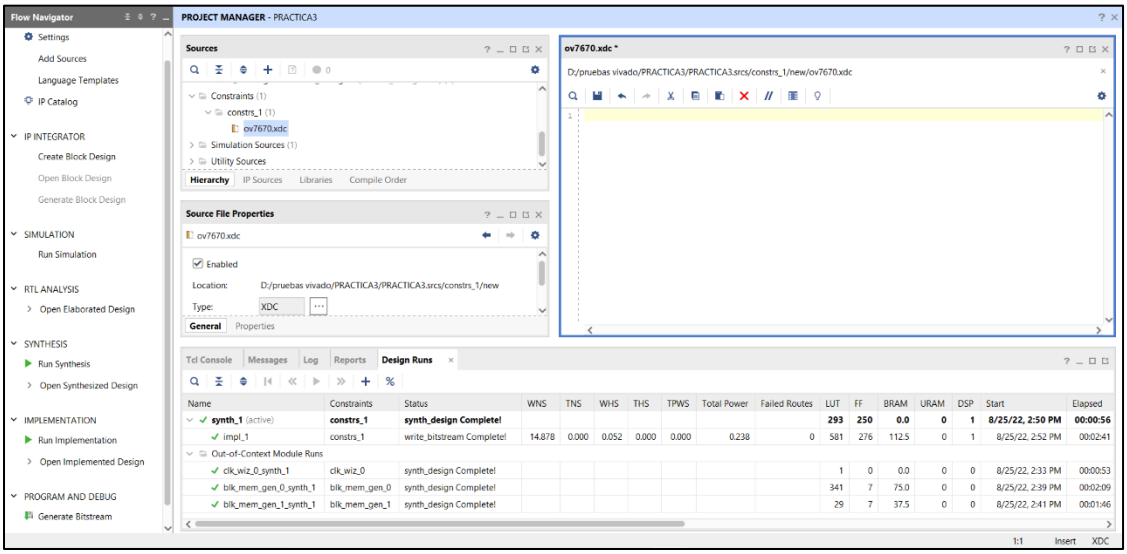

Ilustración 27 Archivo ov7670.xdc

24. De la carpetea ov7670_to_vga-main proporcionada por el docente se copia el archivo.txt en donde se encuentran las entradas y salidas de la FPGA en donde se procede a pegar en ov7670.xdc de la Ilustración 27.Dando como resultado la Ilustración 28, para ello se tiene que basar en las conexiones especificas entre la cámara y la FPGA como se aprecia en la Tabla 2.

MODULE OV7670	FPGA
3.3V	JB6
GND	JB5
SCL	JD4P
SDA	JD2P
VS	JD3P
HS	JD1P
PCLK	JC4P
MCLK	JC2P
D7	JC3P
D6	JC1P
D5	JB10
D4	JB4
D3	JB9
D2	JB3
D1	JB8
D0	JB2
RESET	JB7
PWDN	JB1

Tabla 2 Conexiones MODULE OV7670-FPGA

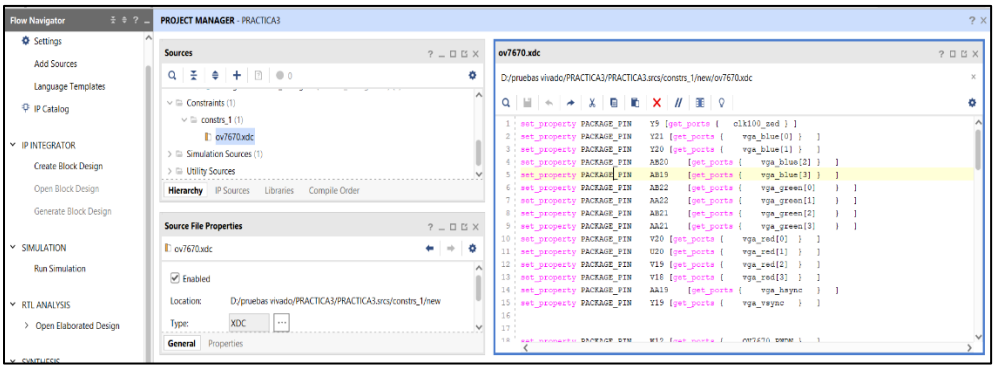

Ilustración 28 ov7670.xdc completo

25. Después de realizar todos los pasos anteriores se procede a seleccionar la opción PROGRAM AND DEBUG-Generate Bitstream-Open Hardware Manager-Open Target como se observa en la Ilustración 29.

