TESIS DE DOCTORADO

Mención procesamiento señales e imágenes

“Desarrollo de un sistema de adquisición ad-hoc para el control y medición remota de magnitudes físicas en vehículos no tripulados”

Autor: Ing. Adrián G. Stacul

Director de Tesis: Dr. Ing. Mario B. Lavorato

Co-Director de Tesis: Ing. Daniel A. Pastafiglia

Buenos Aires – 2019

UTN.BA

UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

"*...Es curioso, a mí me interesa mucho más hablar, o estar en contacto con la gente que piensa lo contrario de lo que yo pienso, que de los que piensan lo mismo que pienso yo*".

Salvador Dalí.

A mis amigos y familia, a mis compañeros de trabajo y de la vida, a la ciencia y el arte...

AGRADECIMIENTOS

Si bien, aún estoy corrigiendo las ultimas cuestiones sobre esta tesis, me he tomado el tiempo de distraerme un rato y escribir estos párrafos y pensar en cada una de las personas que estuvo a mi lado ayudándome en este camino.

La lista es extensa, ya que muchas personas han colaborado de manera directa e indirecta con esta escritura, y desde hace varios años. Sin duda las primeras personas que me gustaría agradecer es a mi director Mario y mi codirector Daniel, ya que han luchado (y siguen) por muchas cuestiones políticas que han sucedido en el instituto en estos últimos años, y me han animado a seguir adelante tanto en buenos momentos como en tiempos difíciles.

También quiero agradecer a Edgardo Comas, Sergio Saluzzi, Martín Morales, Sebastián Alvarez, Gerardo García y Ariel Dalmas Di Giovanni que conviven conmigo en CITEDEF, de los que aprendo día a día y me han tenido una especial paciencia y atención.

Por último y no por ello menos importantes, quiero agradecer su apoyo y comprensión a: mi pareja, a mis padres, a mis amigos de toda la vida, y al resto de toda mi familia.

TABLA DE CONTENIDOS

Agradecimientos iii

Tabla de contenidos vi

Lista de ilustraciones ix

Lista de tablas xiii

INTRODUCCIÓN 1

Antecedentes y fundamentos. 1

Justificación del problema 2

Estado del arte 5

Objetivos 6

Metodología 7

CAPITULO 1 - SISTEMAS DE TELEMETRÍA 9

1.1. Descripción general de un sistema de telemetría 9

1.2. Hardware de a bordo 12

1.2.1. Sensores 12

1.2.2. Modulación 13

1.2.3. Conmutación 16

1.2.4. Patrones de sincronización de datos 17

1.3. Puesto de tierra 18

1.3.1. Reconstrucción de la trama PCM 18

1.3.2. Sincronización de tramas 20

1.3.3. De-conmutación 22

1.3.4. Simulación y codificación 23

1.3.5. Procesamiento en tiempo real 23

1.3.6. Graficadores de tiempo real 23

1.3.7. Gestión de archivos 23

1.3.8. Distribución de datos 23

1.4. Fin del capítulo 24

CAPÍTULO 2 - TECNOLOGÍA FPGA 26
2.1. Conceptos fundamentales 27
2.1.1 PLA 28
2.1.2. PAL 30
2.1.3. CPLD 32
2.1.4. FPGA 32
2.2. Arquitectura de los dispositivos FPGA 33
2.2.1. Bloques lógicos configurables 34
2.2.2. Bloques de entrada/salida 35
2.2.3. Interconexiones 36
2.3. Formas de desarrollo e implementación 37
2.3.1. Lenguaje esquemático 37
2.3.2. Lenguajes de descripción de hardware 38
2.3.3. Suite de desarrollo ISE 41
2.3.4. Flujo de implementación para ISE 43
2.3.5. Librería System Generator para FPGAs Xilinx 44
2.4. Placas de desarrollo Xilinx 48
2.4.1. Atlys 48
2.4.2. 3PX1 50
2.4.3. Análisis comparativo entre las tarjetas Xilinx 51
2.5. Fin del capítulo 52
CAPITULO 3 – DISEÑO Y DESARROLLO DEL SISTEMA ADQUISIDOR 54
3.1. Simulador PCM 57
3.1.1. Características Generales 58
3.1.2. Ventajas y limitaciones 59
3.1.3. Bloques fundamentales 60
3.1.3.1. Enable Generator 61
3.1.3.2. Clock Generator 63

3.1.3.3. Buffer 64
3.1.3.4. Byte to Bits 65
3.2. Adquisidor PCM 66
3.2.1. Ventajas y limitaciones 68
3.2.2. Bloques fundamentales 69
3.2.2.1. Clock Sync 70
3.2.2.2. ReSync 71
3.2.2.3. Store 72
3.2.2.4. Send 74
3.3. Ensayos en hardware 78
3.4. Implementación del filtro digital para PCM 82
3.4.1. Filtro PCM 84
3.4.2. Teoria del filtro CIC 84
3.4.3. Implementación del filtro CIC 87
3.5. Verificaciones del sistema 91
3.6. Resultados 97
CONCLUSIONES 109
BIBLIOGRAFÍA 115
ANEXOS 118
4.1. Formato de Trama PCM 118
4.2. Configuración de la FPGA 120
4.3. Bloques fundamentales del System Generator 122
4.4. Código Frame Generator 127

LISTA DE ILUSTRACIONES

Figura 1. Sistema de telemetría 10
Figura 2. Sistema de adquisición de a bordo 10
Figura 3. Flujo de mediciones en formato TDM 11
Figura 4. Diagrama en bloques de un Puesto de Tierra 12
Figura 5. Circuito puente básico de galgas extensiométricas 13
Figura 6. Diferentes tipos de modulación 14
Figura 7. Ejemplo de señal PDM. Extraído de https://en.wikipedia.org/wiki/Pulse-density_modulation 14
Figura 8. Códigos de datos PCM 15
Figura 9. Conmutación básica con sincronismo de trama 16
Figura 10. Esquema general de una super-conmutación 17
Figura 11. Efectos en una señal PCM. Extraído de (Halsall, 1995) Figura 2.6, Pág. 32 19
Figura 12. Rendimiento teórico de probabilidad de error de bits de para diversas técnicas en PCM (suponiendo una perfecta sincronización de bits). Extraído de (Telemetry Standars, 2015). 20
Figura 13. Estados en la sincronización de tramas (los números en violeta indican la cantidad de tramas válidas que deben estar presentes para poder pasar al próximo estado, los celestes indican números de tramas invalidas) 21
Figura 14. Esquema general de un PLD visto como caja negra 28
Figura 15. Estructura general de un PLA 29
Figura 16. Diagrama a nivel de compuertas de un PLA 30
Figura 17. Ejemplo de implementación en una PAL cumpliendo una función determinada 31
Figura 18. Estructura genérica de un CPLD 32
Figura 19. Estructura general de un FPGA 33
Figura 20. Circuito para una LUT de dos entradas 34
Figura 21. Bloque CLB 35
Figura 22. Organización de LUTs y CLBs en la familia Spartan-6 de Xilinx 35
Figura 23. Esquema simplificado de un IOB de la familia Spartan de Xilinx 36
Figura 24. Enrutamiento en una FPGA Spartan de Xilinx 37
Figura 25. Niveles de abstracción entre Verilog y VHDL. Extraído de (Maxfield, 2004). 41
Figura 26. Flujo de diseño utilizando ISE. Extraído de (Xilinx Inc., s.f.) 42
Figura 27. Vista de un proyecto utilizando la Suite ISE de Xilinx. Extraído de (Xilinx Inc., s.f.). 43
Figura 28. Flujo de diseño utilizando las herramientas de System Generator 46

Figura 29. Vista de los bloques que componen la librería System Generator de Xilinx. Extraído de (Xilinx Inc., s.f.). 47
Figura 30. Concepto de los Gateways de System Generator 47
Figura 31. Placa de desarrollo Atlys con una FPGA Spartan-6 de Xilinx 49
Figura 32. Esquema de puertos disponibles en un kit Atlys 50
Figura 33. Placa de desarrollo 3PX1 de la firma Emtech que utiliza una Spartan 6 de Xilinx 50
Figura 34. Arquitectura de la plaqueta 3PX1. Extraído de (Emtech, s.f.) 55
Figura 35. Diagrama de componentes electrónicos de la plaqueta 3PX1. Extraído de (Emtech, s.f.) 56
Figura 36. Vista del kit 3PX1 56
Figura 37. Conectores externos para el desarrollo en el kit FPGA 56
Figura 38. Diagrama en bloques del simulador 60
Figura 39. Desarrollo del sub-sistema Enable Generator 62
Figura 40. Salida "Out Clock Enable" 62
Figura 41. Salidas del sub-sistema “Enable Generator” 63
Figura 42. Datos y "Clock" sincrónicos 64
Figura 43. Modelo del sub-sistema del "Clock Generator" 64
Figura 44. Entradas y salida del "Clock Generator" 64
Figura 45. Modelo del sub-sistema "Buffer" 65
Figura 46. Modelo del sub-sistema "Byte to Bits" 66
Figura 47. Señal original transmitida (a) y Señal recibida reducida en ancho de banda con ruido adicional (b) 67
Figura 48. Principio de funcionamiento del sincronizador 67
Figura 49. Señal ruidosa sobre muestreada a 4 y 10 veces la tasa de bit 68
Figura 50. Diagrama en bloques del adquisidor 69
Figura 51. Modelo del sub-sistema "Clock Sync" 70
Figura 52. Simulación del sub-sistema "Clock Sync" 71
Figura 53. Modelo del sub-sistema "ReSync" 72
Figura 54. Resultados del sub-sistema "ReSync" 72
Figura 55. Modelo del sub-sistema "Store" 73
Figura 56. Cuadro de configuración del bloque FIFO 74
Figura 57. Resultados de la simulación del sub-sistema "Store" 74
Figura 58. Modelo del sub-sistema "Send" 75
Figura 59. Resultados de la simulación del sub-sistema "Send" 75
Figura 60. Medición del tiempo de bit del Transmisor serie a 460800 bps 76

Figura 61. Medición del tiempo de bit del Transmisor serie a 230400 bps 77
Figura 62. Medición del tiempo de bit del Transmisor serie a 115200 bps 77
Figura 63. Señales resultantes de la simulación del sistema simulador PCM 78
Figura 64. Resultados del "Place and Route" del simulador PCM 79
Figura 65. Resultados de tiempos del "Place and Route" del simulador PCM 79
Figura 66. Medición con osciloscopio del simulador PCM 80
Figura 67. Medición de la estabilidad del "Clock PCM" 81
Figura 68. Medición lógica de los canales de Datos y Clock PCM 82
Figura 69. Implementación del modelo completo 82
Figura 70. Valores de "Noise generator" 83
Figura 71. Modelo para la medición de SNR 83
Figura 72. Medición del SNR a la salida del simulador PCM 84
Figura 73. Diagrama básico de un integrador 85
Figura 74. Diagrama básico de un comb 85
Figura 75. Filtro CIC de 3 etapas 86
Figura 76. Modelo del filtro PCM implementado 88
Figura 77. Análisis espectral de la señal PCM ideal del simulador 89
Figura 78. Análisis espectral de la señal PCM del simulador con ruido gaussiano adicionado ... 89
Figura 79. Análisis espectral de la señal PCM a la salida del filtro CIC 90
Figura 80. Análisis espectral de la señal PCM reconstruida 90
Figura 81. Medición de la respuesta del filtro PCM en función del tiempo 91
Figura 82. Modelo final de prueba 91
Figura 83. Imagen de la matriz de datos PCM con escala de colores HSV 92
Figura 84. Imagen de la matriz de la carga útil con escala de colores HSV 93
Figura 85. Mapa de errores encontrados 94
Figura 86. Probabilidad de ocurrencia de errores en el sistema de adquisición 95
Figura 87. Errores acumulados totales en la adquisición 95
Figura 88. Banco de pruebas del laboratorio 98
Figura 89. Generador/Simulador de datos PCM implementado en el kit de desarrollo Atlys de Digilent 98
Figura 90 Conexión a través de una única línea de datos entre el Simulador implementado con el kit de desarrollo Atlys de Digilent y el adquisidor implementado en el kit 3PX1 de Emtech 99
Figura 91. Análisis del adquisidor con un analizador lógico 100
Figura 92. Medición de la marca de muestreo y el reloj regenerado. 100
Figura 93. Medición del tiempo de detección de la palabra de sincronismo 101

Figura 94. Medición del período entre detecciones de sincronismo 101
Figura 95. Medición del clock regenerado 102
Figura 96. Medición de la trama UART 102
Figura 97. Verificación de tramas. El simulador se conecta directamente al adquisidor. 103
Figura 98. Plaqueta generadora de datos PCM con sensores integrados para aplicación con UAVs 104
Figura 99. Adición de ruido en los datos PCM. 105
Figura 100. Verificación de tramas. La plaqueta de abordo con tramas simuladas y sin presencia de ruido se conecta al adquisidor. 106
Figura 101. Verificación de tramas. La plaqueta de abordo con tramas simuladas y con la presencia de ruido se conecta al adquisidor. 106
Figura 102. Verificación de tramas. La plaqueta de abordo con tramas simuladas y datos pseudo-aleatorios y con presencia de ruido se conecta al adquisidor. 107
Figura 103. Verificación de tramas. La plaqueta de abordo con datos reales de sus sensores internos y con presencia de ruido se conecta al adquisidor. 107
Figura 104. Bloques System Generator de Xilinx para Simulink 122

LISTA DE TABLAS

Tabla 1. Patrones de sincronismo recomendados por el estándar IRIG106 17
Tabla 2. Parámetros en función de la velocidad de transmisión 75
Tabla 3. Tiempos mínimos y máximos de bit para transmisión serie 76
Tabla 4. Versión de software para síntesis y dispositivo FPGA 78
Tabla 5. Formato de datos de la Trama PCM 118
Tabla 6. Detalle del canal de IMU 119
Tabla 7. Pines del modelo asignados a puertos físicos del 3PX1 120

INTRODUCCIÓN

ANTECEDENTES Y FUNDAMENTOS.

Desde el año 2009 en el Departamento de Electrónica Aplicada que pertenece a CITEDEF (Instituto de Investigaciones Científicas y Técnicas para la Defensa), se realizan proyectos de I+D asociados a vectores y vehículos no tripulados (tales como cohetes sonda y sistemas asociados a vectores de aplicaciones civiles y militares), obteniéndose y procesándose las señales y datos de todos los sensores incorporados al vehículo. Además, se adquieren y procesan señales y datos de interés provenientes de otros experimentos científicos que dichos vehículos llevan a bordo. Toda esta información permite obtener, en tiempo real, las coordenadas y trayectoria del vehículo, su "status" interno y todas las variables y parámetros del medio ambiente que rodea al mismo. Antiguamente se trabajaba con el estándar RS232 para la navegación y monitoreo del vehículo, utilizar este tipo de sistemas, en la actualidad, implicaría una limitación importante en la transferencia de datos. Hoy en día esta limitación se supera trabajando sobre una red Ethernet, es por ello que muchas estaciones de monitoreo en tierra, cuentan con esta arquitectura de red a fin de adquirir dichas tramas digitales y realizar un procesamiento en tiempo real.

En el año 2014, surge la necesidad por parte de CITEDEF de realizar un sistema de adquisición de datos serie para la medición remota de magnitudes físicas en sistemas de sondeo no tripulados (cohetes sonda) y en vehículos en general, de tal manera que el sistema permita procesar señales de alta velocidad, con una baja tasa de pérdidas de datos; y así poder manejar tramas digitales de mayor tamaño y redistribuirlas por una red Ethernet.

La aplicación de estos sistemas de adquisición en la transmisión y recepción de datos, ha permitido, no solamente el incremento de la tasa efectiva de datos y el ancho de banda en la trasmisión de información desde el vehículo y hacia el puesto de monitoreo remoto (en forma bidireccional), sino también en la implementación de nuevos algoritmos de procesamiento de señales e imágenes que logran una navegación y control más precisos, lo que los posiciona en un lugar de privilegio en los tópicos más avanzados de la ingeniería aeroespacial actual y toma una mayor dimensión al querer adquirir tramas de mayor tamaño en vehículos más complejos y que contengan mayor cantidad de experimentos científicos a bordo.

Este tipo de Sistema Telemétrico de Adquisición de Datos con un muy alto grado de procesamiento de señales en tiempo real no está desarrollado en el país. Son muy escasos los grupos de Investigación y Desarrollo que participan en el desarrollo este tipo de Sistema y por lo general utilizan material importado y sensible para su exportación. El aporte conceptual de esta Tesis será aplicable a diferentes áreas del conocimiento que requieren la integración de este tipo de sistemas

tales como Sistemas de seguimiento de UAVs (Unmanned Aerial Vehicle, Vehículo aéreo no tripulado), Sistemas de monitoreo de cohetes sondas, entre otros. Se debe destacar la importancia en el desarrollo de una estructura de hardware y software como la propuesta para lograr una integración completa en todas las etapas de un vuelo o lanzamiento. Cabe aclarar que el tipo de Sistema de Adquisición propuesto posee características que no se han encontraron en los manuales de los productos internacionales de marcas de primer nivel.

JUSTIFICACIÓN DEL PROBLEMA

La meta principal del trabajo propuesto se basa en: Realizar un estudio de algoritmos ad-hoc, un desarrollo conceptual y diseño del hardware asociado, la obtención del código para su implementación y la simulación computacional. Para finalmente poder implementar en campo un sistema de adquisición de datos sincrónicos de alta velocidad para medición remota de magnitudes físicas aplicado a unidades y plataformas con movimientos en el espacio.

En la actualidad existen sistemas desarrollados que no son autónomos poseen un complejo algoritmo para la adquisición de datos inaccesible al usuario, el sistema propuesto se destaca ya puede conectarse en cualquier computadora a través de un simple puerto USB y además el usuario puedo modificar el modelo de adquisición adaptándose a sus propios requerimientos. Los fabricantes de este tipo de unidades no permiten definir ni se puede tener acceso a ajustes y/o posibles modificaciones de sus características constitutivas. Es decir que no se pueden definir sus parámetros ni modificar o ajustar sus variables. Estos sistemas se encuentran empaquetados (llave en mano) y solo se accede, en algunos casos, a través de costosas licencias y otras veces solo el fabricante realiza dichas mejoras. Esto representa un grave problema para la construcción de un equipo con un sistema de adquisición y monitoreo de datos llave en mano, por ello, el sistema de adquisición debe ser ad-hoc ya que el mismo necesita tener varios grados de libertad y parametrizaciones específicas en lo que se refiere al estudio del comportamiento de dichos vehículos y al estudio de las magnitudes físicas de los experimentos que se abordan. Utilizar estos sistemas ya existentes implica una seria limitación en los estudios mencionados y un riesgo costoso para los experimentos pertenecientes a la carga útil. Se necesita un Sistema que provea datos sumamente confiables en cuanto a su integridad y calidad.

En los últimos años, se han realizado grandes esfuerzos en el desarrollo de métodos computacionales para resolver las ecuaciones que determinen la posición de un objeto en el espacio (Comas, Vescovo, & Legnani, Análisis del comportamiento caótico del flujo hipersónico en el vector GRADICOM II). Por ello, es sumamente necesario poseer un sistema de desarrollo abierto para poder realizar implementaciones de diversos algoritmos ad-hoc. Por lo tanto, se procederá al

diseño conceptual, construcción de un sistema de desarrollo y programación de nuevos algoritmos de adquisición de datos con el fin de obtener un sistema autónomo de adquisición de datos sincrónicos de alta velocidad controlada por FPGA, para implementarlo en la medición remota de magnitudes físicas y parámetros propios de un vehículo que permita un amplio rango de control de todas sus mediciones, variables y parámetros en estudio.

Se debe destacar que el Sistema completo será puesto a prueba primero con simulaciones computacionales y luego con pruebas y ensayos en el Laboratorio antes de utilizarlo en el campo también. Esto resulta ser una tarea compleja ya que existen diferentes niveles de problemas a resolver durante los procesos de diseño, sincronización, construcción, programación y ensamble del sistema global, ya que el mismo consta de varios subsistemas interconectados que deben funcionar en forma sincrónica, como una única unidad solidaria, y, además, algunos sub-conjuntos del sistema deberán operar de manera autónoma.

Es sumamente importante destacar que se han escrito tres publicaciones en revistas internacionales, una particularmente que se trata del tema de tesis en cuestión, y dos que se desprenden de temas relacionados con la aplicación de esta tesis. El primer artículo se denomina "Filtering and Acquisition of PCM Frames Using System Generator" y se publicó en la revista "International Journal of Electronics and Communication Engineering (IJECE)" con ISSN(P): 2278-9901; ISSN(E): 2278-991X en el Vol. 8, Issue 1, Dec - Jan 2019; 1-10 en la ciudad de Tamil Nadu, India.

International Journal of Electronics and Communication Engineering (IJECE)
ISSN(P): 2278-9901; ISSN(E): 2278-991X
Vol. 8, Issue 1, Dec - Jan 2019; 1-10

International Academy of Science, Engineering and Technology
Connecting Researchers; Nurturing Innovations

FILTERING AND ACQUISITION OF PCM FRAMES USING SYSTEM GENERATOR

Adrian Stacul & Edgardo Comas
Instituto De Investigaciones Cientificas Y Tecnicas Para La Defensa Buenos Aires, Argentina

ABSTRACT

The main purpose of this paper is the design, development, and implementation of a PCM bit-synchronizer based on a System Generator and Simulink model. The entire system will be applied to a ground station with an ad-hoc telemetric data acquisition system to be applied in unmanned aerial vehicles, atmospheric sounding rockets and nano-satellites monitoring. Based on this information, the ground station will be able to compute navigation parameters trajectories, velocities and attitudes. In particular, this PCM module was built to be used in atmospheric sounding vector evaluations by the Instituto de Investigaciones Científicas y Técnicas para la Defensa of Argentina.

KEYWORDS: *Bit-Synchronizer, FPGA, Ground Station, PCM Frame, System Generator*

A su vez, se han escrito dos artículos de proyectos relacionados a esta tesis y aplicación directa de la misma, los mismos ya fueron aceptados, pero aún no están publicados ya que el volumen de la revista no salió al día de la fecha.

El primero de estos de titula “A Hardware System with ARM-based data processing for Nano-Satellites”, fue aceptado en Enero del 2020 y se publicará en Julio del 2020 en la revista “International Journal of Reconfigurable and Embedded Systems (IJRES)” con ISSN 2089-4864 en la ciudad de Yogyakarta, Indonesia.

A Hardware System with ARM-based data processing for Nano-Satellites

Adrián Stacul *, Daniel Pastafiglia*, Ariel Dalmas Di Giovanni*, Martín Morales*, Sergio Saluzzi*, Gerardo García*, Agustín Gadea and Ramiro Puga****

* Digital Techniques Laboratory, Institute of Scientific and Technical Research for Defense (CITEDEF)

** Electronics department, National Technological University (UTN)

Article Info

Article history:

Received Jun 12th, 201x

Revised Aug 20th, 201x

Accepted Aug 26th, 201x

Keyword:

Processing Hardware

Cortex Processors

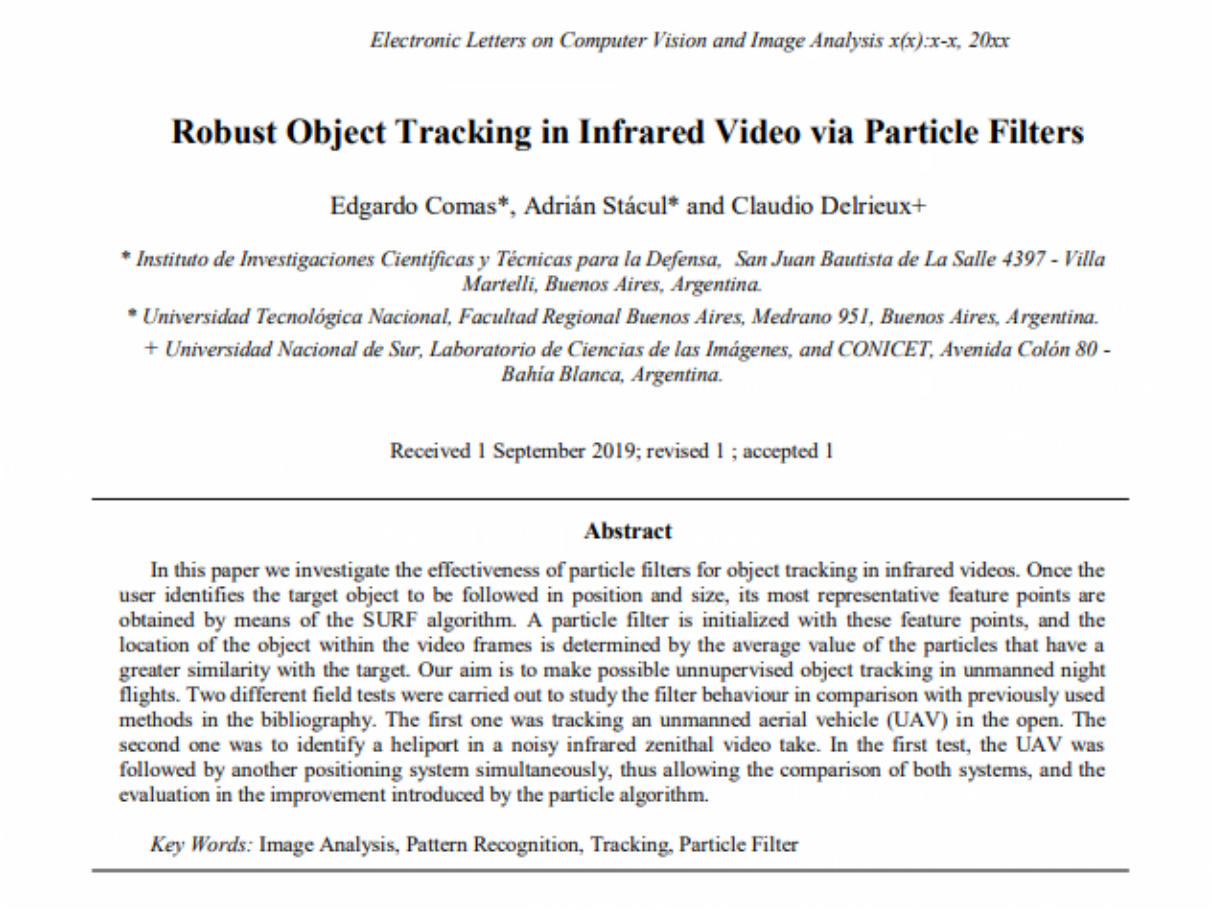
Nanosatellites

CubeSat

ABSTRACT (10 PT)

The Institute of Scientific and Technical Research for Defense in Argentina (Instituto de Investigaciones Científicas y Técnicas para la Defensa - CITEDEF) is developing a processing hardware module based on a ARM Cortex M4 processor from STMicroelectronics. The microcontroller (MCU) has the capacity to run at a maximum clock frequency of 180 MHz, integrates a Floating Point Unit (FPU). An 8MB SDRAM was included for dynamic data allocation. This hardware will host and process the algorithms to calculate and determine the nanosatellite's attitude. The module is intended to be Cubesat compatible, possess a flexible design, handles various inertial sensors and can manage backups on microSD memory cards with sizes up to 32GB.

El segundo artículo tiene como título “Robust Object Tracking in Infrared Video via Particle Filters” aceptado en la revista “Electronic Letters on Computer Vision and Image Analysis" (ELCVIA) con ISSN 1577-5097 en la ciudad de Barcelona, España.



Electronic Letters on Computer Vision and Image Analysis x(x):x-x, 20xx

Robust Object Tracking in Infrared Video via Particle Filters

Edgardo Comas*, Adrián Stácul* and Claudio Delrieux+

* *Instituto de Investigaciones Científicas y Técnicas para la Defensa, San Juan Bautista de La Salle 4397 - Villa Martelli, Buenos Aires, Argentina.*

* *Universidad Tecnológica Nacional, Facultad Regional Buenos Aires, Medrano 951, Buenos Aires, Argentina.*

+ *Universidad Nacional de Sur, Laboratorio de Ciencias de las Imágenes, and CONICET, Avenida Colón 80 - Bahía Blanca, Argentina.*

Received 1 September 2019; revised 1 ; accepted 1

Abstract

In this paper we investigate the effectiveness of particle filters for object tracking in infrared videos. Once the user identifies the target object to be followed in position and size, its most representative feature points are obtained by means of the SURF algorithm. A particle filter is initialized with these feature points, and the location of the object within the video frames is determined by the average value of the particles that have a greater similarity with the target. Our aim is to make possible unnupervised object tracking in unmanned night flights. Two different field tests were carried out to study the filter behaviour in comparison with previously used methods in the bibliography. The first one was tracking an unmanned aerial vehicle (UAV) in the open. The second one was to identify a heliport in a noisy infrared zenithal video take. In the first test, the UAV was followed by another positioning system simultaneously, thus allowing the comparison of both systems, and the evaluation in the improvement introduced by the particle algorithm.

Key Words: Image Analysis, Pattern Recognition, Tracking, Particle Filter

ESTADO DEL ARTE

En la actualidad, el procesamiento digital de señales se realiza por medio de dispositivos electrónicos como las FPGA, entre otros, que ya estas vienen implementadas en plaquetas o kits de desarrollo (Rojas, Franco M, & Pateti M, 2009). Un ejemplo es el caso específico de la familia “Spartan” de la firma Xilinx, estas FPGA son ampliamente utilizadas en aplicaciones de dispositivos como sistemas embebidos, control, sistemas de adquisición de datos y telecomunicaciones. Este tipo de plaquetas nos otorga versatilidad y escalabilidad al momento del diseño y desarrollo de la solución a implementar, y también nos otorga portabilidad y escalabilidad a futuro ya que no queda atado a un dispositivo puntual, sino que es transferible a otros nuevos dispositivos FPGA.

El FPGA es un dispositivo de fácil programación y mediante Lenguajes de Descripción de Hardware (HDL) y Co-Simulación utilización MATLAB. Su principal ventaja es que puede programarse múltiples veces bajando los costos cuando en el diseño se encuentran fallas y el sistema requiera ser actualizado y reprogramado.

Generalmente, en los sistemas diseñados para el procesamiento digital de señales se debe realizar un estudios minucioso y exhaustivo a priori para elegir un dispositivo que satisfaga las necesidades

del proyecto. Este puede tener características tales como: conversores, filtros digitales y sistemas de procesamiento integrados, para que en el tratamiento de la señal se pueda llegar a un trabajo menos extenso y más confiable debido a sus altas tasas de muestreo, velocidad de procesamiento, y grandes cantidades de bits a manejar que implican una menor pérdida de información y de calidad a comparación con sistemas microcontrolados antiguos.

Este proyecto no solamente pretende realizar un prototipo experimental de adquisición de datos, sino de dotar al Laboratorio de Técnicas Digitales de CITEDEF de una técnica nueva como es la de la implementación de código FPGA mediante la herramienta System Generator (Xilinx Inc., s.f.) para acelerar tareas de desarrollo. A su vez, este trabajo involucra la realización de pruebas de campo con vehículos no tripulados reales.

OBJETIVOS

El sistema de adquisición de datos sincrónicos de alta velocidad, que se propone desarrollar posee: algoritmos complejos que manejan el control del sistema, la adquisición de datos y la distribución de información desde y hacia las múltiples plataformas de operación. En la actualidad los sistemas comerciales equivalentes no tienen disponibles dichos algoritmos para ser modificados y/o reprogramados por el usuario, son considerados por los países desarrollados como unidades tecnológicas sensibles a la exportación, por lo tanto, son productos llave en mano. Los ajustes, modificaciones y/o adaptaciones solamente pueden ser realizados por el fabricante para lo cual hay que pagar costosas licencias que, por lo general, se renuevan anualmente.

Los sistemas comerciales equivalentes resultan ser muy costosos solo permiten que el usuario acceda a partes accesorias del software, pero no cumplen para nada con los requerimientos planteados en los objetivos. Esto genera un grave problema cuando se quiere implementar un sistema de adquisición y monitoreo de datos de alta velocidad, ya que el mismo necesita tener muchos grados de libertad en lo que se refiere al estudio del comportamiento de vehículos auto-dirigidos (como el UAV) y al estudio de las magnitudes físicas de los experimentos que se van a lleva abordo. Por ello, el objetivo primario de este trabajo de tesis representa al mismo tiempo un desafío tecnológico y un desarrollo de punta ya que el sistema a construir, deberá manejar palabras de datos entre 8 y 32 bits con velocidades de transferencia en el orden de los Mbits/s y que esta transferencia de información desde y hacia el vehículo autónomo, se realice en tiempo real, lo cual sería imposible implementar con un sistema comercial por los motivos ya descritas anteriormente. Para ello se va a trabajar con sistemas embebidos (en particular una FPGA de la empresa Xilinx).

Los objetivos específicos son:

- ✓ Desarrollar un método sistemático para recepción y sincronización de tramas telemétricas PCM de código NRZ-L sin pérdida de datos.
- ✓ Desarrollar un serializador de datos que permita la traducción de tramas PCM sincronizadas a una interface de datos USB serie mediante el estándar RS-232 y un puerto virtual.
- ✓ Determinar las métricas necesarias y ensayos específicos para validar la síntesis circuital, en una FPGA, obtenida a partir del método desarrollado.
- ✓ Implementar el desarrollo necesario en un kit de evaluación con una FPGA de bajo costo y contextualizarlo en un escenario real de trabajo para pruebas de campo.

Los alcances son:

- ✓ Consolidar la capacidad científica/técnica en materia de la sincronización a nivel de bit de tramas PCM mediante herramientas de diseño, simulación y ensayos con tecnología FPGA, a los efectos de poder determinar remotamente las magnitudes físicas de vectores y vehículos aéreos en general, específicamente trayectorias, velocidades y actitudes de los mismos, con base en la solución matemática de ecuaciones relacionadas, el procesamiento de señales de sensores específicos y la implantación de un puesto de monitoreo y supervisión desde tierra.

METODOLOGÍA

Con el fin de crear unas bases de conocimiento y avance en FPGA, se requiere implementar un sistema de adquisición (mencionado anteriormente) que procese las tramas telemétricas y cumpliendo las necesidades básicas que esto implica, como la calidad y la integridad de datos hacia el usuario. El presente proyecto se dividió en etapas que permitieran facilitar su ejecución, algunas de ellas son:

1. Investigación sobre la temática de telemetría, enfocándose a las estaciones de tierra y la adquisición de datos.
2. Investigación sobre el desarrollo de modelos en FPGA como nueva tecnología aplicada a la adquisición de señales de telemetría PCM.
3. Sistemas simuladores de datos PCM y procesamientos previos y conversiones asociadas.
4. Bloques de programación de FPGA incorporando la técnica de Matlab/Simulink.
5. Verificación y pruebas de campo.

CAPITULO 1 - SISTEMAS DE TELEMETRÍA

La telemetría es el proceso por el cual se miden ciertas características de un objeto (por ejemplo, la velocidad de un avión), y los resultados son transmitidos a una estación remota en tierra donde se visualizan, registran y analizan los mismos. Los medios de transmisión pueden ser alámbricos para entornos terrestres estáticos (e.g. plantas generadoras de energía), aire o el espacio para aplicaciones de vehículos aéreos o satélites.

Para poder entender el objeto de estudio de esta tesis se debe comprender, con anticipación, como opera un sistema de telemetría en su totalidad ya que el desarrollo planteado se lo someterá a pruebas de campo.

En la actualidad, en las aplicaciones de telemetría que soportan un gran número de mediciones, sería muy costoso e impracticable utilizar un canal de trasmisión diferente para cada una de las mediciones. El proceso de telemetría involucra agrupar las mediciones (como ser, presiones, velocidades y temperaturas) en un formato determinado para poder ser trasmitidas en una única trama o flujo de datos. Una vez recibido, la trama de datos se descompone recuperando cada una de las mediciones originales para su análisis posterior.

La telemetría permite estar en un lugar seguro (o conveniente) durante el seguimiento de lo que está ocurriendo en un lugar inseguro (o inconveniente). El desarrollo de vehículos aéreos, por ejemplo, es una de las principales aplicaciones de los sistemas de telemetría. Durante cierta prueba, una aeronave realiza una variedad de maniobras de vuelo. Los datos críticos de tales maniobras de vuelo se transmiten a una estación en tierra donde los ingenieros y técnicos de pruebas de vuelo visualizan los resultados en tiempo real y pueden ser analizados. El monitoreo en tiempo real permite tomar decisiones instantáneas sobre si se procede o se abandona la misión. Con el análisis en tiempo real, el ingeniero de pruebas de vuelo podrá solicitar que se repita una maniobra determinada, analizarla, aprobar los resultados o sustituirla con un plan de prueba alternativo. Los datos en tiempo real también se pueden capturar y guardar en medios de almacenamiento para un análisis posterior.

1.1. DESCRIPCIÓN GENERAL DE UN SISTEMA DE TELEMETRÍA

Hoy en día, los sistemas de telemetría son productos del tipo COTS (Commercial Off-The-Shelf), es decir, productos comerciales diseñados para ser implementados fácilmente en sistemas existentes sin la necesidad de un producto personalizado para tal fin (McKinney, 2001). Sin embargo, a pesar de que todos estos productos tengan muchos elementos en común, en el campo

científico se requiere de un desarrollo específico de telemetría únicamente para cumplir los requisitos de la aplicación a la cual se dará uso.

Un sistema de telemetría muchas veces es vista como dos grandes componentes, el sistema de a bordo y la estación o puesto de tierra (Figura 1). En la actualidad, cualquiera o ambas pueden estar en el aire o en el terreno (Comas, Pastafiglia, Bruña, Dalmas Di Giovanni, & Stacul, 2015).

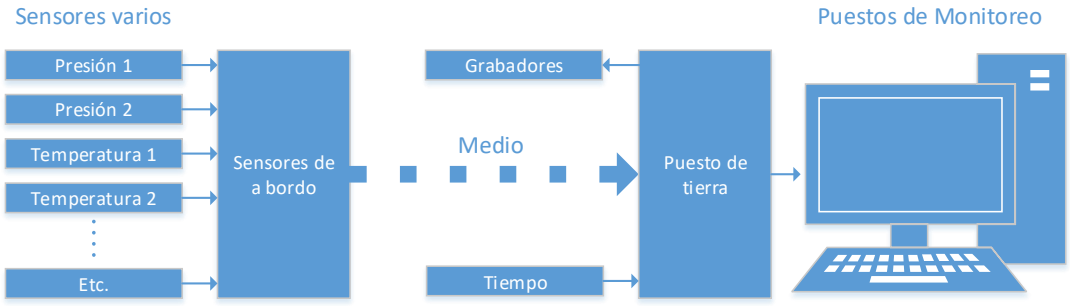


Figura 1. Sistema de telemetría

El sistema de adquisición de a bordo (Figura 2) comienza cuando los sensores (transductores) miden la magnitud o atributo físico y lo transforman en un valor eléctrico. Algunos sensores producen una tensión eléctrica directamente (sensores de termocuplas para medir temperatura o medidores de deformación piezoeléctrica para medir aceleraciones), mientras que otros necesitan de alguna excitación (medidores de tensión resistivos, potenciómetros de rotación, etc.). Los sensores generalmente están conectados a acondicionadores de señales, los cuales proveen energía a estos para operar y ser compatibles con la próxima etapa de adquisición.

El multiplexor toma cada una de las mediciones analógicas y produce un único flujo de pulsos con una tensión eléctrica relativa a la medición del canal. Esta rigurosa fusión de datos en una única serie de pulsos se denomina “Multiplexación por División de Tiempo” ó TDM (Time Division Multiplexing).

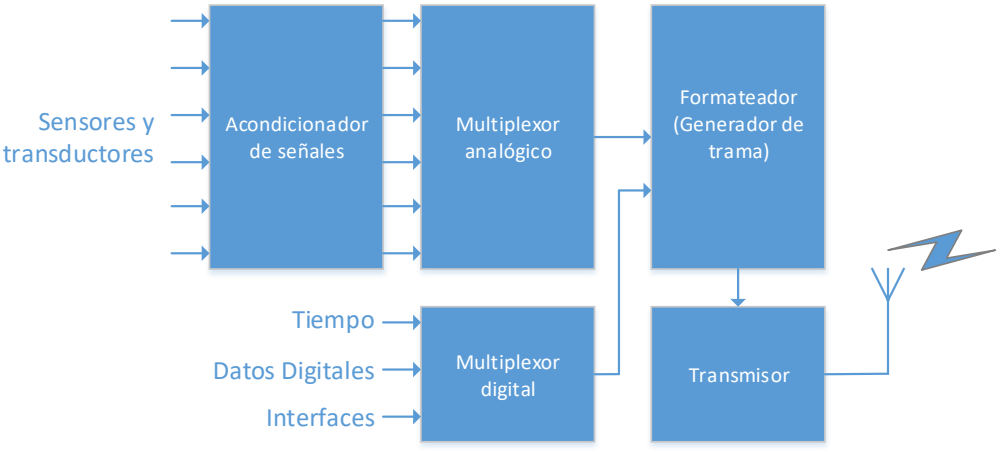


Figura 2. Sistema de adquisición de a bordo

La configuración en la que la altura del pulso del flujo TDM es proporcional al valor de la medición se denomina “Modulación por Amplitud del Pulso” (PAM – Pulse Amplitude Modulation). A dicha señal se concede un único set de pulsos con el fin de identificar la fuente de medición respecto del valor de dicha medición (Schwartz). La modulación PAM posee muchas limitaciones, la cual incluye la precisión, el número de mediciones que soporta y una pobre capacidad de integrarse a señales digitales. Hoy en día se utiliza la “Modulación por Pulsos Codificados” (PCM - Pulse Code Modulation) para sistemas de telemetría (Consultative Committee for Space Data Systems, 2000), donde se tiene una alta precisión, donde existe una resolución limitada por el conversor analógico-digital ADC (Analog to Digital Converter), y cientos de mediciones pueden ser adquiridas a lo largo de la trama digital desde múltiples fuentes, incluyendo datos y memoria de la computadora de a bordo (Figura 3).

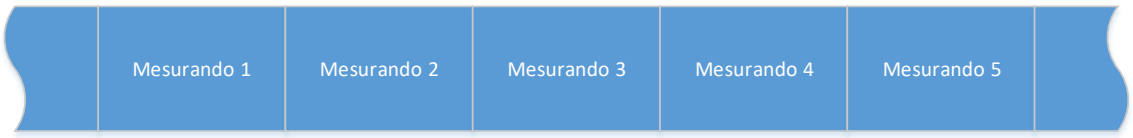


Figura 3. Flujo de mediciones en formato TDM

En un sistema basado en PCM, la salida PAM original del multiplexor es digitalizada a formato paralelo. Esto, junto a otras fuentes de datos digitales, se fusionan y formatean con algún dato de sincronismo para la identificación del mesurando. La salida del formateador serializa el flujo de datos paralelos en una cadena de datos binarios para su transmisión en el medio elegido. Todos los componentes a partir de los sensores hasta el formateador comprenden el “Encoder” (Codificador). Existen otros “Encoders” remotos que a menudo se utilizan para multiplexar datos de sensores adicionales en la salida del codificador principal, esto no solamente se utiliza para ampliar el número de mediciones, sino que también para eliminar el recorrido de cables requeridos para cada sensor.

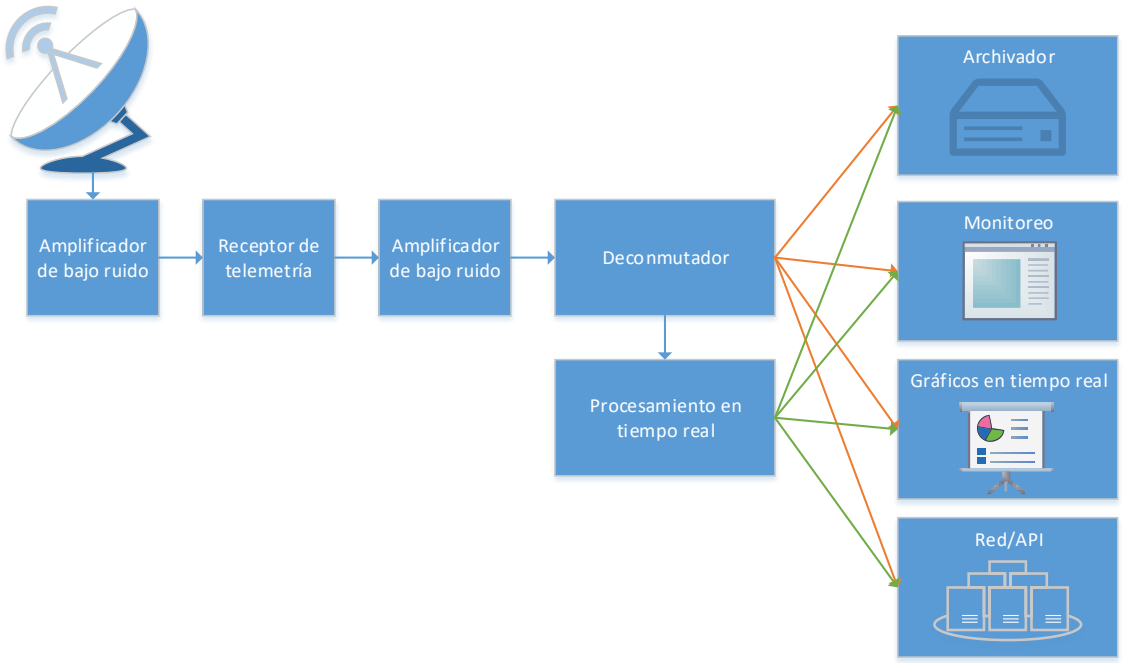


Figura 4. Diagrama en bloques de un Puesto de Tierra

La salida del "Encoder" principal es filtrada y enviada para ser transmitida vía un transmisor de radiofrecuencia, cable coaxil, línea telefónica, etc. El filtrado suaviza la señal digital, es decir, elimina los abruptos cambios entre los estados, y se utiliza para reducir el contenido de frecuencia y por lo tanto el ancho de banda del transmisor necesario. En la estación terrena, Figura 4, el dato recibido es amplificado, y ya que el medio de trasmisión distorsiona la señal filtrada, un sincronizador a nivel de bit (Bit Synchronizer) reconstruye la señal a la serie de pulsos originales. Luego se realiza una de-conmutación (Decom) que reconoce el patrón de sincronismo y paraleliza la señal serie, y además separa la trama PCM en sus valores originales de mediciones (también conocido como parámetros principales) y de datos. El sistema a desarrollar en esta tesis integra el Bit Synchronizer y el Decom en una única unidad de adquisición de bajo costo para luego transferir los datos a la o las computadoras que componen el o los clientes de monitoreo.

1.2. HARDWARE DE A BORDO

1.2.1. SENSORES

Una amplia variedad de sensores (también conocidos como transductores) se utilizan para medir y adquirir un valor de cierta propiedad física (ver Figura 5). Generalmente el ingeniero de instrumentación selecciona el dispositivo adecuado para satisfacer los requerimientos adecuados para la aplicación de: respuesta, exactitud, tamaño, costo, especificaciones ambientales, etc. Los acondicionadores de señal sirven de interfaz entre los sensores y el sistema de adquisición de datos de a bordo, se acoplan a los sensores analógicos que requieran algún tratamiento de señal como

ser: energía, calibración, amplificación de señal, filtrado, etc. y darle compatibilidad a la próxima etapa del sistema.

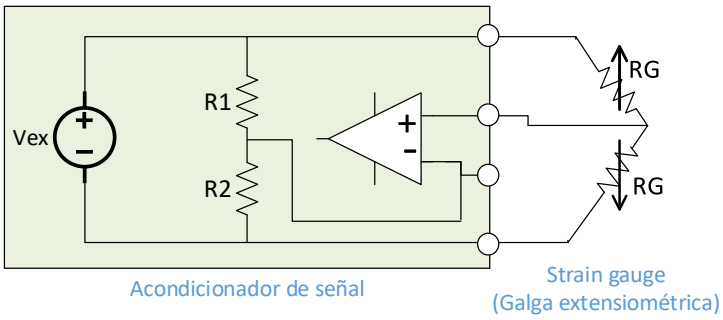


Figura 5. Ejemplo de un sensor: Circuito puente básico de galgas extensiométricas

La relación absoluta entre la salida de tensión eléctrica del acondicionador y el valor físico real de la magnitud a medir puede variar con el tiempo, la altitud, presión, temperatura, etc. Por lo tanto, los acondicionadores de señal también incorporan características de calibración para corregir las relaciones mencionadas. Un sistema bajo prueba (DUT – Device Under Test) puede ser sometido a características físicas conocidas y su salida es medida y comparada para determinar y verificar la relación entre el sensor y su salida.