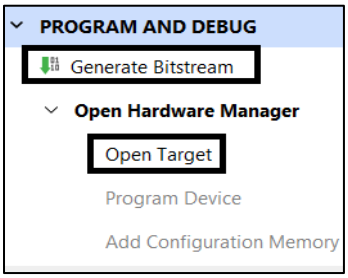

Ilustración 29 Generate Bitstream-Open Target

26. Al prender la FPGA se despliega la opción en la pestaña superior Program device. Se selecciona y se da a Program como se aprecia en la Ilustración 30.

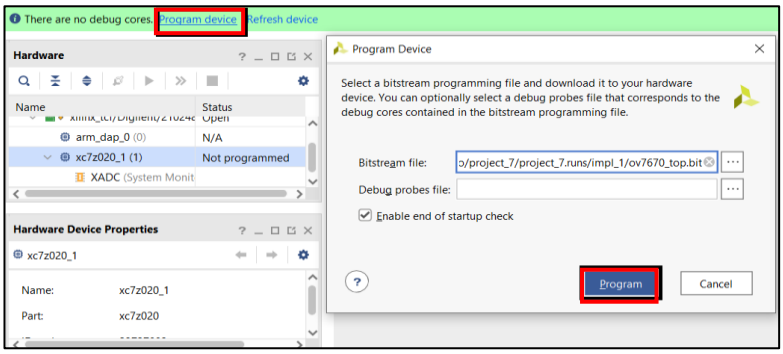

Ilustración 30 Programar FPGA

27. Al finalizar todos los pasos la fpga se debe encontrar en modo Programmed y Open como se observa en la Ilustración 31

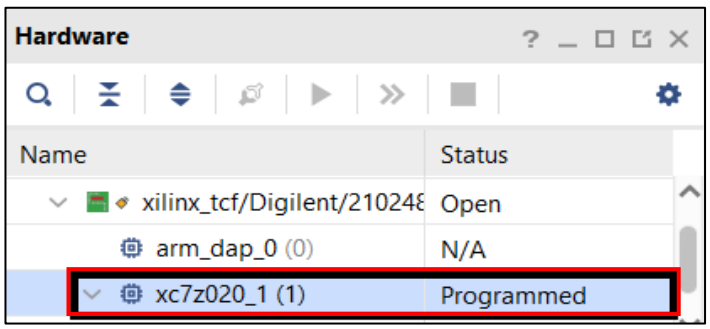

Ilustración 31 FPGA programada

BIBLIOGRAFIA

[1] Novatronic, "Novatronic," [Online]. Available: https://novatronicec.com/index.php/product/modulo-de-camara-ov7670-vga/. [Accessed 08 09 2022].

[2] National Instruments, "ni.com," 08 2018. [Online]. Available: https://www.ni.com/es-cr/innovations/white-papers/08/fpga-fundamentals.html. [Accessed 07 2022].

[3] Peanut-JR, "github," 08 11 2021. [Online]. Available: https://github.com/ESCA-RISC-V/ov7670_to_vga/blob/main/README.md. [Accessed 08 09 2022].

[4] Tarjtetas graficas PC, "Tarjetas graficas PC," [Online]. Available: https://tarjetasgraficaspc.com/monitores/que-es-vga/. [Accessed 08 09 2022].

[5] Mercado libre, "Mercado libre," [Online]. Available: https://articulo.mercadolibre.com.ec/MEC-518681615-cable-vga-15m-_JM#redirectedFromParent. [Accessed 08 09 2022].