Por ejemplo, estando inmóvil, los flaps de un avión pueden moverse en ángulos conocidos, y se toman mediciones sobre el sensor mismo o en la salida de la electrónica de a bordo. Un gráfico de ángulo vs. salida podrá ser utilizado por el puesto de tierra para el monitoreo de datos en tiempo real, y así evaluar el comportamiento del sistema de adquisición de a bordo.

1.2.2. MODULACIÓN

La modulación es la técnica en la que el valor de cada muestra (es decir, la señal modulante) cambia sistemáticamente ciertas características de una señal portadora (por ejemplo, la amplitud (altura) o la frecuencia (tiempo)) (Bryant, 2013). La onda modulada resultante transporta los datos. Inversamente, la eliminación de la señal portadora da como resultado la medición original.

El flujo de datos TDM generado en el esquema básico del multiplexor se lleva a cabo por la señal PCM. También se utilizan otras tres formas de modulación: PDM (modulación de ancho de pulso), PPM (modulación por posición de pulso) y PAM (pulso modulación de amplitud). Las formas de onda resultantes de estas técnicas de modulación de una simple señal de datos analógicos se muestran en la Figura 6.

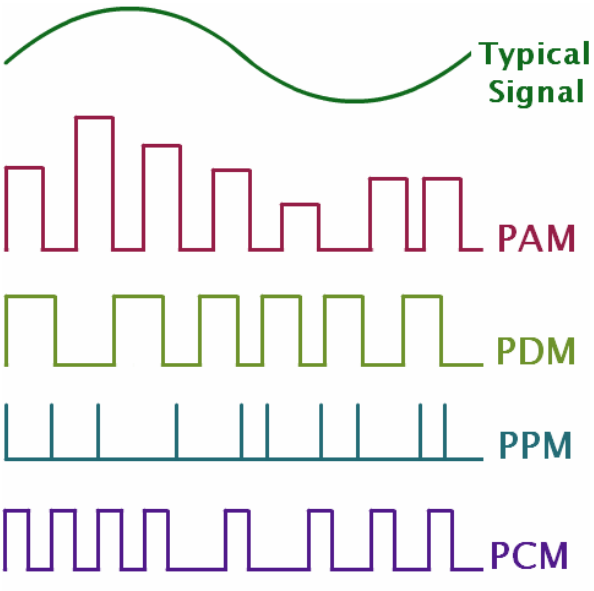


Figura 6. Diferentes tipos de modulación

Generalmente los valores de las amplitudes son degradados a causa del ruido, por ello se utilizan flujos de datos multiplexados con un tipo de modulación de pulsos de amplitud constantes. La modulación tipo PDM (Modulación de señal por Densidad de Impulsos) contiene la información en el ancho de pulso, que varía directamente con la amplitud de la señal (Stauth & Sanders, 2008). La Modulación por Posición de Pulso, PPM (Pulse Position Modulation), es el resultado de la diferenciación de la señal PDM y luego rectificada, donde la distancia entre los dos pulsos representa la amplitud muestreada de una onda sinusoidal, con el primer impulso se marca una referencia de tiempo cero. La potencia media de un sistema con PPM es mucho menor que la requerida para una PDM, pero a costa de un mayor ancho de banda.

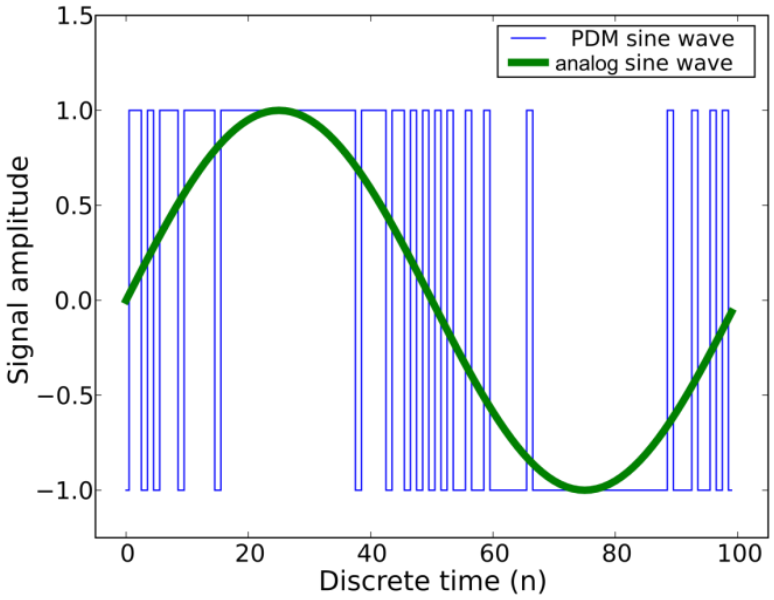


Figura 7. Ejemplo de señal PDM. Extraído de https://en.wikipedia.org/wiki/Pulse-density_modulation

Tanto la PDM como la PPM utilizan pulsos de amplitud constante, pero siguen siendo representaciones analógicas de una señal analógica. En un sistema con PCM, cada pulso es codificado en una representación binaria equivalente antes de la transmisión, esto se denomina "código de línea" [1]. En muchos casos, el dato PCM, no solamente es trasmitido, sino almacenado a bordo también.

En ambos casos, tanto en la grabación como en la transmisión de los datos, se deben establecer que patrones se utilizan para representar una lógica determinada. A través de los años, se han diseñado códigos PCM para representar niveles lógicos, con el fin de lograr el mayor rendimiento para una cierta aplicación (ver detalle en la Figura 8).

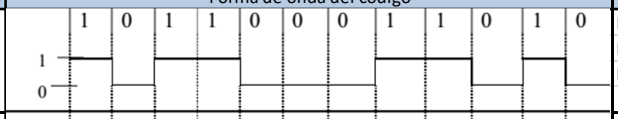
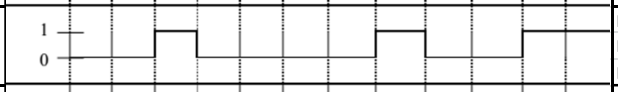
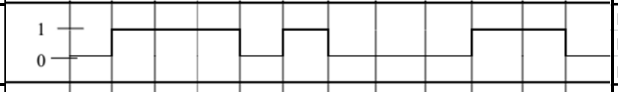
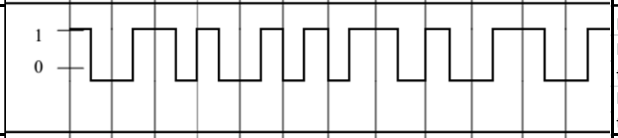
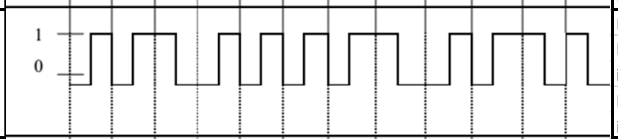
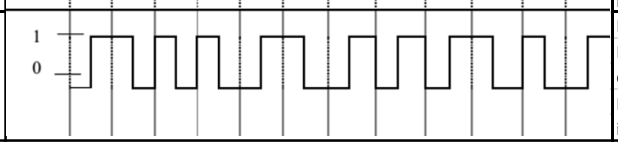
Nombre del código	Forma de onda del código	Definiciones
NRZ-L		Non-Return-to-Zero - Level El 1 "uno" está representado por un nivel El 0 "cero" está representado por el otro nivel
NRZ-M		Non-Return-to-Zero - Mark El 1 "uno" está representado por un cambio en el nivel El 0 "cero" está representado por un NO cambio en el nivel
NRZ-S		Non-Return-to-Zero - Space El 1 "uno" está representado por un NO cambio en el nivel El 0 "cero" está representado por un cambio en el nivel
BiØ-L		Bi-Phase-Level (*) El 1 "uno" está representado por un nivel de "uno" con una transición a un nivel de "cero" El 0 "cero" está representado por un nivel de "cero" con una transición a un nivel de "uno"
BiØ-M		Bi-Phase-Mark (*) El 1 "uno" está representado por un NO cambio de nivel al inicio del período de bit El 0 "cero" está representado por un cambio de nivel al inicio del período de bit
BiØ-S		Bi-Phase-Space (*) El 1 "uno" está representado por un cambio de nivel al inicio del período de bit El 0 "cero" está representado por un NO cambio de nivel al inicio del período de bit
(*) Los códigos BiØ pueden obtenerse de los correspondientes códigos NRZ invirtiendo el nivel de la segunda mitad de cada intervalo de bit		

Figura 8. Códigos de datos PCM

[1] En telecomunicaciones, un código en línea o código de línea (modulación en banda base) es un código utilizado en un sistema de comunicación para propósitos de transmisión. Los códigos de línea son frecuentemente utilizados para el transporte digital de datos. Estos códigos consisten en representar la señal digital transportada respecto a su amplitud respecto al tiempo. La representación de la onda se suele realizar mediante un número determinado de pulsos. Estos pulsos representan los "1" (unos) y los "0" (ceros) lógicos. Los tipos más comunes de codificación en línea son el unipolar, polar, bipolar y Manchester.

1.2.3. CONMUTACIÓN

Un barrido completo por el multiplexor produce un flujo de datos con palabras que contienen el valor de cada medición, y cada escaneo genera la misma secuencia de palabras, solamente el valor de cada mesurando es capturado, no el nombre de la medición. Si cada dato de la medición es adquirido, no existe manera de distinguirlas entre sí. Por lo tanto, se adiciona una palabra única llamada sincronismo de trama, al principio o al final de cada trama para ser utilizada como referencia en el proceso de de-conmutación de datos donde se extrae cada valor individual (ver Figura 9).

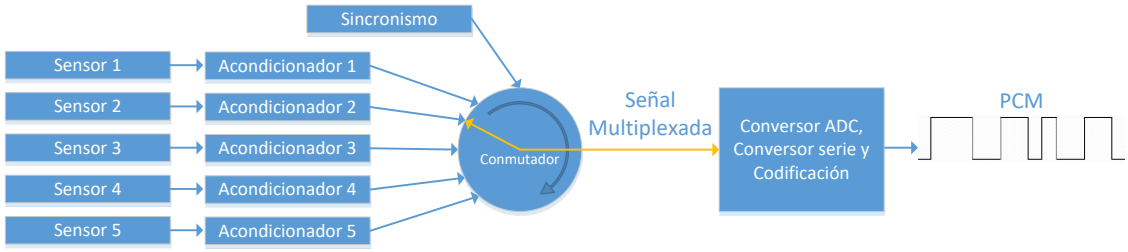


Figura 9. Conmutación básica con sincronismo de trama

Como ejemplo podríamos decir que para caracterizar una vibración determinada en una aeronave requerimos varias muestras por segundo (cientos) a comparación de querer adquirir una temperatura determinada. Teniendo en cuenta el teorema de Nyquist, uno debe muestrear el dato como mínimo el doble de la máxima componente de frecuencia que quiere ser capturada, típicamente se utiliza un muestreo de 5 veces por encima de la máxima frecuencia para estos casos, y se le adiciona un filtro pasabajo para eliminar frecuencias que uno no puede digitalizar precisamente y prevenir el efecto conocido como “aliasing” (ScienceDirect - ELSEVIER, s.f.).

Si tomamos la peor condición de muestrear todo a la máxima tasa, es de esperar que exista una cantidad de residuos apreciables en el espectro de frecuencia de la portadora y la potencia. Por lo tanto, las tasas de muestreo deben variar con respecto al contenido frecuencial y ser un tanto independiente uno de otro, los valores de mediciones que posean diferentes velocidades de adquisición. Esto se puede solucionar con la técnica de super-conmutación (Figura 10) donde el mesurando aparece varias veces en cada trama.

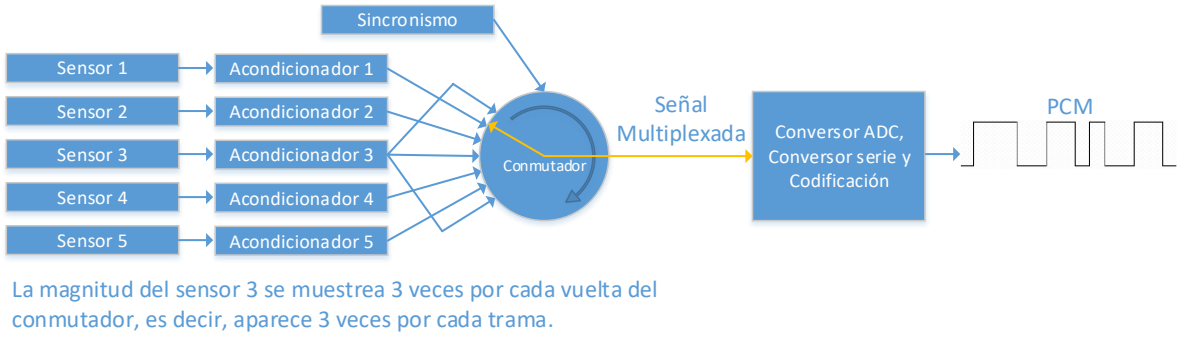


Figura 10. Esquema general de una super-conmutación

El esquema contrario ocurre en la sub-conmutación y datos asíncronos embebidos, donde una posición en el tiempo tiene varios significados.

1.2.4. PATRONES DE SINCRONIZACIÓN DE DATOS

La identificación de cada trama se logra a través de la palabra de sincronización, la cual es una secuencia única de unos y ceros. Generalmente, el patrón de sincronismo es una secuencia pseudo-aleatoria que es poco probable que se produzca al azar en los datos adquiridos y típicamente ocupa dos palabras (o más) en el principio de una trama determinada.

El estándar IRIG106 recomienda una lista de patrones de sincronismo para longitudes de 16 hasta 33 bits, ver Tabla 1, donde los primeros tres bits a transmitir deben ser siempre un "1", independientemente si se utiliza una alineación de MSB (Bit Más Significativo - Most Significant Bits) o LSB (Bit Menos Significativo - Least Significant Bit).

Tabla 1. Patrones de sincronismo recomendados por el estándar IRIG106

Longitud	Patrones
16	111 010 111 001 000 0
17	111 100 110 101 000 00
18	111 100 110 101 000 000
19	111 110 011 001 010 000 0
20	111 011 011 110 001 000 00
21	111 011 101 001 011 000 000
22	111 100 110 110 101 000 000 0
23	111 101 011 100 110 100 000 00
24	111 110 101 111 001 100 100 000
25	111 110 010 110 111 000 100 000 0
26	111 110 100 110 101 100 110 000 00
27	111 110 101 101 001 100 110 000 000
28	111 101 011 110 010 110 011 000 000 0
29	111 101 011 110 011 001 101 000 000 00
30	111 110 101 111 001 100 110 100 000 000
31	111 111 100 110 111 110 101 000 010 000 0

32	111	111	100	110	101	100	101	000	010	000	00
33	111	110	111	010	011	101	001	010	010	011	000

La longitud de la sincronización de trama generalmente es más larga que las palabras de datos para reducir la probabilidad de que los datos reales se igualen a ella. El sincronismo de trama también debe estar en proporción con el número de palabras de la trama (por lo general, ocupa de 1% a 5% de la trama). El de-commutator es programado para engancharse a este patrón para comenzar la regeneración de los valores de las mediciones originales conmutadas.

1.3. PUESTO DE TIERRA

La creación de una estación terrestre de telemetría incluye:

- ✓ Definir para el sistema de adquisición de datos, las características de los sensores y acondicionadores de señal.
- ✓ Verificar que la estructura de telemetría se adapte a los requisitos de frecuencia de muestreo, así como las limitaciones del hardware de adquisición.
- ✓ La trama de datos se deberá definir hasta el nivel de palabras y bits para poder mostrar los resultados o los datos analizados en tiempo real.
- ✓ Definición de los datos para las palabras adecuadas para gestionar un simulador PCM para la evaluación del sistema.
- ✓ Establecer los datos de calibración para cada sensor (por ejemplo, utilizando base de datos) y de ser necesario crear algoritmos de procesamiento y coeficientes de escalado para convertir parámetros a unidades físicas de ingeniería.
- ✓ Creación de visualizadores para cada tipo de medición, con sus atributos necesarios y unidades físicas.
- ✓ Definir qué datos se almacenan en disco.

El tiempo necesario para configurar y verificar los sistemas de telemetría es significativo dado que los archivos de instalación para el sistema aéreo y terrestre contienen un gran número de datos. Para ello pueden ser útiles el uso de herramientas de auto-configuración o sistema de base de datos.

1.3.1. RECONSTRUCCIÓN DE LA TRAMA PCM

En la estación de tierra, la trama PCM, ya sea que ingrese directamente por líneas de cobre o fibra óptica, o que se reciba por una antena y un receptor de telemetría, es reconstruida en sus datos crudos originales (Figura 1).

Debido a que el sistema de trasmisión distorsiona la señal (Figura 11), la señal PCM recibida debe ser reconstruida. Antes de ser transmitida, la señal de datos PCM es filtrada para reducir el ancho de banda requerido y asegurando que la potencia se concentre en el espectro que lleva el dato (Dostis).

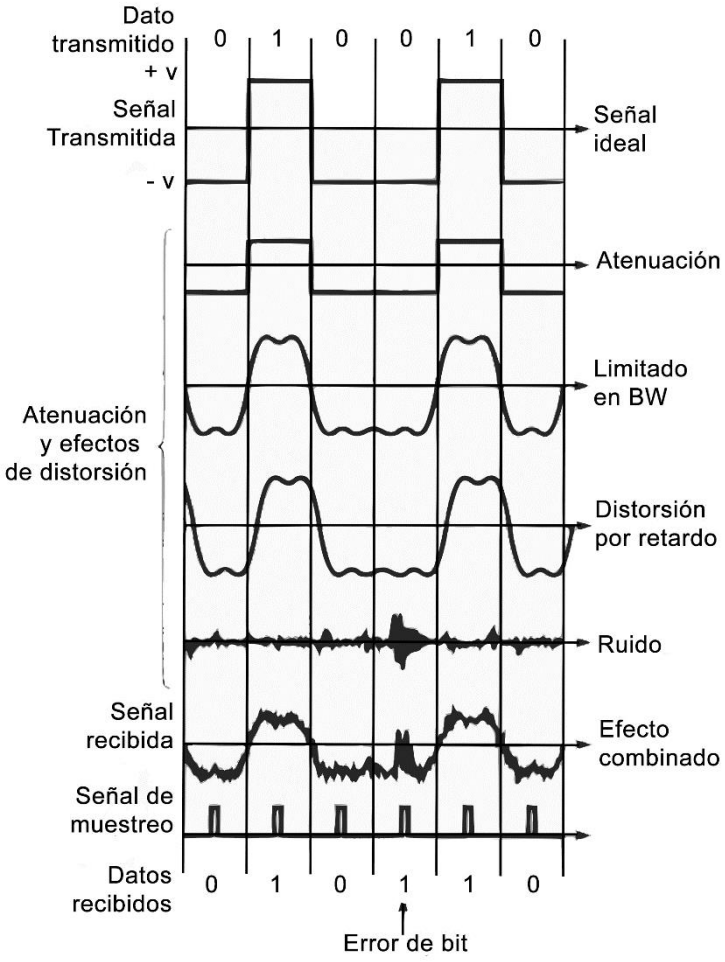


Figura 11. Efectos en una señal PCM. Extraído de (Halsall, 1995) Figura 2.6, Pág. 32

La primera función en el procesamiento de la señal de la adquisición es reconstruir el flujo de datos con un número mínimo de errores en símbolos (en este caso, cada símbolo está representado por un byte). A continuación, a la información reconstruida se le genera una temporización sincrónica, es decir, se reconstruye el "clock" original de la señal PCM. Esta función fundamental, de procesamiento de señal, se denomina sincronización de bits. Un sincronizador de bits (Bit Synchronizer o Bit-Sync) es un dispositivo que establece una serie de pulsos de "clock" que son sincrónicos en una señal entrante ("enganchados" con el patrón de sincronismo).

En este desarrollo, la señal que ingresa al Bit-Sync se amplifica e ingresa a un oscilador interno que se engancha con los datos. Cada bit que ingresa al Bit-Sync se reconstruye con un algoritmo

específico (en este trabajo de tesis se explica el método en el Capitulo 3), determina si el nivel de la señal se corresponde con un "0" lógico o un "1" lógico, y su rendimiento (cuan eficaz es la interpretación de bits del Bit-Sync) se obtiene de una curva teórica de "Relación señal ruido vs. Tasa de error de bit", ver en la Figura 12.

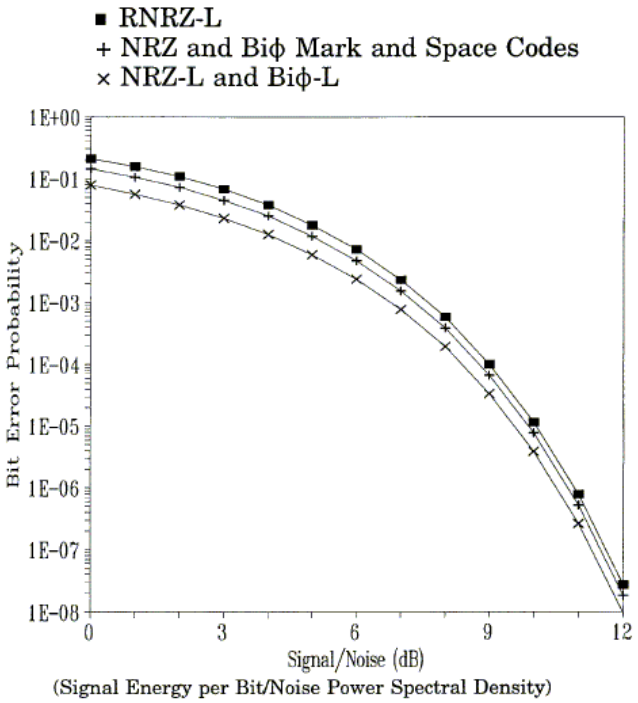


Figura 12. Rendimiento teórico de probabilidad de error de bits de para diversas técnicas en PCM (suponiendo una perfecta sincronización de bits). Extraído de (Telemetry Standars, 2015).

1.3.2. SINCRONIZACIÓN DE TRAMAS

Las tramas PCM no solamente deberían ser recibidas en forma contigua, completas y sin errores, además deben ser aisladas del sincronismo, lo cual es una terea es ardua y compleja de por sí debido a la presencia de:

- ✓ Errores de bits: Aparece un "1" donde debería haber un "0", o viceversa.
- ✓ Desplazamientos: Perdidas de un bit o varios bits contiguos, bits no detectados.
- ✓ Secuencias aleatorias de datos: Ruido inherente al sistema.

El operador del puesto de tierra podría elegir el número de tramas válidas antes de aceptar los datos. Esto se podría realizar mediante alguna especificación en la detección del sincronismo en la trama. En este trabajo se utiliza solamente una trama, ya que el sincronismo es chequeado constantemente en la recepción. En algunas aplicaciones específicas, generalmente científicas, esto es una tarea crítica y se debe contemplar no recibir tramas erróneas, y si fuera así, corregirla para obtener todas las tramas válidas. Para esto, se puede considerar cuatro modos operacionales, ver Figura 13.

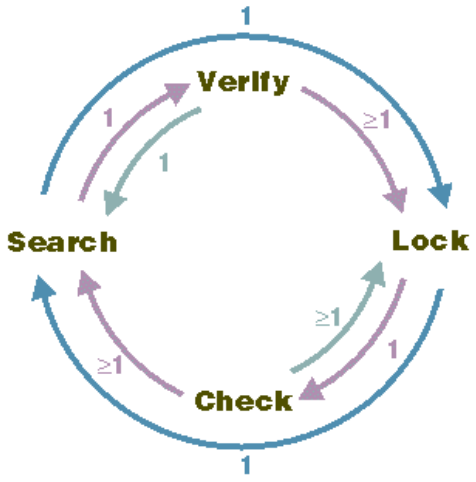


Figura 13. Estados en la sincronización de tramas (los números en violeta indican la cantidad de tramas válidas que deben estar presentes para poder pasar al próximo estado, los celestes indican números de tramas invalidas)

- Search (Búsqueda): El sincronizador rastrea algún posible patrón de sincronismo.
- Verify (Verificación): Una vez que se detecta un tentativo patrón, se fija una ventana en un tiempo fijo y se predice la re-ocurrencia del patrón de sincronismo. Si el patrón vuelve a aparecer en la ventana preestablecida, para un número de dado de tramas determinado, el sincronizado avanza al estado "Lock".
- Lock (Enganche): Una vez que la sincronización se establece, podrán ser identificados los diferentes canales que contienen los valores de las mediciones.
- Check (Chequeo): Si luego de un enganche, no se encuentra un patrón de sincronismo, que queda en este estado.

Las condiciones para pasar de un estado a otro se definen en:

- "Search" to "Lock" (de búsqueda a enganche): Debe ser detectado un número consecutivo de patrones válidos de sincronismo en el flujo de datos antes de avanzar en el Deconmutador. Cuando se detecta el primer patrón válido de sincronismo de trama, se cambia desde el estado "Search" al modo "Verify", y se mantiene en dicho estado hasta que se detecta el segundo patrón válido de sincronismo. Si un tercer patrón de sincronismo es válido, se avanza desde "Verify" a "Lock", si esto no sucede se regresa al estado de "Search". Un emparejamiento del 100% al patrón programado puede no suceder siempre, es por dicho motivo, que el Deconmutador puede tener opciones de configuración para permitirse pasar al próximo estado.
- "Lock" to "Search" (de enganche a búsqueda): Debe ser detectado un número consecutivo de patrones inválidos de sincronismo en el flujo de datos antes de avanzar en el Deconmutador al estado "Search". Cuando se detecta el primer patrón inválido de

sincronismo de trama, se avanza desde el modo “Lock” al modo “Check”, y queda en el modo “Check” mencionado cuando se detecta el segundo patrón invalido de sincronismo. Si un tercer patrón de sincronismo es inválido, se avanza a “Search”, sino, regresa al estado “Lock”.

Otras características del Bit-Sync son:

- Sync Pattern Bit Errors (errores de bit en el patrón de sincronismo): Se calcula el número de bits correctos para determinar si un patrón de sincronismo de trama es válido o no. Por ejemplo, si el patrón de sincronismo posee una longitud de 32 bits, y el “Sync Pattern Bit Errors” se configura en 4, se buscará que al menos 28 bits estén correctos dentro de dicho patrón.
- Bit Aperture (apertura de bit): Se puede habilitar o deshabilitar el desplazamiento de bits (bit slip) en el patrón de sincronismo de trama. Por ejemplo, si dicha constante se configura en “1”, se permite que el patrón de la trama de sincronismo llegue un bit “antes” o 1 bit “después” y sigue siendo válido.

1.3.3. DE-CONMUTACIÓN

Después de la sincronización de trama, los canales individuales que contienen los valores de las mediciones son identificados de acuerdo a la posición en dicha trama. Existen diferentes arquitecturas de hardware/software que contienen las definiciones y relaciones de los datos. Por ejemplo, se puede anexar una etiqueta única a cada mensurando de la trama cruda o palabra de datos, y que puede denominarse arquitectura de flujo de datos, y esta etiqueta mantiene el valor de la medición para luego ser convertido en unidades físicas o ser procesado. Otra arquitectura común es reacomodar los datos en un nuevo formato o estructura que sea más apropiada para el tratamiento de señales.

Las aplicaciones como el monitoreo de cohetes sonda o ensayos en vehículos aéreos requieren una visualización de múltiples parámetros en simultáneo tanto remotos como locales. Por lo tanto, se pueden unificar todos los datos en una única gran estructura y así poder contemplar todos los valores de mediciones posibles en la visualización.

Para poder obtener una rápida respuesta, el deconmutador puede contener definiciones de tipos de datos ya integradas al hardware. El estándar IRIG-106 soporta 16 formatos. En este caso se requieren algunos pocos formatos comunes para aplicaciones de lanzamiento de cohetes sonda y vuelos de aeronaves, pero las velocidades de muestreo son relativamente altas, por este motivo se emplea un conjunto de hardware dedicado y software particular. En otras aplicaciones, por ejemplo, en el monitoreo de satélites, si bien se utilizan ciento de tipos de formatos de datos, las

tasas de muestreo son relativamente bajas, y mayormente se utiliza software para el procesamiento de datos.

1.3.4. SIMULACIÓN Y CODIFICACIÓN

Es altamente deseable contar con un simulador de datos PCM para producir datos PCM que permita verificar y ensayar el sistema en general. Existen varios tipos de simuladores con deferentes características y su salida PCM es compatible con el estándar IRIG-106. Los mesurandos contenidos en la trama pueden ser estáticos o variables conocidos para poder cotejar los resultados y evaluar el comportamiento del sistema adquisidor cuando los sistemas sean críticos o de fines científicos.

1.3.5. PROCESAMIENTO EN TIEMPO REAL

Es una característica donde el operador del sistema de tierra puede contemplar un procesamiento en los datos del deconmutador, por ejemplo, una manipulación a nivel de bits, la cual se debe realizar en tiempo real y evitar latencias en el sistema.

1.3.6. GRAFICADORES DE TIEMPO REAL

La visualización en un puesto de tierra es diferente para cada tipo de dato. Por ejemplo, se puede utilizar un tipo de monitoreo de valores para datos alfanuméricos o graficadores en tiempo real para datos procesados, o escenarios en tres dimensiones para evaluaciones de trayectorias.

La utilización de redes e intranet amplían el dominio del monitoreo más allá de la consola del operador, de modo que cada usuario visualiza los datos de interés de manera precisa e independiente de los demás.

1.3.7. GESTIÓN DE ARCHIVOS

El almacenamiento de los datos telemétricos capturados se utiliza para post-procesamiento de la información y para evaluar la optimización de algoritmos u otras cuestiones de la aplicación. Típicamente se almacena el dato crudo, el cual es el más cercano a la señal recibida, también se almacena el dato de-conmutado y su estructura cruda. Estos datos son almacenados en discos rígidos o memorias de estado sólido. Se puede tener la opción para almacenar toda la información PCM o solamente la de interés, aunque en este trabajo se almacenarán ambas.

1.3.8. DISTRIBUCIÓN DE DATOS

Las arquitecturas basadas en redes de computadoras tipo LAN (Local Area Network – Red de área local) permiten la distribución de los datos en tiempo real hacia estaciones de trabajo

(computadoras), donde se visualiza, analiza y almacena el flujo de información de manera individual.

Los grandes sistemas de puerto de tierra a menudo responden a características particulares (de hardware y de software) para poder distribuir datos en tiempo real. Los clientes o nodos conectados a una red deben estar diseñados específicamente para no solapar los datos cuando el sistema es diseñado para altas tasas de muestreo. Para evitar este problema, en ciertos casos, se recurre a técnicas de compresión de datos, el cual reduce la cantidad de información notablemente en la red, aunque en aplicaciones científicas esto podría implicar tener un error y demoras considerables en el tratamiento de la información. En este trabajo, dada las características y requerimientos proyectuales, fue sumamente necesario diseñar el sistema entero sin pérdida de datos y sin compresión.

1.4. FIN DEL CAPÍTULO

A partir de esta información teórica, se pasará a estudiar la tecnología FPGA para luego implementar el adquisidor. Se pretende que el adquisidor forme parte de un puerto de tierra completo aplicando lo mencionado en este capítulo, tanto en hardware como en software.

CAPÍTULO 2 - TECNOLOGÍA FPGA

En este capítulo se introduce a los dispositivos lógicos programables y su evolución hasta los FPGA (siglas en inglés, *Field Programmable Gate Arrays*) que es nuestro caso de estudio para el diseño e implementación, analizando su arquitectura, tecnología, sus formas de programación[2] e implementación y las diferentes áreas en donde se puede aplicar esta tecnología.

Las FPGA son dispositivos semiconductores que contienen bloques interconectados en forma matricial programables denominados CLBs (Bloques Lógicos Programables - Configurable Logic Blocks). Estas FPGAs pueden ser programadas y re-programadas para cumplir con una aplicación deseada o requerimientos específicos, esta característica denota la diferencia con las ASICs (Circuitos Integrados para Aplicaciones Específicas), las cuales son fabricadas concretamente para un único diseño o única tarea. Si bien existen FPGA con memoria OTP (programables una única vez), las dominantes son las basadas en memoria del tipo SRAM donde pueden ser re-programadas a medida que evoluciona el diseño.

Una particularidad de las FPGAs es que son dispositivos multinivel programables de propósito general ya que integran una gran cantidad de dispositivos lógicos programables en un solo chip. El tamaño y velocidad de los FPGA es comparable a los ASICs, pero los FPGA son más flexibles, su ciclo de diseño es más corto y los recursos necesarios generan menores gastos. La adopción de chips FPGA en la industria ha sido impulsada por el hecho de que los FPGAs combinan lo mejor de los ASICs y de los sistemas basados en procesadores.

Debido a su naturaleza programable, las FPGAs son un complemento ideal para muchos mercados diferentes. Una firma líder de estos dispositivos es Xilinx (Xilinx, 2020), la cual proporciona soluciones completas que consisten en dispositivos FPGA, software avanzado y núcleos integradores de procesamiento (en inglés, IP cores), listas para su uso en aplicaciones (Xilinx Inc., s.f.), tales como:

- ✓ Defensa y entorno Aeroespacial: Las FPGAs tolerantes a la radiación en conjunto con diseño de propiedad intelectual para el procesamiento de imágenes, la generación de formas de onda, y la reconfiguración parcial de sistemas de radiocomunicaciones del tipo SDR (del inglés Software Defined Radio),
- ✓ Prototipos de ASICs: EL desarrollo de prototipos de ASICs con FPGAs permite el modelado rápido y preciso y la verificación de software embebido para SoC (sistemas en chip – system on chip).

[2] Anteriormente, no se solía utilizar el término programación (software) sino síntesis, ya que las FPGA no contienen software dentro, sino se modeliza un hardware dedicado. Ambos términos son aceptados hoy en día.

- ✓ Sistemas de Audio: Las FPGAs y sus plataformas de diseño habilitan altos grados de flexibilidad en el desarrollo, un rápido time-to-market (tiempo de puesta del producto al mercado) y menores costos de ingeniería para una amplia gama de audio, comunicaciones y aplicaciones multimedia.
- ✓ Mercado Automotriz: Soluciones diversas para sistemas de asistencia al conductor, confort, de información y entretenimiento en el vehículo.
- ✓ Electrónica de Consumo: Soluciones rentables que permitan actualizaciones de producto con campo en aplicaciones de consumo tales como teléfonos móviles inteligentes, pantallas planas digitales, dispositivos de información, redes domésticas y decodificadores residenciales.
- ✓ Centro de Datos: Diseñado para grandes anchos de banda, servidores de baja latencia, redes y aplicaciones de almacenamiento e implementaciones en la nube.
- ✓ Computación de alto rendimiento y almacenamiento de datos: Soluciones para el almacenamiento conectado a red (sistemas NAS - Network Attached Storage), redes para áreas de almacenamiento y servidores.
- ✓ Sector Industrial: Las FPGAs y sus plataformas de diseño específicas para la industria, ámbito científico y área médica, permiten un mayor grado de flexibilidad, tiempo de salida al mercado más rápido y menores costos generales de ingeniería para una amplia gama de aplicaciones, tales como imágenes industriales y de vigilancia, automatización industrial y equipos de imágenes médicos.

2.1. CONCEPTOS FUNDAMENTALES

La invención de los circuitos integrados ha dado lugar al desarrollo de nuevos chips y entre ellos los dispositivos lógicos programables. Un PLD (Dispositivos Lógicos Programables - Programmable Logic Device) es un chip de uso general para implementar alguna circuitería lógica. Incluye un conjunto de elementos de circuitos lógicos que pueden adaptarse de diferentes formas. Un PLD puede considerarse una “caja negra” que contiene compuertas lógicas e interruptores programables, como se ilustra en la Figura 14. Estos interruptores programables permiten que las compuertas lógicas en el interior del PLD se conecten de determinada manera para implementar el circuito lógico que se necesite.

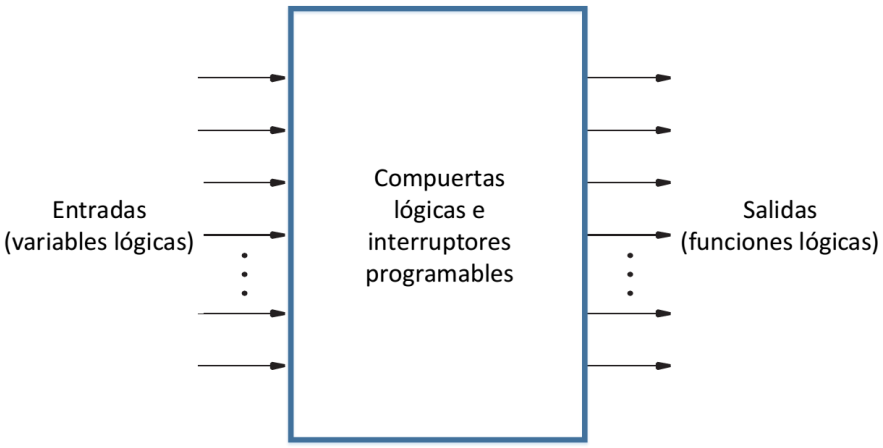


Figura 14. Esquema general de un PLD visto como caja negra

Hay diferentes tipos de PLD comerciales como: los SPLD (Dispositivos Lógicos Programables Simples - Simple Programmable Logic Device), los CPLD (Dispositivos Lógicos Comerciales Complejos - Complex Programmable Logic Device) y los conocidos como los FPGA.

2.1.1 PLA

El primero en desarrollarse de la familia de los SPLDs fue el PLA (Arreglo Lógico Programable - Programmable Logic Array), cuya estructura general se presenta en la Figura 15. Basándose en la idea de que las funciones lógicas se pueden realizar en forma de suma de productos, un PLA comprende un juego de compuertas AND que alimenta un conjunto de compuertas OR. Como se muestra en la figura, las entradas del PLA, de x_1 hasta x_n pasa por un grupo de buffers (compuertas que proporcionan tanto el valor verdadero como el complemento de cada entrada) hacia un bloque de circuito llamada matriz de compuertas AND. La matriz AND produce un juego de términos de productos denominados P_1 a P_k, cada uno de los cuales puede configurarse para implementar cualquier función AND con cualquier entrada de x_1 a x_n. Estos términos de productos sirven como entradas a un arreglo o matriz OR, que produce las salidas que van desde f_1 hasta f_m. Cada salida puede configurarse para realizar cualquier suma de P_1,…, P_k y, por ende, cualquier función de suma de productos de las entradas al PLA (formas canónicas del algebra de Boole).

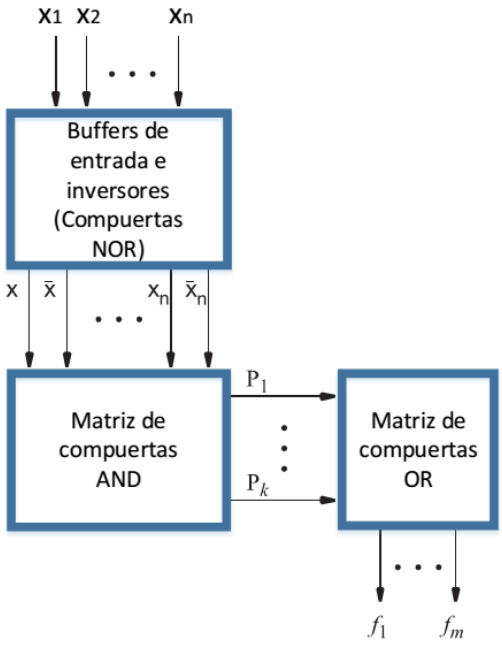


Figura 15. Estructura general de un PLA

Un diagrama más detallado de un pequeño PLA ejemplo se muestra en la Figura 16. El cual tiene tres entradas, cuatro términos de productos y dos salidas. Cada compuerta AND en la matriz AND posee seis entradas, que corresponden a los estados verdaderos y complementados de las tres señales de entrada. Cada conexión a una compuerta AND es programable; la señal que conecta a una compuerta AND se indica con una línea ondulada, y la que no se conecta, con una línea quebrada. Los circuitos se diseñan de tal modo que cualquier entrada no conectada de la compuerta AND no afecta la salida de dicha compuerta.

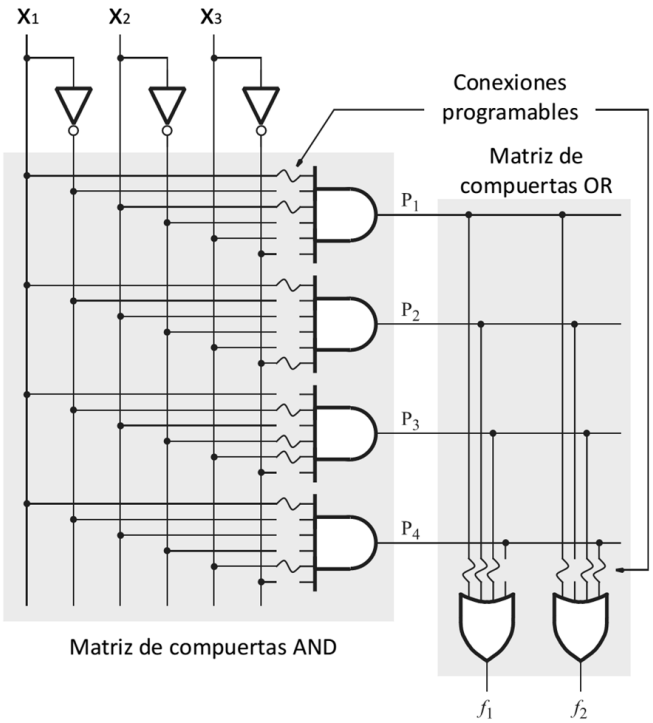


Figura 16. Diagrama a nivel de compuertas de un PLA

El PLA es eficiente en términos del área necesaria para implementarlo en un chip de circuito integrado. Por ello, con frecuencia se incluyen PLA como parte de chips más grandes, como por ejemplo los microprocesadores.

2.1.2. PAL

En un PLA los arreglos AND y OR son programables. Históricamente, los interruptores programables planteaban dos dificultades a sus fabricantes: eran difíciles de fabricar correctamente y reducían la velocidad de rendimiento de los circuitos implementados en los PLA. Tales desventajas llevaron al desarrollo de un dispositivo similar en el que el plano AND es programable pero la matriz OR es fija. Este chip se conoce como PAL (Programmable Array Logic - Matriz Lógica Programable), puesto que son más simples de fabricar y por lo tanto menos costosos que los PLA, aparte de ofrecer mejor rendimiento, las PAL se han vuelto populares en las aplicaciones prácticas. Por ejemplo, con este tipo de dispositivos pueden simularse arreglos del tipo de Productos Lógicos, o en su caso realizar decodificación de direcciones. Sin embargo, la mayor aportación de los dispositivos PAL fue generar aplicaciones específicas en muy corto tiempo. La operación de estos dispositivos inicio manejando velocidades de 4.7 Mhz para la IBM PC hasta 33 Mhz, para posteriormente alcanzar los 50 Mhz; hoy se pueden encontrar dispositivos que operan con un retraso de propagación de la señal del orden de 5ns.

En la Figura 17 se representa un ejemplo de implementación en un dispositivo PAL con tres entradas, cuatro términos producto y dos salidas. La misma se bosqueja como una sola línea horizontal unida a un símbolo de compuerta AND. Las posibles entradas a la compuerta AND se dibujan como líneas verticales que cortan la línea horizontal. En cualquier cruce de una línea vertical y horizontal puede hacerse una conexión programable, la cual se indica con una cruz (×).

Para compensar su menor flexibilidad, las PAL se fabrican de diversos tamaños, con varios números de entradas y salidas, y diferentes números de entradas a las compuertas OR.

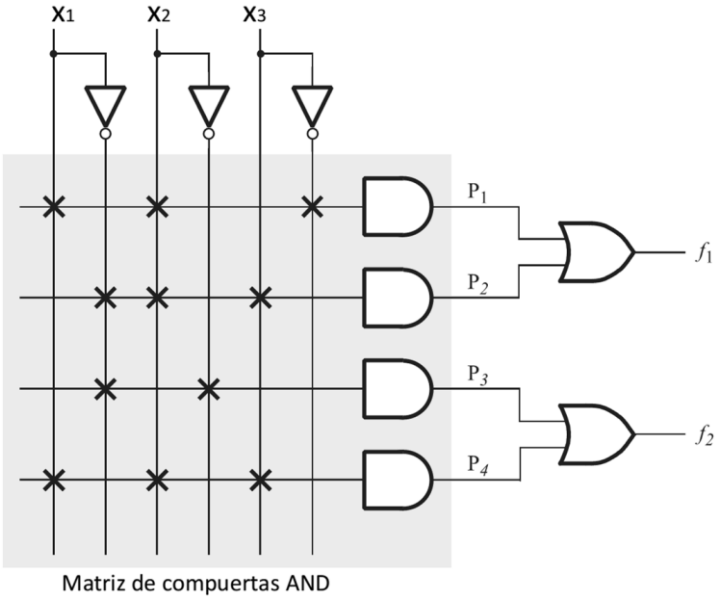


Figura 17. Ejemplo de implementación en una PAL cumpliendo una función determinada

Los SPLDs son útiles para implementar una amplia variedad de pequeños circuitos digitales. Tanto los PLA y los PAL pueden utilizarse para implementar circuitos que no requieren más que el número de entradas, términos producto y salidas que se ofrecen en el chip específico. Estos chips están limitados a tamaños muy modestos, y soportan una cantidad de entradas más salidas que en combinación típicamente no exceden de 32.

Para la implementación de circuitos que precisan más entradas y salidas, se pueden emplear múltiples PLA o PAL o bien un tipo más moderno de chip, llamado CPLD.

2.1.3. CPLD

Un CPLD comprende múltiples bloques de circuitos en un solo chip, con recursos de cableado interno para conectarlos. Cada bloque de circuito es similar a un PLA o a una PAL; nos referimos a ellos como bloques parecidos a PAL. En la Figura 18 se representa un ejemplo de CPLD.

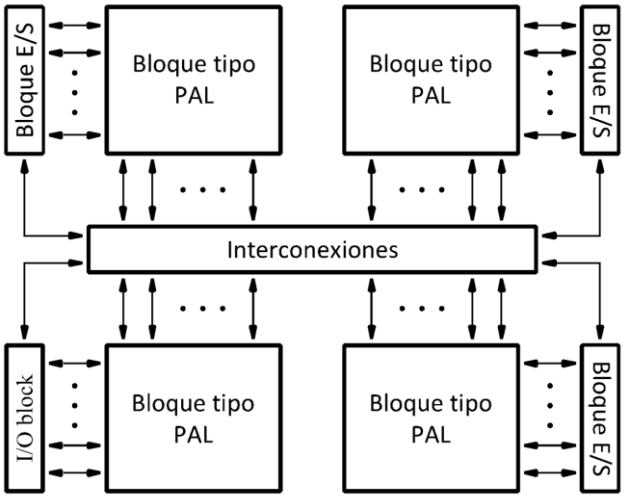


Figura 18. Estructura genérica de un CPLD

En la Figura 18 se incluyen cuatro bloques similares a una PAL que se conectan a un conjunto de vías de interconexión. Cada bloque tipo PAL también está conectado a un sub-circuito etiquetado bloque de entrada/salida (I/O - Input/Output) que a su vez está unido a varios de los pines de entrada y salida del chip.

Los canales de interconexión contienen interruptores programables que se usan para conectar los bloques tipo PAL. Cada uno de los caminos horizontales puede conectarse a alguno de los verticales que cruza.

Los SPLD y CPLD (Brown) son útiles para implementar una amplia variedad de circuitos lógicos. Excepto por el CPLD, estos dispositivos son más bien pequeños y adecuados solo para aplicaciones hasta cierto punto simples. Incluso para los CPLD, nada más pueden acomodarse circuitos lógicos moderadamente grandes en un solo chip. Para implementar circuitos mayores conviene usar un tipo de chip con una capacidad lógica mayor, una FPGA.

2.1.4. FPGA

A mediados de los 80s fueron introducidas la FPGA, las cuales se diferencian de los CPLD en su arquitectura, tecnología y costos (Digilent, 2015). Estos dispositivos fueron creados principalmente para la implementación de circuitos de alto rendimiento.

Las FPGA son muy diferentes de los SPLD y de los CPLD, ya que no contienen arreglos AND y OR. En vez de ello, los FPGA ofrecen bloques lógicos para la implementación de las funciones

requeridas. La estructura general de un FPGA se ilustra en la Figura 19. Contiene tres tipos principales de recursos: bloques lógicos, bloques de I/O para conectar a los pines exteriores, y vías de interconexión e interruptores.

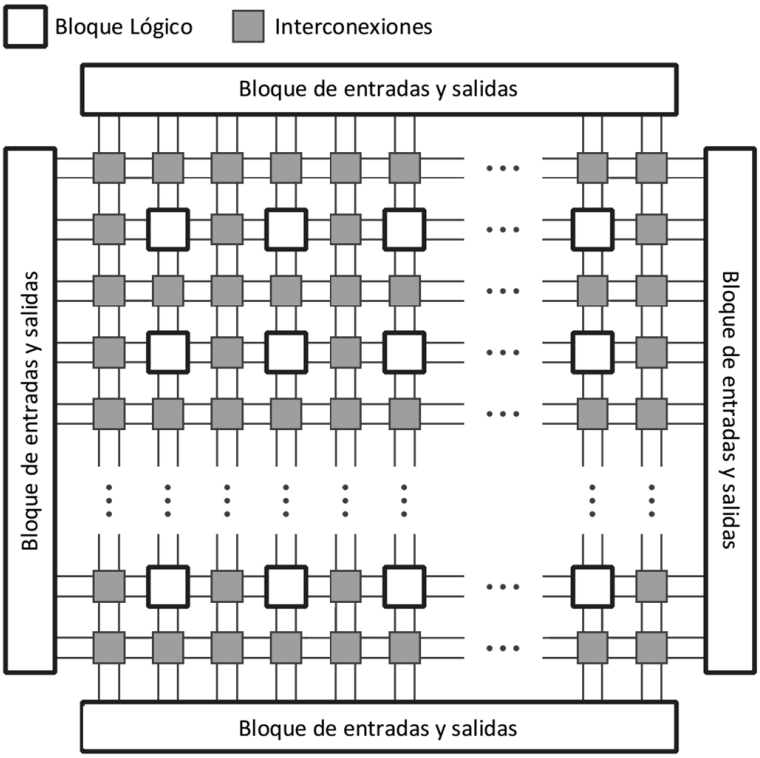


Figura 19. Estructura general de un FPGA

Los bloques lógicos están dispuestos en una matriz bidimensional, en tanto que las vías de interconexión están organizadas como canales de enrutamiento horizontales y verticales entre filas y columnas de bloques lógicos. Los canales de interconexiones contienen hilos e interruptores programables que permiten que los bloques se comuniquen de diversas maneras.

Los FPGA son dispositivos compuestos por ciento de miles, e incluso millones de compuertas lógicas, muy flexibles que pueden trabajar a altas frecuencias y con capacidad de procesamiento en paralelo. Además, presentan recursos especiales para implementar de forma eficiente funciones aritméticas (comparadores, sumadores, contadores, etc.), mientas que los CPLD carecen de estos.

2.2. ARQUITECTURA DE LOS DISPOSITIVOS FPGA

Como se observa en la Figura 19, la estructura interna de un FPGA consiste en arreglos de bloques lógicos que se comunican entre sí a través de canales de interconexión verticales y horizontales.

2.2.1. BLOQUES LÓGICOS CONFIGURABLES

Cada bloque lógico de un FPGA tiene un cierto número de entradas y salidas, estos son capaces de implementar funciones lógicas, tanto combinacionales como secuenciales (Digilent, s.f.).

El tipo de bloque más utilizado es la LUT (Look-Up Table – Tabla de Consulta), que consiste en una memoria del tipo SRAM (Static Random Access Memory -Memoria Estática de Acceso Aleatorio) compuesta por celdas de almacenamiento que sirven para implementar una pequeña función lógica. Estas funciones lógicas se pre-calculan y se almacenan en la memoria de la LUT. Cada celda puede contener un solo valor lógico, 0 o 1, luego dicho valor almacenado es el obtenido a la salida de la celda de almacenamiento. En la Figura 20 se muestra la estructura de una LUT, la cual tiene dos entradas, x_1 y x_2, y una celda de salida que corresponde al valor de B, y es capaz de implementar cualquier función lógica de dos variables (es decir, cuatro celdas de almacenamiento). Las variables de entrada x_1 y x_2 se usan como las entradas de selección de tres multiplexores, los cuales, según sus valores, eligen el contenido de una de las cuatro celdas como la salida de la LUT.

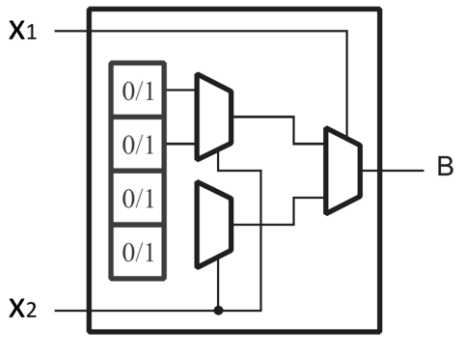


Figura 20. Circuito de una LUT de dos entradas

Generalmente, los FPGA tienen circuitos lógicos extras, además de las LUTs, en cada bloque lógico. Por ejemplo, incluyen flip-flops dentro de algún bloque lógico como se muestra en la Figura 21. La estructura en sí de cada bloque varía de acuerdo al fabricante, por ejemplo Xilinx en su familia Sartan-6, emplea bloques CLBs interconectados matricialmente que cumplen diversas funciones como se muestra en la Figura 22.

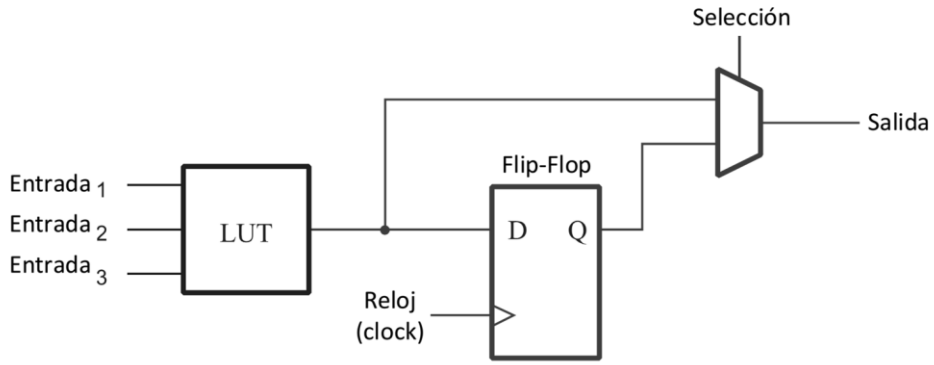


Figura 21. Bloque CLB

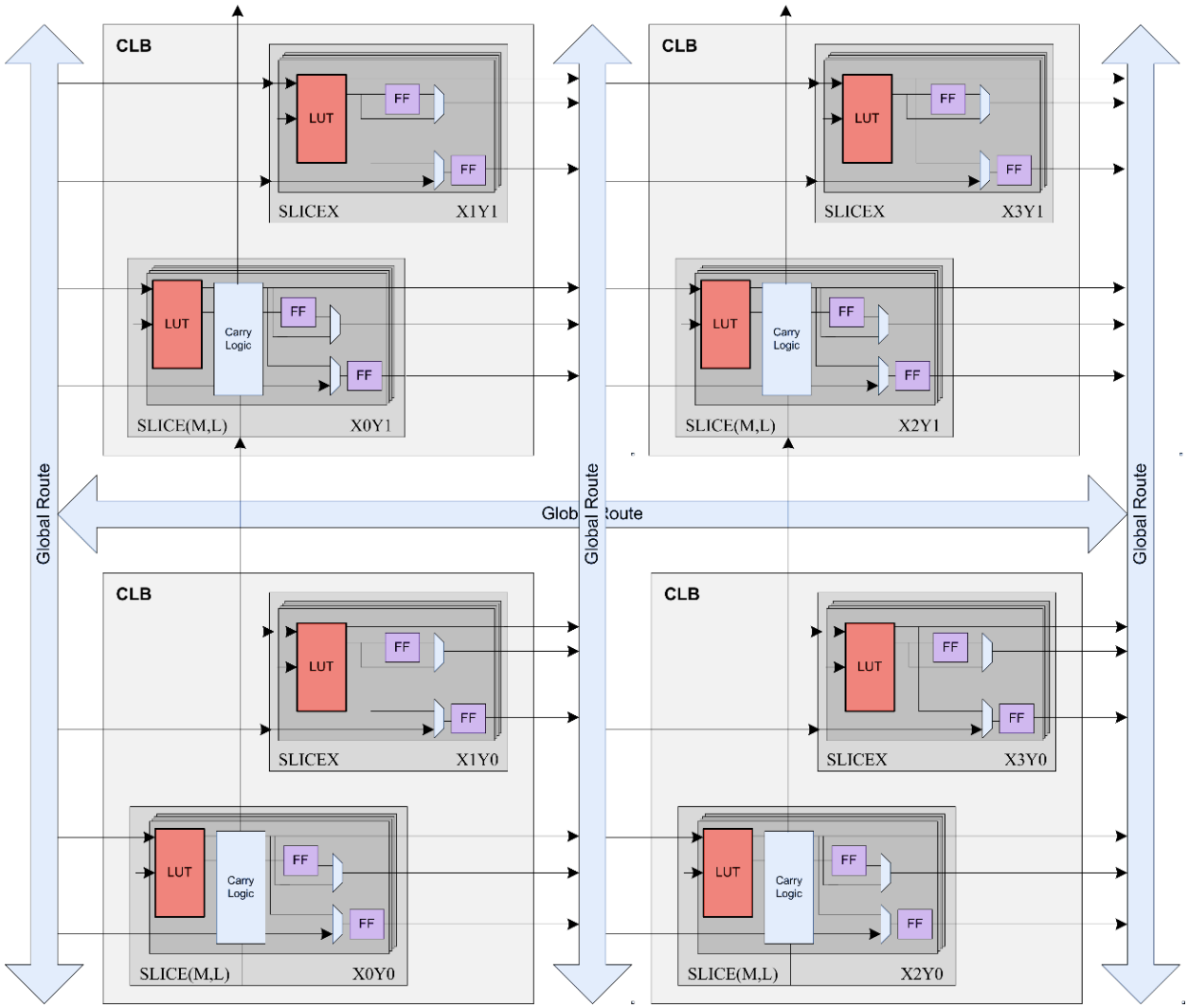


Figura 22. Organización de LUTs y CLBs en la familia Spartan-6 de Xilinx

2.2.2. BLOQUES DE ENTRADA/SALIDA

Estos bloques de entrada/salida (IOB – Input/Output Block) tienen la función de permitir el paso de una señal hacia el interior o hacia el exterior del dispositivo. Cada uno de los bloques disponibles puede ser configurado independientemente para funcionar como entrada, salida o línea

bidireccional. En la Figura 23 se muestra un esquema de un IOB que adopta Xilinx en su familia Spartan (Xilinx Inc., s.f.).

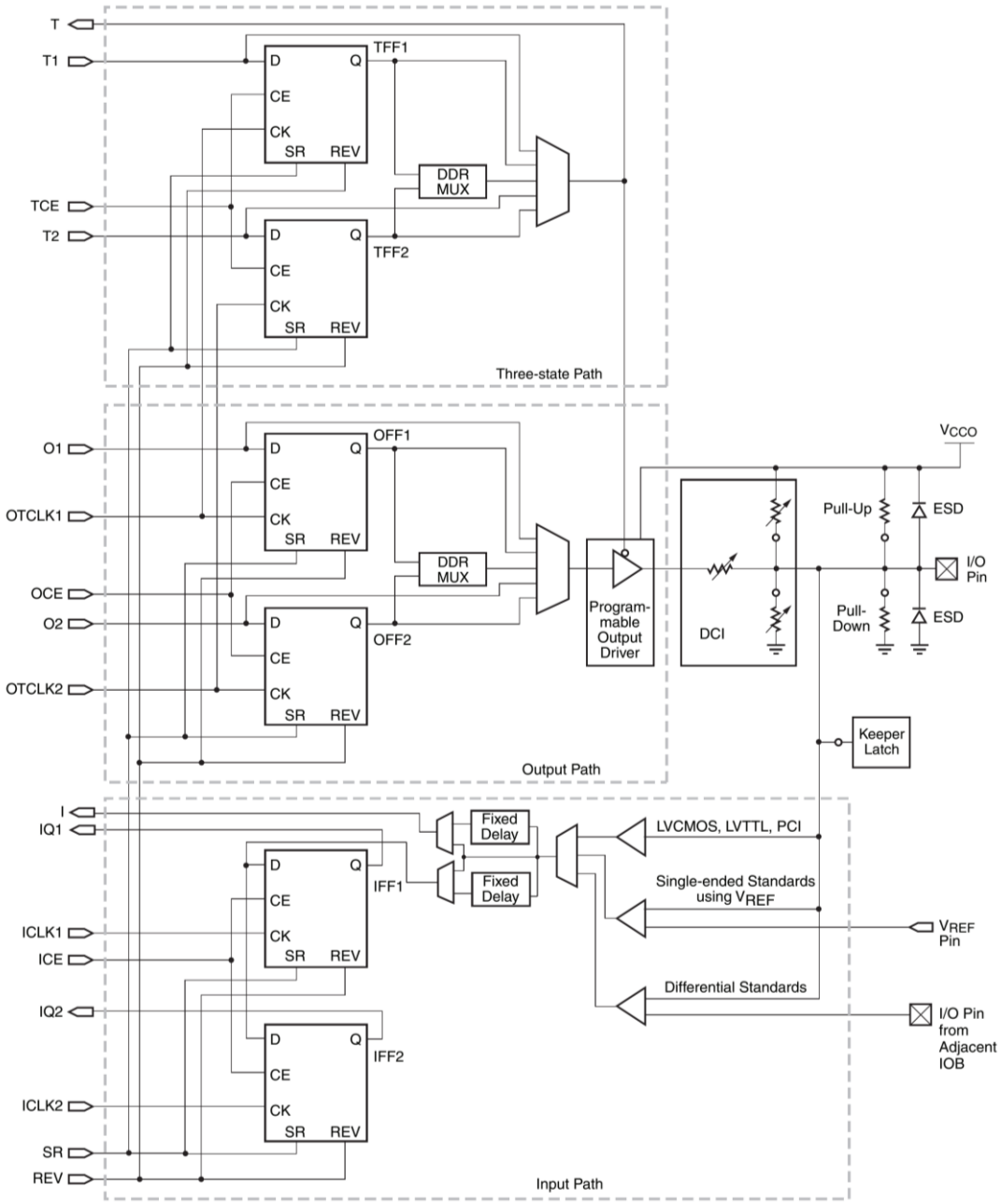


Figura 23. Esquema simplificado de un IOB de la familia Spartan de Xilinx

Dependiendo del fabricante estos bloques pueden ser considerados o no parte del bloque lógico, y se pueden implementar con flip-flops, latches, o circuitos combinacionales y en algunos casos la salida que presenta es triestado (TS).

2.2.3. INTERCONEXIONES

Los canales de interconexión están organizados como vías de enrutamiento horizontales y verticales entre filas y columnas de los bloques lógicos como se muestra en la Figura 24. Los canales de enrutamiento contienen cables e interruptores programables que permiten que los

bloques lógicos se interconecten de muchas formas, además de interconectar a los bloques lógicos entre si también se conectan con los bloques de I/O.

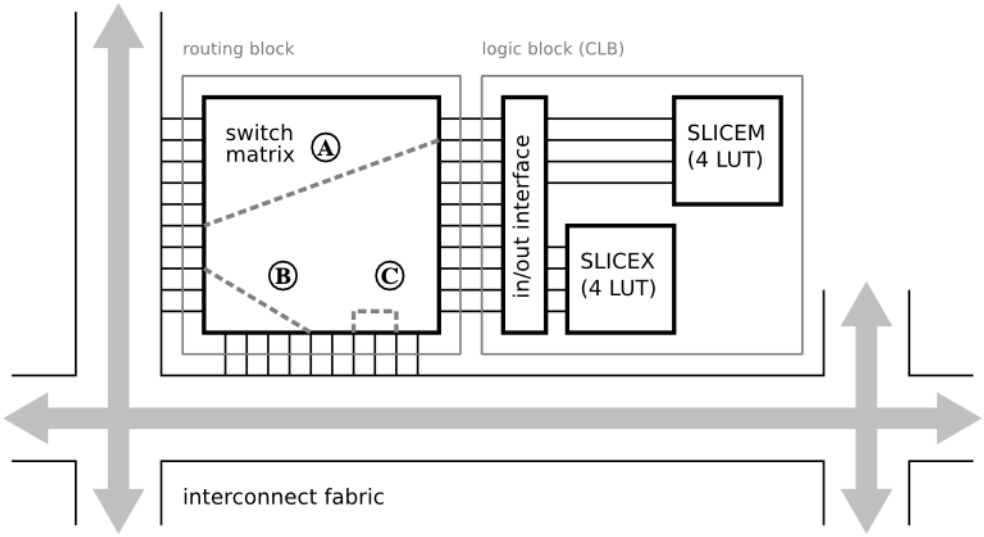


Figura 24. Enrutamiento en una FPGA Spartan de Xilinx

2.3. FORMAS DE DESARROLLO E IMPLEMENTACIÓN

El punto de partida en el proceso de diseñar un circuito lógico es la concepción de lo que se supone debe hacer y del planteamiento de su estructura general. Esto lo efectúa manualmente el diseñador, pues se requiere experiencia de diseño. El resto del proceso de diseño se realiza con el auxilio de las herramientas de diseño asistido por computadora (CAD - Computer-Aided Design), el cual generalmente depende de cada fabricante. La primera etapa de este proceso supone ingresar en el sistema CAD una descripción del circuito que se va a diseñar. Esta etapa se denomina “ingreso de diseño”. Describiremos dos métodos de ingreso de diseño: el uso de lenguaje esquemático o captura esquemática y la escritura de código fuente en un lenguaje de descripción de hardware.

2.3.1. LENGUAJE ESQUEMÁTICO

Un circuito lógico puede describirse dibujando las compuertas lógicas e interconectándolas con cables. Las herramientas de un IDE (Entorno de Desarrollo Integrado) para ingresar el diseño de un circuito de esta manera se llama herramienta de diseño esquemático. La palabra esquemático se refiere al diagrama de un circuito en el que los elementos de éste, como las compuertas lógicas, se muestran como símbolos gráficos y las conexiones entre tales elementos se indican con líneas.

Una herramienta de diseño esquemático utiliza una interfaz gráfica de usuario (GUI - Graphical User Interface) de una computadora y permite al usuario trazar un diagrama esquemático. Para facilitar la inclusión de compuertas en el esquema, la herramienta ofrece un juego de símbolos gráficos que representan compuertas de varios tipos con diferentes números de entradas. Este juego

de símbolos se llama biblioteca o librería. Las compuertas de la biblioteca pueden importarse al esquema del usuario, y la herramienta brinda una forma gráfica de interconectarlas para crear un circuito lógico.

Es posible representar sub-circuitos como símbolos gráficos e incluirse en el esquema. En la práctica es común que el usuario de un sistema CAD cree un circuito que comprenda otros circuitos más pequeños. Este método se conoce como diseño jerárquico y provee una buena forma de manejar la complejidad propia de los circuitos grandes.

2.3.2. LENGUAJES DE DESCRIPCIÓN DE HARDWARE

El diseño del circuito o sistema comienza de la forma más generalizada posible, habitualmente con una especificación algorítmica en un lenguaje de alto nivel. En cada nivel se refina la descripción correspondiente y se compara con la del nivel superior. Este método implica la utilización de herramientas CAD para la simulación y prueba de cada nivel, y por ello los lenguajes deben proporcionar compatibilidad y consistencia entre niveles. De esta manera se trabaja de forma flexible y se logra que en un determinado momento coexistan descripciones de diferentes niveles que se pueden probar conjuntamente.

De lo mencionado anteriormente se deduce la necesidad de que exista un lenguaje que permita utilizar el mayor rango de herramientas posibles y combinar diferentes niveles de descripción de acuerdo con el estado de desarrollo de un proyecto. Así nace el interés por establecer un conjunto de reglas que permitan facilitar el dialogo entre los diseñadores, entre las herramientas CAD y entre los diseñadores y herramientas. Este conjunto de reglas recibe el nombre de Lenguaje de Descripción del Hardware (HDL – Hardware Description Level). Actualmente los HDL de uso más extendido son Verilog (IEEE SA , 2001) y VHDL (IEEE SA, 2019), estandarizados por el IEEE (Instituto de Ingeniería Eléctrica y Electrónica).

No existe unanimidad en lo que se refiere al establecimiento de distintas fases de un diseño digital complejo, aunque la necesidad de sistematizar esta metodología ha llevado al establecimiento y progresiva aceptación de algunas propuestas. Una propuesta ampliamente aceptada divide el diseño en cinco formas de caracterización, también denominados niveles de abstracción:

- Nivel SITEMA: Describe el sistema como un conjunto de módulos semiautónomos y cooperantes, cuya interacción se analiza para obtener un conjunto óptimo. No se especifica la forma de realizar cada uno de los módulos.
- Nivel ALGORITMO: También denominado funcional o de comportamiento. Cada módulo de nivel superior se define mediante un algoritmo en un lenguaje de algo nivel (HLL – High Level Programming Language). Dada la naturaleza paralela del hardware, en el que

varios procesos pueden necesitar un mismo recurso (por ejemplo, un bus de datos) o en el que se deben sincronizar procesos independientes, se utilizan instrucciones muy similares a las de los lenguajes de programación concurrentes, como, por ejemplo, semáforos y regiones críticas.

- Nivel RTL (Register Transfer Level): En él se describe el sistema mediante diagramas de transferencias entre registros, tablas de verdad o ecuaciones lógicas. Se le concede más importancia a lo que hace el sistema que a cómo lo hace. Los elementos básicos de este nivel son registros, memorias, lógica combinacional y buses. Se distingue entre elementos con capacidad de almacenamientos y elementos sin ella.
- Nivel LOGICO: Consiste en la descripción del sistema mediante la interconexión de bloques básicos, como puertas lógicas y biestables. No se realiza una descripción del comportamiento, sino de la estructura del mismo. Si se incluyen además otro tipo de bloques, en general a este nivel se le suele denominar “estructural”.
- Nivel COMPUERTA: Estos dispositivos ideales toman los valores cero o uno. La descripción es, por lo tanto, básicamente la misma que en nivel anterior, con la única diferencia de que, en algunas tecnologías, como por ejemplo CMOS, surgen nuevos tipos de circuitos, que pueden ser bidireccionales.

A continuación, se describen las características funcionales más importantes de los HDL:

- Multinivel: Capacidad para utilizar el rango más amplio posible de niveles de especificación y combinarlos dentro de una misma descripción para ser utilizados junto con las herramientas de diseño.
- Inteligible: Facilidad de lectura y comprensión para simplificar la documentación.
- Capacidad de descripción: Facilidad y potencia para describir los distintos elementos que forman parte de un sistema digital, esta característica es un general fruto de un compromiso.
- Estructura sintáctica: Los HDL deben poseer un conjunto de elementos que definan una sintaxis independiente del nivel. Por ejemplo, los módulos (module) del lenguaje Verilog o el conjunto entidad-arquitectura (entity-architecture) de VHDL. Además, deben poseer instrucciones dependientes del nivel de descripción en que se encuentre el diseño.
- Capacidad para diseño físico estructurado: Deberían, asimismo, facilitar la especificación de ciertos detalles de la realización física en una fase temprana del diseño. Un ejemplo de ello es la planificación de la disposición geométrica de las máscaras, o canales de las celdas de entrada, salida o bidireccionales. Los HDL actuales no poseen esta característica.

- Independencia tecnológica: Un HDL debe ser independiente de la tecnología en la que se realice el circuito.
- Universalidad: Debe, asimismo, ser compatible con el mayor número posible de herramientas de automatización del diseño electrónico (EDA – Electronic Design Automation), como, por ejemplo, simuladores, generadores de patrones de prueba, etc.
- Capacidad de simular la concurrencia: Dado que los componentes de los sistemas digitales están activos simultáneamente, la concurrencia es un aspecto fundamental a tener en cuenta por los HDL. Ello se traduce en que las sentencias de asignación, descripciones de componentes, instanciación de los mismos, etc., han de ejecutarse de tal manera que parezcan ejecuciones simultáneas. Para ello se introdujo en VHDL y en Verilog el concepto de retardo delta.

VHDL: Es un lenguaje de descripción de hardware de gran generalidad derivado del lenguaje de alto nivel ADA. Dispone de tipos abstractos para definir el formato y valores de señales, variables, constantes, etc., y proporciona amplias facilidades para la realización de algoritmos.

Admite casi todos los niveles de descripción comentados en los apartados anteriores, desde el algorítmico hasta el lógico. Para ello proporciona herramientas semánticas y sintácticas que se pueden agrupar así:

Verilog: El lenguaje Verilog es un HDL utilizado para modelar sistemas electrónicos: soporta el diseño, prueba e implementación de circuitos analógicos, digitales y de señal mixta a diferentes niveles de abstracción.

Los diseñadores de Verilog desarrollaron este lenguaje con el fin de obtener un estilo de sintaxis similar a la del lenguaje de programación C, de tal manera que le resultara familiar a los investigadores e ingenieros y así fuera rápidamente aceptado.

El modelado que construye VHDL y Verilog cubre un espectro ligeramente diferente en términos de niveles de abstracción, Figura 25, aunque generalmente el diseñador se basa mayormente en sus preferencias personales al hacer la elección del lenguaje que utilizará.

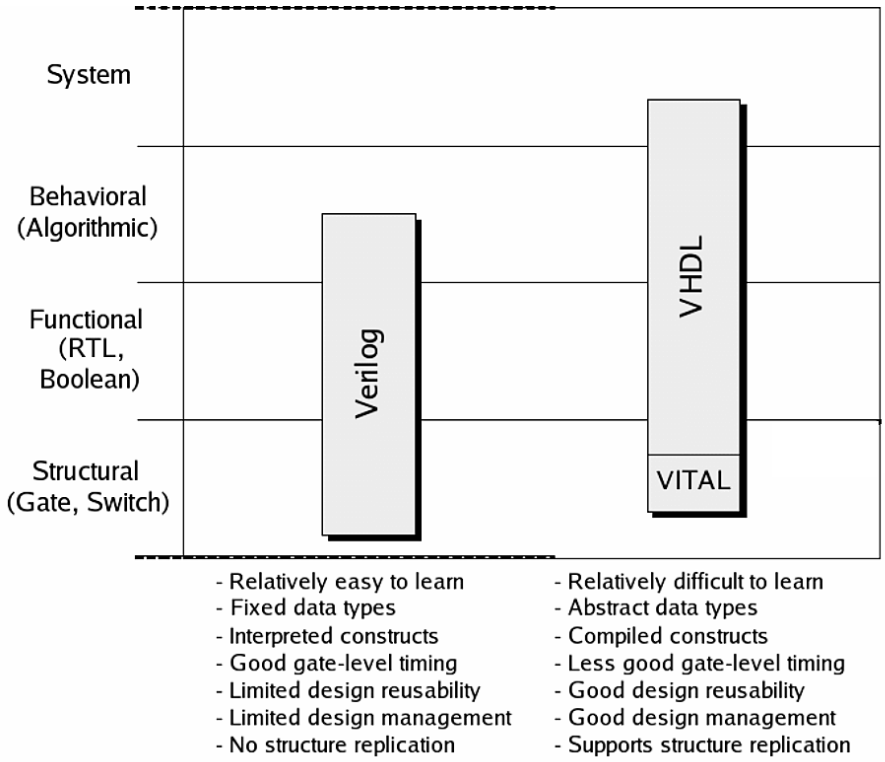


Figura 25. Niveles de abstracción entre Verilog y VHDL. Extraído de (Maxfield, 2004).

2.3.3. SUITE DE DESARROLLO ISE

El Entorno Integrado de Síntesis (ISE - Integrated Synthesis Environment) es una herramienta de software producido por Xilinx para la síntesis y el análisis de diseños de HDL, lo que permite al desarrollador sintetizar sus diseños, realizar análisis de tiempos, examinar los diagramas RTL, simular las reacciones de un diseño a diferentes estímulos, y configurar el dispositivo objetivo FPGA con el programador. El flujo de diseño se puede ver en la Figura 26.

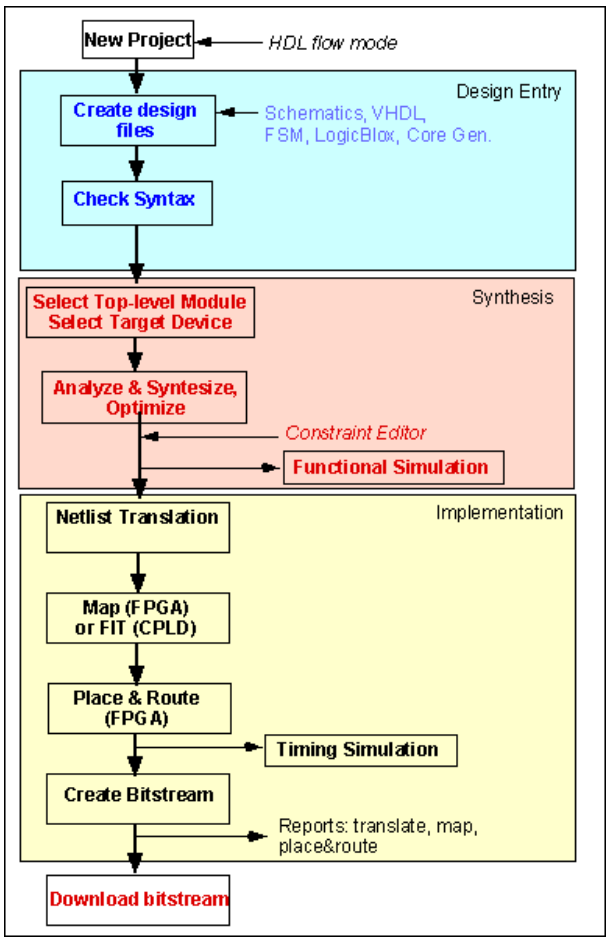


Figura 26. Flujo de diseño utilizando ISE. Extraído de (Xilinx Inc., s.f.)

El ISE, Figura 27, es un entorno de diseño para los dispositivos FPGA, y está optimizado para la arquitectura de circuitos integrados introducidos por Xilinx, y no se puede utilizar con productos FPGA de otros proveedores (Xilinx Inc., s.f.). Otros componentes adosados al ISE y provistos por Xilinx pueden ser: el Kit de Desarrollo Integrado (EDK - Embedded Development Kit) y un Kit de Desarrollo de Software (SDK - Software Development Kit) entre otros.

La suite de diseño ISE incluye además un gran repositorio de núcleos de propiedad intelectual (IP core - Intellectual Property Core) de hardware ya desarrollados (algunos son de uso comercial y otros de forma libre) incluyendo al procesador MicroBlaze.

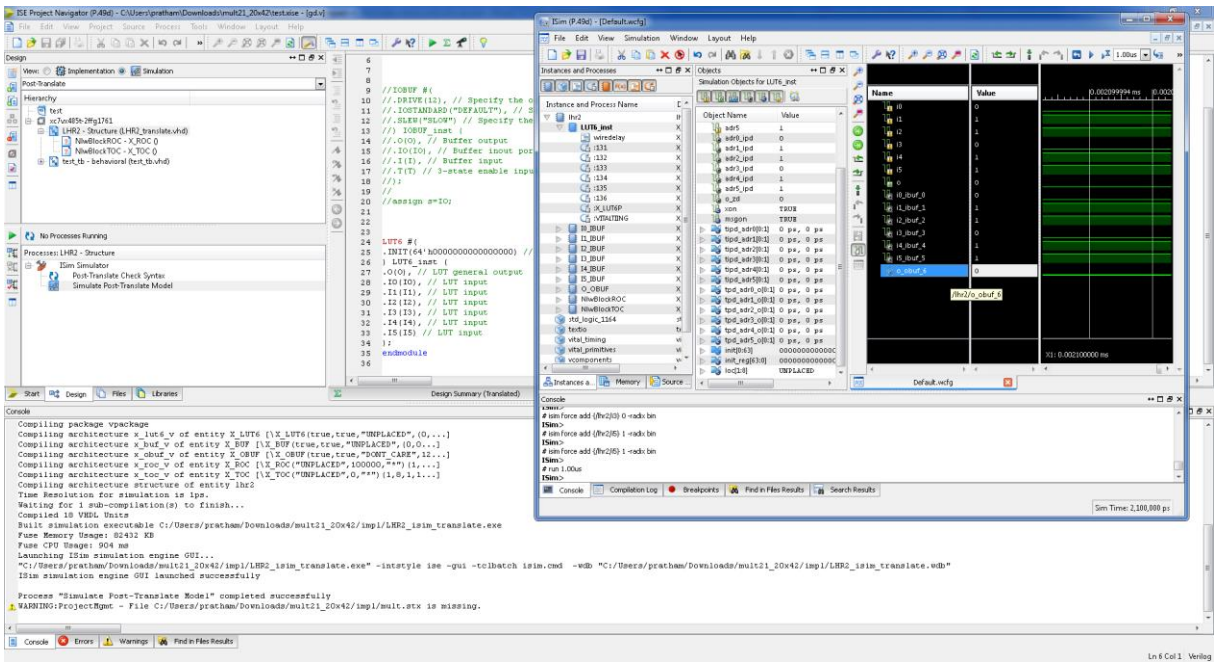


Figura 27. Vista de un proyecto utilizando la Suite ISE de Xilinx. Extraído de (Xilinx Inc., s.f.).

2.3.4. FLUJO DE IMPLEMENTACIÓN PARA ISE

El flujo de diseño para ISE implica a cumplir los siguientes pasos. Ver Figura 26.

1. Creación del diseño: Es posible realizarlo mediante el uso de VHDL/Verilog en el entorno de ISE o mediante otros entornos como por ejemplo MATLAB (MathWorks, s.f.). Se pueden incluir otros archivos HDL, modelos o esquemáticos, procesadores embebidos, módulos de procesamiento digital de señales (DSP).
2. Síntesis: Durante la síntesis, el motor de software para realizar la misma, compila el diseño para transformar los archivos fuentes HDL en una "netlist" (lista de conexiones) para el diseño específico de la arquitectura FPGA. El software ISE admite el uso de XST (Xilinx Synthesis Technology) que se utiliza en este trabajo para realizar esta tarea.
3. Implementación: Después de la síntesis, se ejecuta la implementación del diseño, que convierte el diseño lógico en un formato de archivo físico que se puede descargar al dispositivo de destino seleccionado, es decir, grabar en la FPGA. La implementación comprende las siguientes etapas:
    a. Translate (traducir): Fusiona las listas de conexiones y especificaciones de diseño adicionales en un archivo de diseño Xilinx®.
    b. Map (mapeo o partición): Ajusta el diseño a los recursos disponibles del dispositivo destino. Es la fase en la que el sistema objeto del diseño se divide en módulos, a cada uno de los cuales se le asigna un recurso lógico del circuito FPGA.
    c. Place and Route (ubicación y enrutado o interconexión): Coloca y establece rutas internas de conexiones en base a las "constrains" del diseño. Una vez realizada la

asignación de los módulos que forman parte del sistema es necesario llevar a cabo la ubicación, es decir, su ubicación en una parte determinada del circuito FPGA y proceder a su interconexión o enrutado. Como la interconexión depende de la ubicación, ambas fases están, en la práctica, interrelacionadas y la segunda fase está en indefectiblemente ligada a la primera.

Es conveniente resaltar que tanto la partición como la ubicación y el enrutado dependen de la arquitectura y de los recursos lógicos disponibles en el circuito FPGA utilizado y, por ello, cada fabricante ha desarrollado sus propias herramientas de diseño. Existen también herramientas de diseño genéricas que utilizan bibliotecas de elementos de varios fabricantes.

d. Generate Programming File (generar archivo de programación): Crea un archivo binario (con el patrón de programación y a la configuración del circuito FPGA) que se puede descargar al dispositivo, en otras palabras, es el archivo que se grabará finalmente en la FPGA.

2.3.5. LIBRERÍA SYSTEM GENERATOR PARA FPGAS XILINX

El System Generator (Xilinx Inc., s.f.) es una herramienta específica de modelado de sistema para el diseño de hardware FPGA. Se utiliza por medio del software Simulink® de MATLAB® proporcionando un entorno completo de diseño, desarrollo e implementación de hardware. La herramienta proporciona un alto nivel de abstracción que puede compilar automáticamente o mediante scripts[3] en un dispositivo FPGA. La herramienta también proporciona acceso a los recursos FPGA subyacentes a través de abstracciones de bajo nivel, lo que permite la construcción de diseños FPGA de alta eficacia.

Flujo de diseño utilizando Sytem Generator: El System Generator puede ser utilizando de diversas maneras, como ser, explorar un algoritmo específico sin trasladar dicho diseño a hardware o utilizar un diseño de un gran sistema y evaluar el comportamiento en el hardware FPGA, de lo cual se deducen tres posibilidades.

Evaluación de algoritmos: Mayormente se utiliza System Generator para la evaluación de algoritmos, realización de prototipos de diseño y análisis de modelos. Cuando estos son los objetivos, la herramienta sirve para dar cuerpo a un algoritmo con la finalidad de conocer a priori los problemas de diseño que probablemente se hallarán, y estimar el rendimiento de una

[3] Un script es un archivo con extensión .m que contiene varias líneas secuenciales de comandos y llamadas a funciones de MATLAB.

implementación en hardware. Generalmente el trabajo es de preparación y análisis, y hay poca necesidad de traducir el diseño en el hardware.

Implementar una parte de un diseño general: A menudo, System Generator se usa para implementar un sub-sistema, dentro de un sistema general. Por ejemplo, System Generator es un buen escenario para ensayar control de datos de un trabajo dado, y luego combinar las partes en un todo.

Implementar un diseño completo: Muchas veces, todo lo necesario para un diseño, se encuentra disponible en las librerías de System Generator. Para este diseño, el System Generator "genera" la traducción en HDL, y escribe los archivos necesarios para implementar el HDL en el hardware FPGA especifico incluyendo:

- El HDL generado implementa el diseño completo en sí mismo.
- Un HDL para realizar testbench (banco de pruebas), que resulta de simulaciones realizadas por Simulink® para ser luego contrastadas con las producidas por el simulador lógico.
- Archivos que permiten al HDL ser utilizados por otras herramientas como el ISE.

El System Generator permite que los diseños de hardware específicos del dispositivo sean construidos directamente en un entorno flexible de modelado de sistemas en alto nivel de abstracción. En un diseño de System Generator, las señales no son solamente bits, sino también que puedan ser números de punto fijo, valores signados o sin signo, y los cambios realizados en el diseño se traducen automáticamente en los cambios apropiados en los tipos de señales. Los bloques no son sólo sustitutos de hardware, sino que responden a su entorno general, ajustándose automáticamente los resultados que estos producen y el hardware en que se trasforman.

Esta librería permite diseños compuestos por una variedad de ingredientes: modelos de flujo de datos, lenguajes tradicionales de diseño de hardware (VHDL y Verilog) y las funciones derivadas del lenguaje de programación MATLAB®. El System Generator otorga resultados de la simulación de forma precisa y en bits, esto significa que los resultados observados en la simulación coinciden exactamente con los resultados que se observan en hardware. Las simulaciones en System Generator son considerablemente más rápidas que las de los simuladores de HDL tradicionales, y los resultados son más fáciles de analizar.

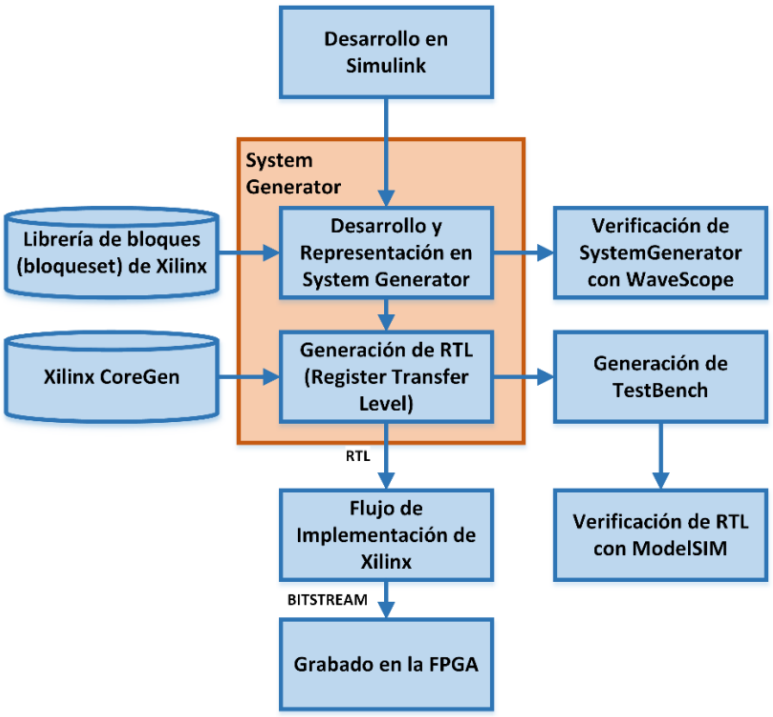


Figura 28. Flujo de diseño utilizando las herramientas de System Generator

Cuando se instala System Generator, automáticamente se incluye toda una extensa librería (blockset) en Simulink® de Matlab®, ver Figura 29. Básicamente, el System Generator permite reducir el tiempo dedicado por el diseñador para la descripción y la simulación del circuito. Por otra parte, el diseño es flexible; es posible cambiar los parámetros de diseño y comprobar rápidamente el efecto en la performance y la arquitectura del sistema. La simulación funcional es posible, incluso antes de la compilación del modelo diseñado. La compilación genera los archivos de la descripción estructural del sistema en un lenguaje de descripción de hardware estándar para el entorno ISE de Xilinx.

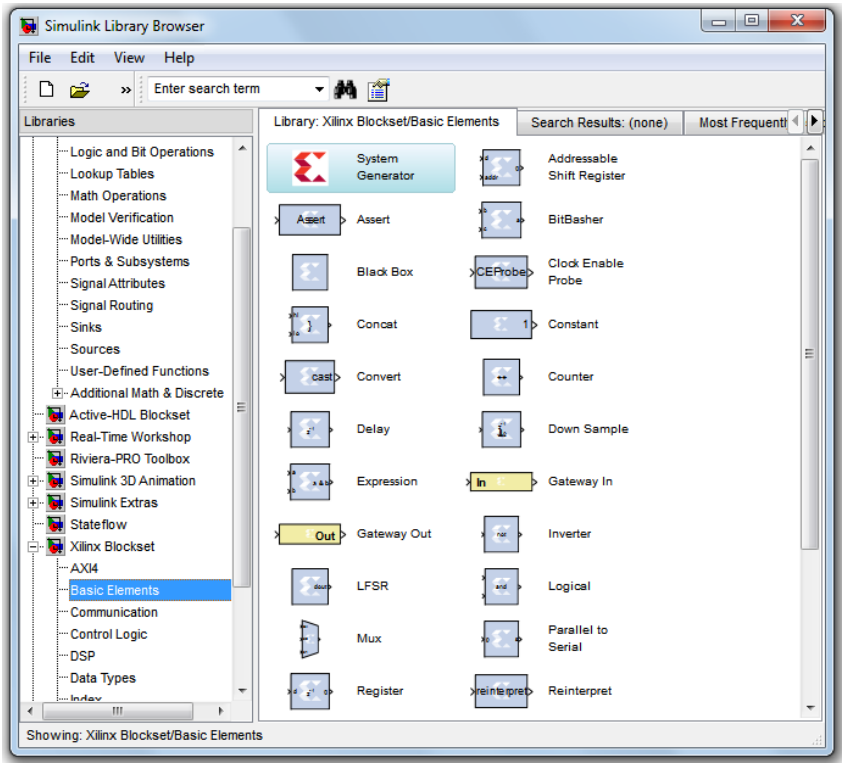


Figura 29. Vista de los bloques que componen la librería System Generator de Xilinx. Extraído de (Xilinx Inc., s.f.).

El límite entre el modelado de hardware que se transfiere a la FPGA y el modelado propio de Simulink® se define por medio de bloques "Gateway In" y "Gateway Out", ver Figura 30. El bloque "Gateway In" convierte una entrada de datos del tipo punto flotante de Simulink® a un formato de punto fijo, con parámetros como saturación y redondeo definidos por el diseñador. El bloque "Gateway Out" convierte el formato de punto fijo del modelo de hardware de la FPGA al formato numérico de Simulink® de punto flotante con doble precisión.

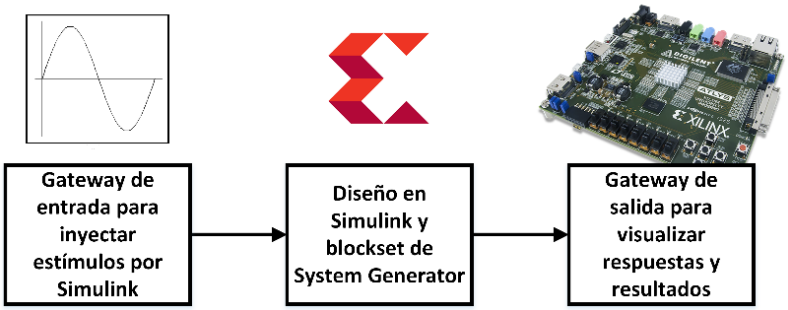


Figura 30. Concepto de los Gateways de System Generator

En el System Generator, el diseñador no percibe las señales como bits, sino que los bits se agrupan en formato de datos con punto fijo signados o no. Uno puede fabricar un bloque en MATLAB con System Generator y este no necesariamente es un circuito de hardware; sino que se relaciona con otros bloques para generar el hardware adecuado. El diseñador puede incluir bloques descriptos

en un lenguaje de descripción de hardware, diagramas de flujo de máquinas de estados finitos, archivos de Matlab®, etc.

Cuando un diseño es generado por medio de los bloques estándares de Simulink® (ver Figura 29), puede ser utilizando una precisión numérica de punto flotante y no los detalles de hardware. Una vez que se han definido la funcionalidad y el flujo de datos de base, el System Generator puede ser utilizado para especificar los detalles de implementación de hardware de los dispositivos de Xilinx.

El System Generator utiliza una librería especifica de DSP (DSP Blockset) de Simulink® e invocará automáticamente al Xilinx Core Generator para generar listas de conexiones (netlist) altamente optimizadas para unir los diferentes bloques del diseño. Del System Generator se puede obtener un archivo de flujo de bits para la implementación en hardware y programar el dispositivo FPGA. A su vez, permite la creación automática de un banco de pruebas (testbench) utilizando parámetros extraídos del entorno de Simulink® compatibles con los simuladores del entorno ISE de Xilinx.

2.4. PLACAS DE DESARROLLO XILINX

Una tarjeta educacional FPGA consta de varios elementos y dispositivos soldados a la placa base como en un sistema embebido, dependiendo del fabricante podemos encontrar kits de desarrollo con diferentes características dependiendo el propósito al que se le vaya a otorgar.

Como mencionamos se analizarán los kits de desarrollo que presten un entorno adecuado para prácticas de laboratorio haciendo énfasis a lo dispuesto anteriormente.

2.4.1. ATLYS

El kit de desarrollo Atlys, ver Figura 31, es una plataforma completa lista para el diseño e implementación de circuitos digitales en FPGA basada en la familia Spartan-6 de Xilinx. Los circuitos en la placa base son periféricos de gama alta, incluyendo un puerto Ethernet de Gigabit, interface HDMI, matriz de memoria DDR2 de 128 Mbyte, puertos de audio y USB que hacen que el módulo Atlys sea el anfitrión ideal para sistemas completos diseñados con procesadores embebidos y el co-diseño de entornos complejos.

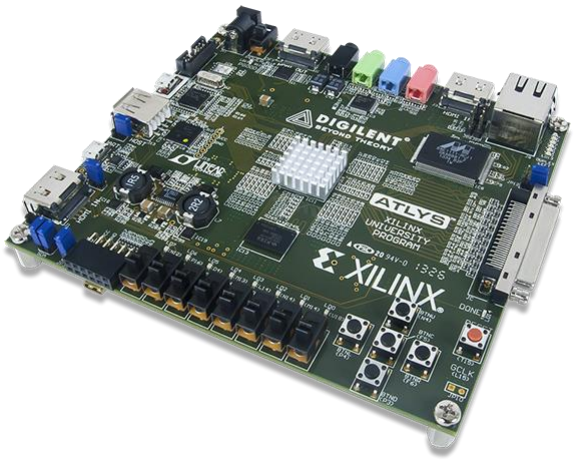


Figura 31. Placa de desarrollo Atlys con una FPGA Spartan-6 de Xilinx

El Spartan-6 esta optimizado para la lógica de alto rendimiento y este kit ofrece:

- 6882 slices, cada una con 4 LUTs de 6 entradas y 8 flip-flops.
- Bloque rápido de RAM de 2.1Mbits.
- Cuatro estilos de "clock" (8 DCMs & 4PLLs).
- 6 PLLs.
- 58 slices para DSP.
- Velocidad de "clock" de hasta 500Mhz.

La plaqueta Atlys incluye además un nuevo sistema de USB 2.0 de la firma Digilent que ofrece características como la programación del dispositivo, monitoreo fuente de alimentación en tiempo real, pruebas de la placa automatizados, E/S virtuales entre otras.

Detalles técnicos:

- FPGA de la familia Spartan-6 de Xilinx con encapsulado BGA de 324 pines.
- Memoria de 128Mbyte tipo DDR2 con ancho de datos en 16 bits.
- Interface PHY (physical layer – capa física) para Ethernet 10/100/1000.
- Puerto USB 2.0 para programación y trasferencia de datos.
- Chip USB a UART.
- Puerto de entrada de video HDMI y dos puertos de salida HDMI.
- Códec integrado AC-97 para entrada y salida de línea de audio, micrófono y auriculares.
- Monitoreo en tiempo real de consumo y tensiones de alimentación.
- Memoria tipo Flash de 16Mbyte x4 SPI para configuraciones y almacenamiento de datos.
- Oscilador de 100MHz tipo CMOS.

- Conectores de expansión enrutados para 48 I/O.
- Entradas y salidas de propósito general que incluyen 8 LEDs, 6 botones y 8 interruptores.

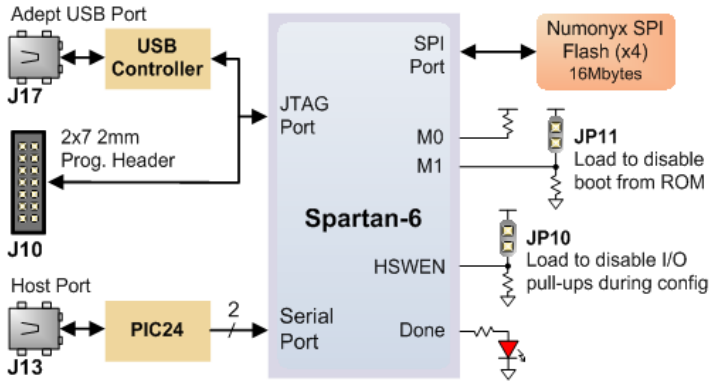


Figura 32. Esquema de puertos disponibles en un kit Atlys

2.4.2. 3PX1

La placa 3PX1 (Emtech, s.f.), ver Figura 33, es una placa de desarrollo basada en la FPGA Spartan 6 de Xilinx. Incluye todos los recursos necesarios para comenzar a desarrollar y probar diseños digitales (núcleos o "cores") sin la necesidad de programadores u otros instrumentos adicionales. Esto la hace ideal tanto para el aprendizaje de lenguajes de descripción de hardware (HDL) y técnicas digitales como para la implementación rápida de sistemas digitales en prototipos.

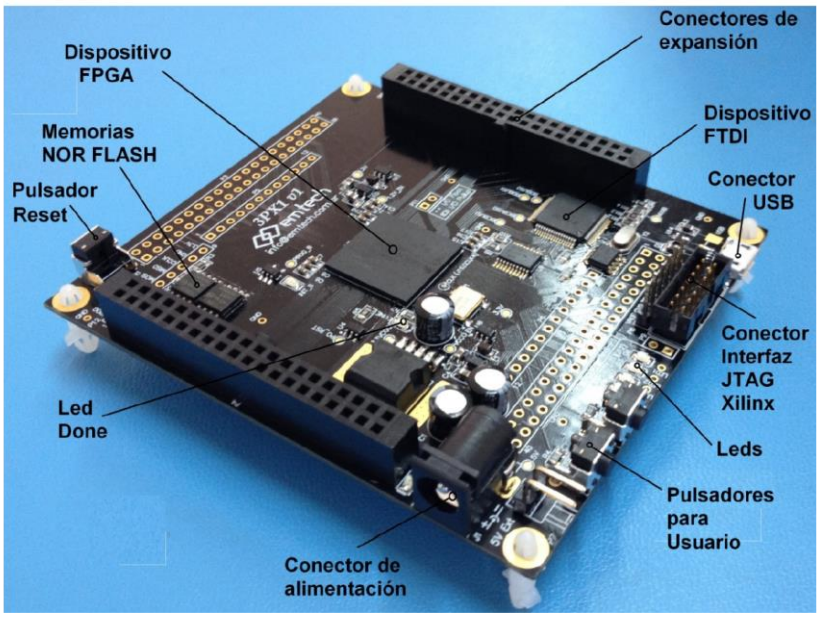


Figura 33. Placa de desarrollo 3PX1 de la firma Emtech que utiliza una Spartan 6 de Xilinx

Detalles técnicos:

FPGA:

- Xilinx Spartan 6 XC6SLX25-2FTG256C

- 24.000 celdas lógicas
- 30.000 flip-flops
- 936 kb de RAM
- 38 bloques DSP

Programación y depuración:

- USB JTAG mediante chip de la firma FTDI

Interfaces

- USB/Serial
- USB/Paralelo
- 148 Entradas / Salidas LVTTL sobre conectores estándar de 0,1”
- Leds y pulsadores

Componentes

- Oscilador de 50 MHz
- FT2232HL
- EEPROM QSPI de 128Mb

Requerimientos de Alimentación

- Adaptador AC/DC de 5VDC @ 1Amp de 2.1mm x 5.5mm
- Mini-USB B

2.4.3. ANÁLISIS COMPARATIVO ENTRE LAS TARJETAS XILINX

No se han tomado en cuenta para este análisis todas las plaquetas o kits de desarrollo Xilinx existentes en el mercado. Tampoco se han analizado otras marcas (por ejemplo, Altera) ya que las plaquetas de Xilinx mencionadas anteriormente se adecuaron a la perfección en este caso de estudio.

Además, estos kits de desarrollo presentados para este desarrollo, prestan los suficientes puertos de expansión a través de distintos productos según la finalidad o aplicación que se le quiera dar.

En este caso, dado que el objetivo final es: diseñar, desarrollar e implementar en un circuito especifico ad-hoc y que ofrezca al laboratorio un equipo adecuando para el diseño y programación en el ámbito FPGA, que cuente con características y recursos suficientes para dar una facilidad a los operadores en cuanto a puertos de comunicación de E/S, elementos visuales de salida para el programador y puertos de expansión. Para realizar las pruebas en hardware independiente uno de otro, se implementó el sistema Adquisidor en el Kit 3PX1 y el Simulador en el kit Atlys.

2.5. FIN DEL CAPÍTULO

Para hacer frente a este desafío se utilizarán las dos plaquetas de desarrollo mencionadas. El modelo incluido en la FPGA deberá contener las nociones básicas de un adquisidor según lo visto en el Capítulo 1. Para lograr esta meta, se realizará un primer diseño conceptual del sistema adquisidor, que contempla el modelo de un simulador también a fin de evaluar el sistema. Luego, se describe el desarrollo en sí de todo el sistema completo y por último las pruebas de campo para evaluar su comportamiento en la realidad dentro de un puesto de tierra, con aplicaciones de software y con datos reales de una electrónica de sensores abordo de un vehículo no tripulado.

CAPITULO 3 – DISEÑO Y DESARROLLO DEL SISTEMA ADQUISIDOR

Una vez que se ha introducido a la tecnología FPGA y a los sistemas de telemetría, se propone como parte del objetivo general de este trabajo desarrollar un algoritmo que permita implementar un sincronizador a nivel de bit de tramas PCM en HDL. Este conjunto tendrá el desafío de ser de alta velocidad, deberá operar en tiempo real y será en tecnología FPGA con bajo costo. El sistema en su totalidad será aplicado a un puesto de tierra con un sistema de adquisición de datos telemétricos ad-hoc para ser aplicado al monitoreo de UAVs y cohetes sonda.

El desarrollo con System Generator permite tener una herramienta de alto nivel para el desarrollo de sistemas de alta performance utilizando los dispositivos Xilinx con tecnología FPGA, y así definir y caracterizar circuitos lógicos para que cumplan una función específica.

La principal diferencia entre cualquier HDL y el resto de los lenguajes de programación es que los lenguajes de descripción se sintetizan, no son compilados ni ejecutados como cualquier otro programa. Esto se debe a que los lenguajes de programación se definen como procedimientos en cambio la descripción de hardware se basa en la definición de comportamientos de acuerdo a las entradas y al procesamiento deseado de forma concurrente.

Durante la síntesis se define la interconexión de los recursos disponibles en la FPGA para que en su conjunto se comporten de la forma descripta, es parte del trabajo de las herramientas de desarrollo llevar a cabo las optimizaciones necesarias para que ocupe menos recursos o que el bloque pueda operar a mayores frecuencias. El System Generator, traduce automáticamente el desarrollo en bloques de un modelo de Simulink en HDL optimizando tiempos y área de la FPGA, y también genera el archivo binario final que se grabará en la misma.

La implementación del desarrollo en hardware del Adquisidor se llevó a cabo en una placa modelo 3PX1 diseñada y ensamblada por la firma Emtech S.A. La misma cuenta con un dispositivo FPGA Spartan 6 modelo XC6SLX25, su arquitectura se puede ver en la Figura 34. Dicha placa cumple con las necesidades básicas para iniciar a desarrollos y prototipos de aplicaciones a sistemas específicos con ésta tecnología, ya que dispone de las fuentes de alimentación empleadas en la operación de dispositivos de este tipo y todas las entradas/salidas accesibles por medio de conectores estándar del tipo “header”. También se puede encontrar en la placa una memoria flash donde guardar el firmware, pulsadores para utilizar como entradas y LEDs para emplear como indicadores de estados. La Figura 35 muestra un esquema de la disposición de componentes electrónicos. Una de las ventajas que llevó a elegir éste módulo en vez de otros disponibles en el mercado en el mismo segmento de bajo costo es que cuenta con una interfaz Serie/USB ya conectada directamente con la FPGA.

La placa se montó sobre un gabinete abierto realizado en acrílico para dar mayor rigidez a la placa, también para poder unificarla como un único módulo con conectores BNC para conexión y desconexión sin poner en riesgo al dispositivo FPGA. A continuación se muestran fotografías del gabinete (Figura 36).

En el panel frontal se colocaron tres conectores, dos con las salidas de Transmisor PCM (línea de datos y de "clock") y un tercer conector con la entrada al adquisidor de Tramas (ver Figura 37).

El diseño de hardware para la FPGA se realizó utilizando Matlab® en conjunto con el System Generator, una herramienta provista por Xilinx para trabajar en dicho entorno y para algunos casos se utilizó "MCode" para poder implementar el lenguaje de MATLAB directamente sobre una FPGA, sin necesidad de programar bajo VHDL. Para más información ver ANEXO – BLOQUES FUNDAMENTALES DEL SYSTEM GENERATOR.

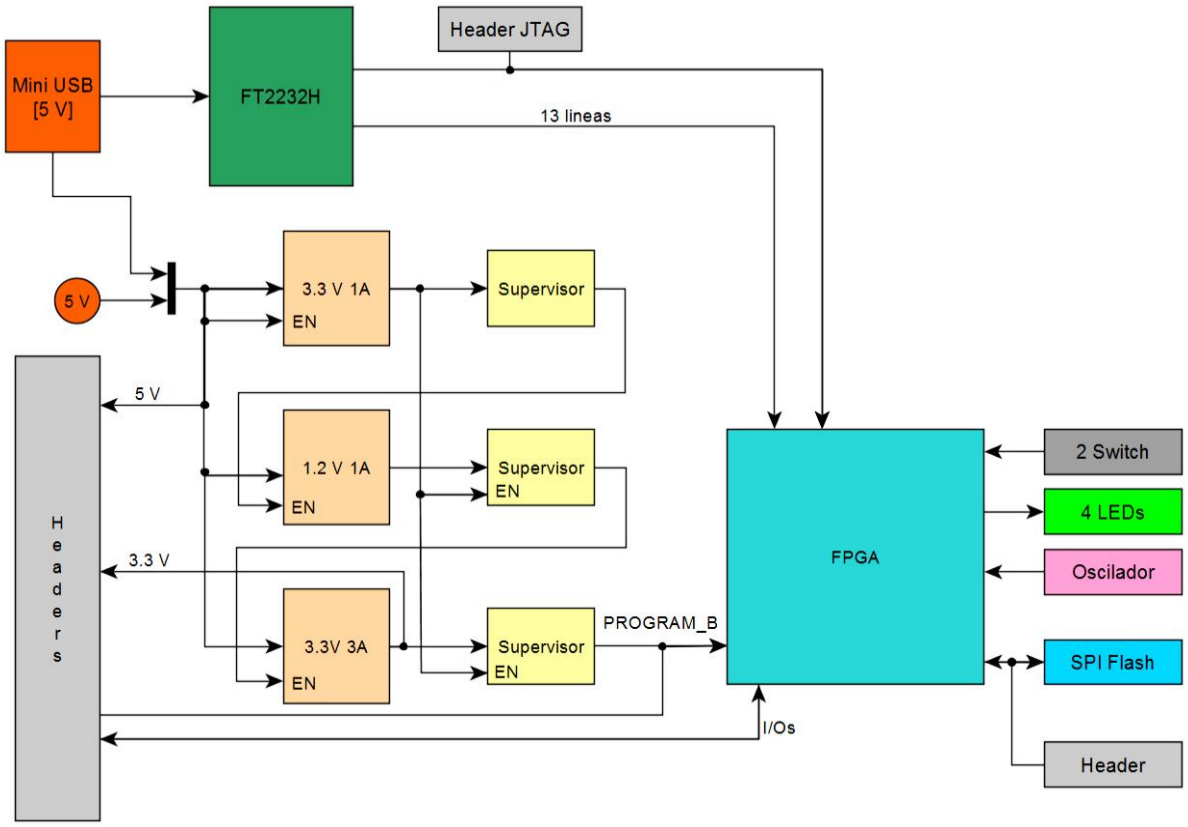


Figura 34. Arquitectura de la plaqueta 3PX1. Extraído de (Emtech, s.f.)

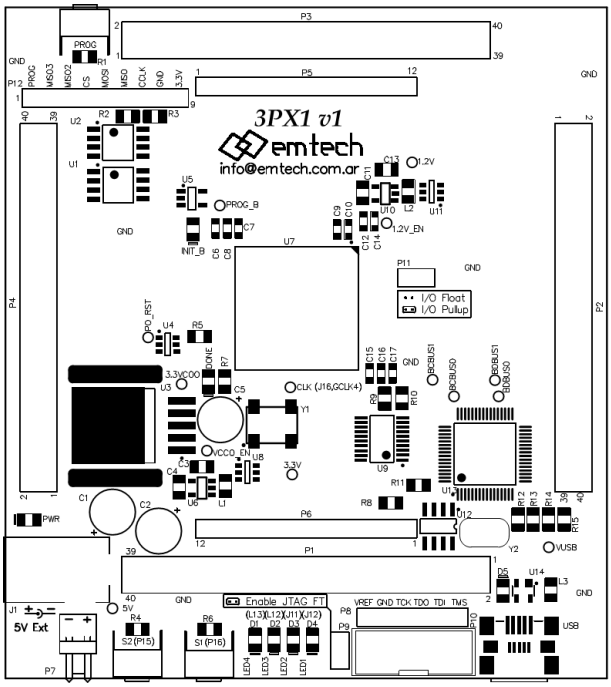


Figura 35. Diagrama de componentes electrónicos de la plaqueta 3PX1. Extraído de (Emtech, s.f.)

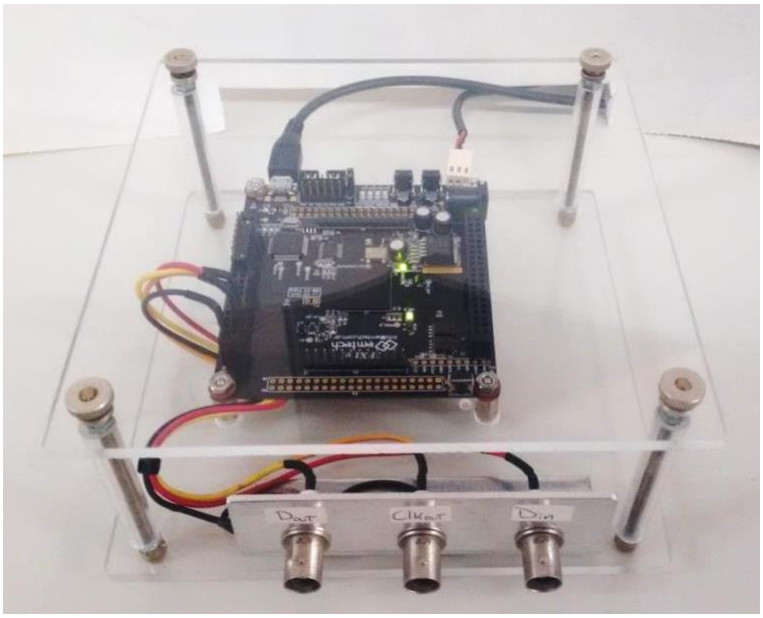


Figura 36. Vista del kit 3PX1

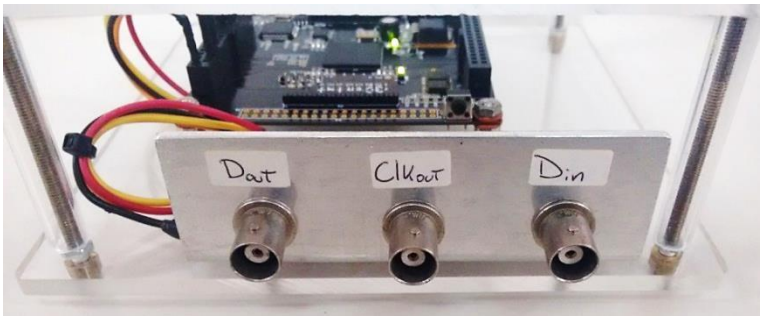


Figura 37. Conectores externos para el desarrollo en el kit FPGA

3.1. SIMULADOR PCM

En todo momento, se debe determinar el estado en que se encuentra un UAV, y para esto es necesario contar con un equipamiento específico en tierra que monitoree y procese la información remota.

Los datos de las magnitudes físicas (mediciones de los sensores) de la aeronave son recibidos a través del sistema de telemetría dentro del puesto de tierra, para ello es necesario contar con un adquisidor de señales telemétricas ad-hoc. Como se ha mencionado en el Capítulo de Telemetría, el sistema adquisidor de datos se denomina “Bit Synchronizer” y es el punto más crítico por encentrarse al frente de toda la cadena de procesamiento de señales. Cualquier error introducido por el Bit Synchronizer se propaga a toda la cadena de procesamiento del puesto de tierra y trae como consecuencia la determinación errónea de los parámetros de interés del vehículo (por ejemplo, el cálculo de navegación y actitud). Este punto amerita ser evaluado de una manera rigurosa.

Para poder someter al Bit Synchonizer a múltiples ensayos es fundamental contar con un módulo que simule la señal esperada, por este motivo es que se diseña un Simulador PCM. Este desarrollo en particular además tiene la característica de probar toda la cadena de módulos que componen el puesto de tierra, desde el adquisidor de señales telemétricas hasta el conjunto de aplicaciones que se ejecutan en el mismo, encargadas de monitorear la información remota de abordo de la plataforma.

Las señales PCM transmitidas son un flujo de datos binarios, el estándar IRIG 106-13 (Telemetry Standards, 2013) define como se debe codificar una estructura de pulsos y las características bajo las que se deben implementar los sistemas de telemetría (Consultative Committee for Space Data Systems, 2000), bajo éste estándar es que se determinaron algunas de las características del simulador. El estándar hace distinción entre dos categorías, clase I y clase II, se eligieron valores de tasa de transferencia y cantidad de bits por trama que van a permitir clasificar el Simulador dentro de la primera categoría.

La tasa de transferencia es la velocidad a la que se transmiten los bits, esto indirectamente establece un tiempo de duración de cada uno de los símbolos. A su vez, el estándar establece los márgenes de variación y error dentro de los cuales se considera valida la tasa de transferencia.

La cantidad de bits por trama está definida por la cantidad de sensores y de actuadores que se desean comunicar, se estableció un formato fijo de acuerdo al estándar en el que cada palabra que conforme la trama no puede superar los 16 bits. El total de la trama asciende a 512 bits, lo que

incluye palabra de sincronismo, contador de paquetes y control de errores. Según el estándar adoptado, los equipos de Clase I no pueden tener palabras de mayor tamaño.

El estándar IRIG106 categoriza la detección de errores por sistema de Control Cíclico de Redundancias (CRC) para los equipos del tipo II, pero no categoriza ni especifica ningún otro método. El simulador se implementó con un control de detección de errores del tipo "checksum", que se calcula como la suma de módulo 2 de todos los datos que componen la carga útil del mensaje.

La planta desarrollada responde a frecuencias de operación entre 100 y 500Hz, lo que aplica para poder operar con aviones no tripulados y cohetes sonda. Se adoptaron tres valores por defecto: 10, 5 y 2 ms. Que el dispositivo haya sido clasificado como Tipo I implica que no puede incorporar algún tipo relleno en la transmisión, todos los bits transmitidos son datos. La frecuencia de trasmisión y el tiempo de bit se pueden calcular mediante las ecuaciones (1) y (2):

$$F_{tx} = \frac{V_{tx}}{L} \tag{1}$$

Donde:

- Frx es la frecuenbcia de transmission en Hz
- Vtx es la velocidad de trnasmision en bits/seg
- L es la longitud de la trama en bits

Despejando tenemos que para una trama de 512 bits (64 bytes) nos da el tiempo de bit.

$$t_{bit} = \frac{512}{F_{transmisión}} \tag{2}$$

Donde el tamaño de trama es fijo en 512 bits y la velocidad de transmisión es debe ser un divisor de la frecuencia de "clock" del sistema.

3.1.1. CARACTERÍSTICAS GENERALES

El simulador cumple con las siguientes características:

- ✓ La salida PCM de datos está compuesta por dos señales, una de datos y otra de "clock".
    - Independientemente de que el estándar IRIG 106-13 no especifica esto, el formato de datos serie y "clock" sincrónico es ampliamente utilizado, lo que permite procesar la señal con cualquier otro dispositivo.
- ✓ El equipo opera a velocidades de transmisión de hasta 5Mbps.
    - Para poder categorizar el dispositivo dentro de la Clase I del estándar IRIG 106-13 en necesario que pueda soportar esa tasa de transferencia máxima.

- ✓ Los datos de salida poseen un código de línea del tipo NRZ-L.
    - El estándar define distintas codificaciones de línea para la transmisión, pero en el dispositivo se adoptó ésta ya que es la misma con la que se transmitirán los bits por el enlace telemétrico. Por lo que no es necesario implementar más codificaciones.
- ✓ El período de transmisión entre tramas es configurable entre 10.24, 5.12 y 2.048 milisegundos.
    - Estos valores quedan determinados por las condiciones impuestas por la planta y las limitaciones de la transmisión. Lo valores quedan determinados por la ecuación (1) y (2) y se adoptaron esos valores por ser los más aproximados a los propuestos inicialmente.
- ✓ La salida UART soporta velocidades de transmisión de hasta 2Mbps.
    - Para poder cumplir con las exigencias impuestas por los períodos de transmisión y el tamaño de cada una de las tramas es una necesidad poder transmitir a esa velocidad.
- ✓ El módulo fue desarrollado en una plataforma Spartan6 de Xilinx
    - Para cumplir con las condiciones y exigencias de bajo costo del sistema en su totalidad, el módulo del simulador deberá ser implementado en una FPGA de bajo costo también.

3.1.2. VENTAJAS Y LIMITACIONES

El sistema cuenta con varias ventajas, una de las más importantes es que se trata de un diseño propio ya que se dispone del código fuente con el que se realizó el mismo, esto permite expandir sus funcionalidades a medida que surjan necesidades futuras o de otros proyectos. La disponibilidad del código posibilita incorporar el simulador a otros desarrollos en los que se disponga de una FPGA.

Los recursos utilizados por el modelo de hardware del simulador son muy escasos, lo que permite incorporarlo en cualquier otro sistema sin costo adicional de recursos. El mismo ocupa únicamente 300 registros y de 600 LUTs en una FPGA de Xilinx. En el caso del chip empleado para el desarrollo, XC6SLX25, esto representa aproximadamente el 1% de su capacidad.

El sistema es compacto para su utilización, no se necesita de ningún tipo de interfaz para su operación. Toda la configuración del mismo se determina en tiempo de compilación, por lo que no se necesita de ningún software ni interfaz adicional. Los datos necesarios para configurar el

sistema son las velocidades de transmisión. Esto también impacta a nivel costos, ya que no se requiere software ni licencias adicionales para que el equipo funcione, ni para agregar módulos al equipo ya existente.

El desarrollo fue hecho en torno a una trama de datos propietaria lo que permite integrar el simulador como una herramienta para probar otros sistemas. La principal ventaja de que sea un sistema hecho a medida radica en que permite agregar las funcionalidades necesarias para poder probar la cadena de elementos intervinientes de forma integral.

Como último punto a destacar, este módulo simulador está diseñado completamente aparte del adquisidor, para que funcione independientemente de cualquier otro sistema.

3.1.3. BLOQUES FUNDAMENTALES

El sistema se implementó de forma modular para poder probar cada uno de los sub-sistemas por separado, los distintos bloques fueron implementados para poder separar las unidades que cumplen diferentes tareas y que posteriormente puedan ser reutilizables.

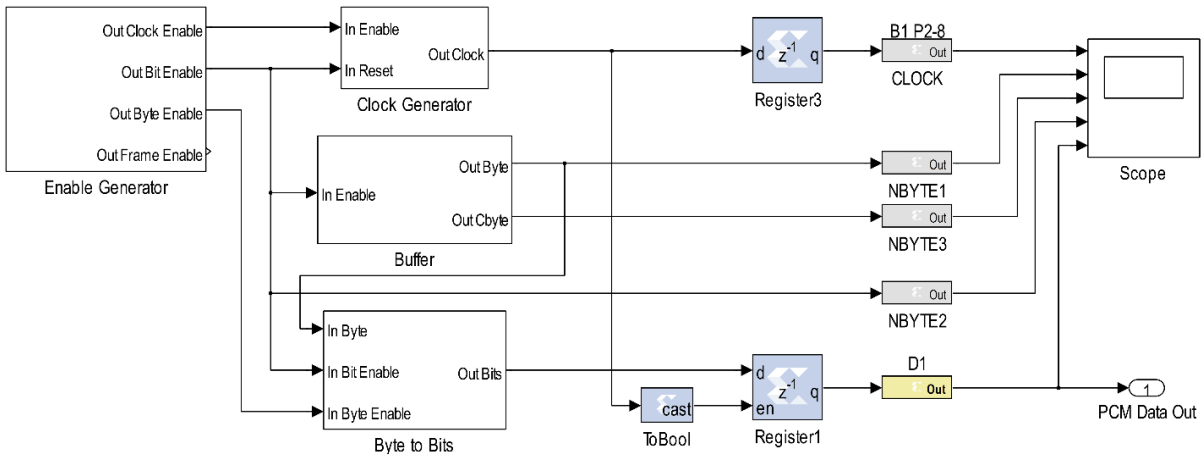


Figura 38. Diagrama en bloques del simulador PCM

Se muestra en la Figura 38 un diagrama de interconexión de los bloques principales (sun-sistemas) que componen el Simulador PCM, los cualesde son:

- ✓ Enable Generator
- ✓ Clock Generator
- ✓ Buffer
- ✓ Byte to Bits

El modelo de la Figura 35 muestra también otros bloques de registros internos y "gateways" para poder visualizar las señales directamente en un "scope" utilizando el Simulink y sin necesidad de sintetizar el hardware en la FPGA ahorrando tiempo de desarrollo.

3.1.3.1. ENABLE GENERATOR

Este bloque se encarga de generar una cadencia de tiempos precisa y de alimentar a los demás bloques con la finalidad que todo el conjunto trabaje de manera sincrónica. Las salidas de este bloque son pulsos que marcan los tiempos de todo el sistema, y el ciclo activo está determinado por el período del clock de la FPGA (en este caso de 20 ns), por ese motivo es tan preciso.

La importancia de éste bloque radica en garantizar la cadencia entre mensajes y conseguir un tiempo de espera nulo entre tramas PCM transmitidas, ya que estas deben ser continuas.

Para el desarrollo de este bloque utilizando System Generator fue necesario combinar el uso de los bloques "Down Sample" y "Clock Enable Probe" (ver Figura 39) para obtener diferentes divisiones de la frecuencia. Una vez determinada la velocidad del canal PCM, mediante la ecuación (3) se puede calcular la cantidad de pulsos N que deben dejarse en estado bajo (low) hasta volver a activarse, y así obtener una marca de tiempo interna en la FPGA en relación con la misma por medio de la ecuación (4). Para este estudio, con 100kbps, nos da 250 pulsos que debemos dejar pasar hasta volver activar una nueva marca.

$$V_{PCM} = \frac{C}{Tm} \tag{3}$$

Donde:

- V_{PCM} es la velocidad del canal PCM
- C es la cantidad de bytes por cada frame
- Tm es el tiempo de muestreo

$$N = \frac{F_{FPGA}}{2 * V_{PCM}} \tag{4}$$

Donde:

- N es la cantidad de pulsos a contar
- F_{FPGA} es la frecuencia de reloj de la FPGA
- V_{PCM} es la velocidad del canal PCM

En este caso, la señal en cuestión es denominada "Out Clock Enable", ver Figura 40.

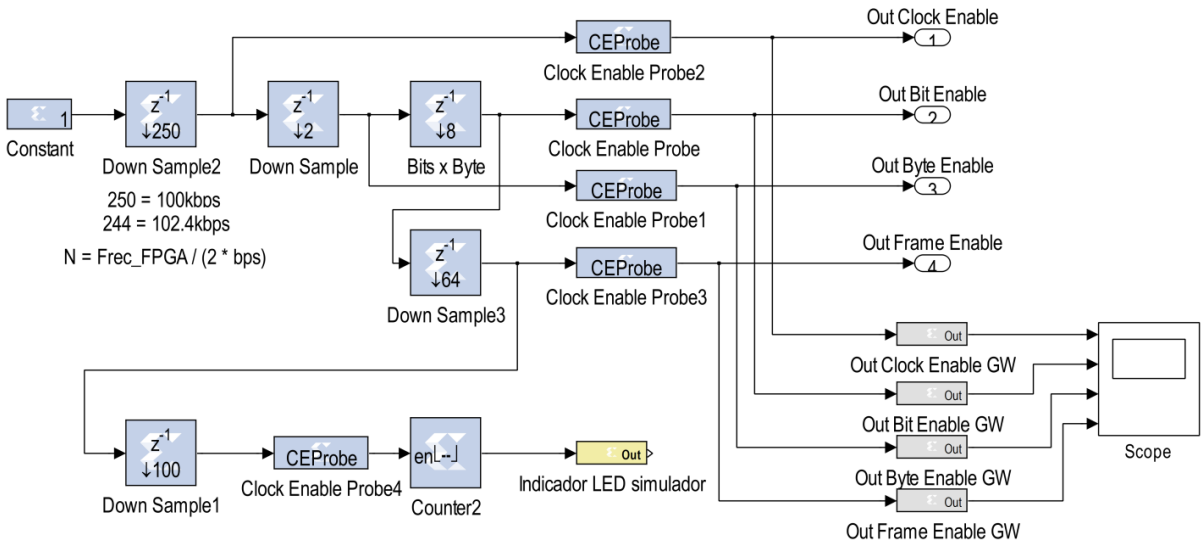


Figura 39. Desarrollo del sub-sistema Enable Generator

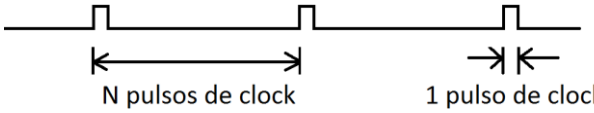


Figura 40. Salida "Out Clock Enable"

Otro ejemplo de cálculo es establecer una longitud de trama de 64 bytes con una actualización de la misma cada 5 ms:

$$Velocidad\ del\ canal\ PCM = \frac{64\ bytes}{5\ ms} = \frac{64 * 8\ bits}{0{,}005\ seg} = 102400\ bps$$

$$N = \frac{50MHz}{2 * 102400bps} \cong 244{,}14$$

Por lo tanto, se adopta un valor de 244 (el redondeo se debe a que debe ser un valor entero) en el primer bloque de “Down Sample” para generar la salida “Out Clock Enable”.

Pero, si necesitamos que el canal de datos PCM esté a una velocidad fija de 100kbps, entonces se debe calcular nuevamente el número N, y el tiempo de muestreo cambia a 5,12 ms.

La ventaja de este sistema es que es configurable y uno puede adoptar si necesita una velocidad “x” del canal de datos PCM o un tiempo de muestro “x”, y con esos datos calcular el valor de N que se necesita.

$$N = \frac{50MHz}{2 * 100000bps} = 250$$

$$Velocidad\ del\ canal\ PCM = \frac{64\ bytes}{5\ ms} = \frac{64 * 8\ bits}{0{,}00512\ seg} = 100000\ bps$$

Como se puede ver, la velocidad máxima del canal PCM está limitada por la frecuencia de trabajo del dispositivo FPGA.

Las salidas de este bloque son:

- Out Clock Enable: Con período de 250 pulsos. El ciclo activo de la señal es de 1 pulso de "clock" de la FPGA, en este caso 1/50MHz, es decir 20ns.
- Out Bit Enable: Es una señal de las mismas características que la anterior, pero el doble de período.
- Out Byte Enable: Se toma la señal anterior y se la multiplica ocho veces su período, ya que 8bits=1byte.
- Out Frame Enable: Establece una marca por cada frame total enviado.

Por último, para determinar a simple vista que el hardware esté respondiendo, se agrega un puerto denominado "LED L13" como indicador luminoso de la plaqueta. La forma de onda de las salidas se puede ver en la Figura 41.

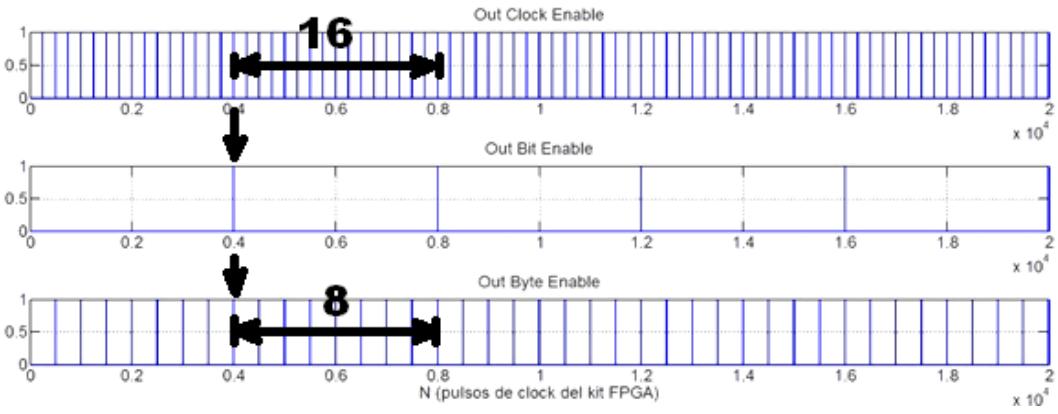


Figura 41. Salidas del sub-sistema "Enable Generator"

3.1.3.2. CLOCK GENERATOR

Este bloque se encarga de obtener una señal sincrónica con los datos PCM. Esta salida es compatible para otros sistemas adquisidores donde se necesitan Datos y "Clock". Luego veremos que el módulo adquisidor solamente se alimenta con datos y reconstruye el reloj internamente. Ver ejemplo en la Figura 42.

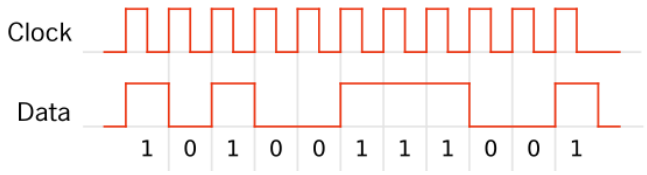


Figura 42. Datos y "Clock" sincrónicos

Esta señal de reloj tiene un ciclo activo del 50% y un período igual a la velocidad del canal PCM. Internamente fue desarrollado con un único contador descendente libre de 1 bit. El modelo del sub-sistema y las formas de onda de entradas y salida se observa en la Figura 43 y Figura 44 respectivamente.

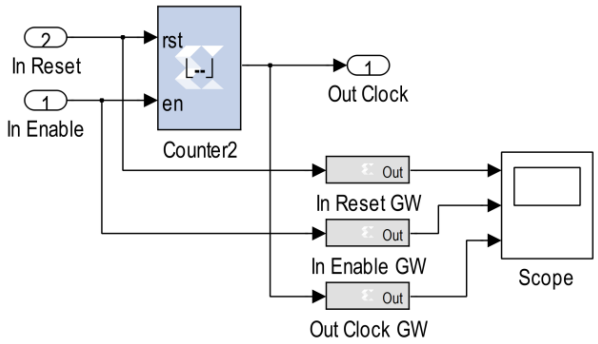


Figura 43. Modelo del sub-sistema del "Clock Generator"

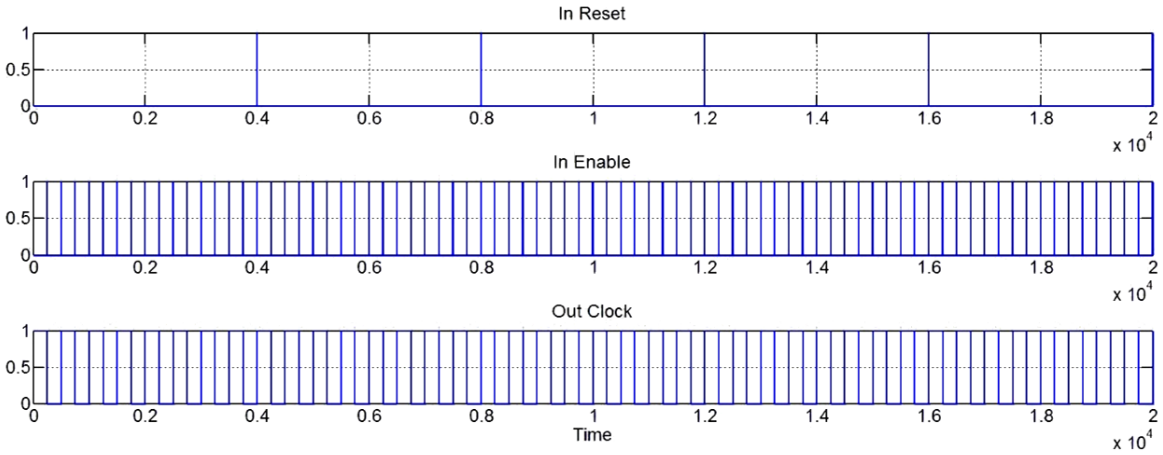


Figura 44. Entradas y salida del "Clock Generator"

3.1.3.3. BUFFER

Este bloque se encarga de almacenar datos de mediciones y generar un vector de bytes con cada uno de los campos de la trama propuesta. En este vector se deben agregar datos adicionales como palabra de sincronismo, un contador de paquetes y el código de verificación de errores.

Los datos de las mediciones que contiene el vector, es información estática que permite tener información conocida, por lo tanto, se logra una trazabilidad al momento de probar todo el sistema completo. Esta información puede cambiarse en código para poder generar distintos escenarios de simulación.

La única entrada de control que dispone el bloque es la de 'In Enable', o de validación, que se utiliza para refrescar los datos a la salida del bloque. También, con cada pulso de 'In Enable' se incrementa un contador interno para identificar cada trama.

Este bloque genera un vector de bytes con información ficticia a transmitirse. Está compuesto por un contador de bytes y un código en M-Code que implementa una máquina de estados, ver "Anexos - Código Frame Generator" para información sobre el código implementado.

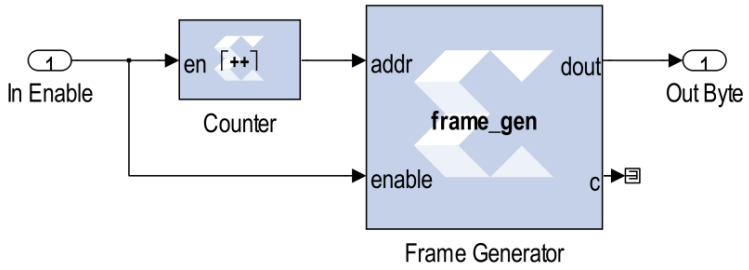


Figura 45. Modelo del sub-sistema "Buffer"

3.1.3.4. BYTE TO BITS

El bloque se encarga de convertir la entrada en formato PCM, es decir, que serializa los bytes en su entrada convirtiéndolos a bits uno a continuación de otro. Este módulo que transforma generados por el "Buffer" y se alimenta del "Enable Generator" quien le dice que duración debe tener cada bit a su salida.

Para lograr esto se implementa un contador ascendente de 3 bits, que cuenta de 0 a 7, quien le da la orden al multiplexor para sacar el dato de un byte, bit a bit. El modelo para Simulink se lo puede ver en la Figura 46.

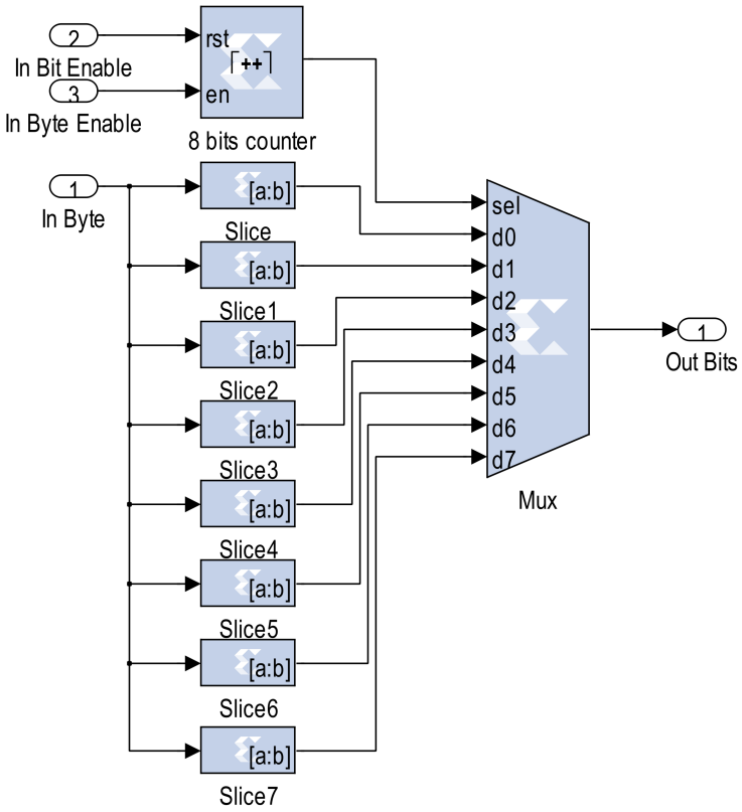


Figura 46. Modelo del sub-sistema "Byte to Bits"

3.2. ADQUISIDOR PCM

La reconstrucción de los datos telemétricos está dividida en varias etapas, hay que tener en cuenta que los datos recibidos están limitados en ancho de banda y tienen ruido sumado. Esto representa un desafío ya que para poder discernir entre un '0' lógico o '1' lógico debe tenerse un criterio adecuado para no interpretar erróneamente estos valores y alterar la cadena de procesamiento. Es por eso que se opta por realizar, antes de la regeneración de la señal, un filtrado digital, es decir que se debe realizar algún tipo de procesamiento que mejore la señal recibida.

Para poder lograr la regeneración requerida, es necesario observar que la información del régimen permanente de la señal. Se sabe que el nivel de la señal, '0' o '1' lógico, en el punto medio de cada bit se mantiene estable pese a la presencia ruido. Se observa en la Figura 47 la diferencia entre la señal original transmitida y la recibida está en que la segunda está acotada en banda, por eso los flancos abruptos se perdieron.

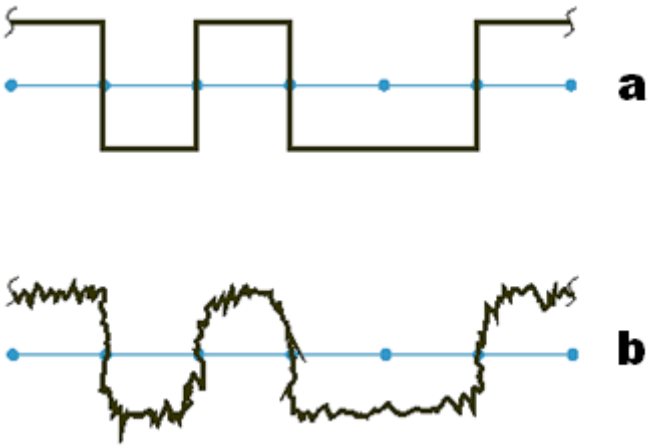


Figura 47. Señal original transmitida (a) y Señal recibida reducida en ancho de banda con ruido adicional (b)

El principio de funcionamiento de la re-sincronización se encuentra en un registro de desplazamiento de mayor tamaño que el de la palabra a encontrar y con un "clock" generado de forma local también superior al de la señal a detectar. Como se puede ver en la Figura 48, a modo de ejemplo, se trabaja con una palabra de sincronismo de 4 bits y con un "clock" 4 veces superior, esas condiciones determinan un registro de desplazamiento de 16 bits. No es necesario que exista una relación entre dichos números, en el caso real de aplicación se utilizó una palabra de 16 bits y un "clock" 10 veces mayor.

La operación del registro de desplazamiento a una mayor frecuencia permite hacer un sobre muestreo de la señal de datos, de forma que cada bit de la señal quede replicado N veces, según la ecuación (5):

$$N = \frac{F_{sobre\ muestreo}}{F_{reloj\ original}} \tag{5}$$

El registro cuenta con una salida paralela entrelazada, en la que cada bit está separado del siguiente por N registros. De esta salida es que se espera encontrar la primera ocurrencia de la palabra de sincronismo, se activa la salida de datos serie y un nuevo "clock" es regenerado.

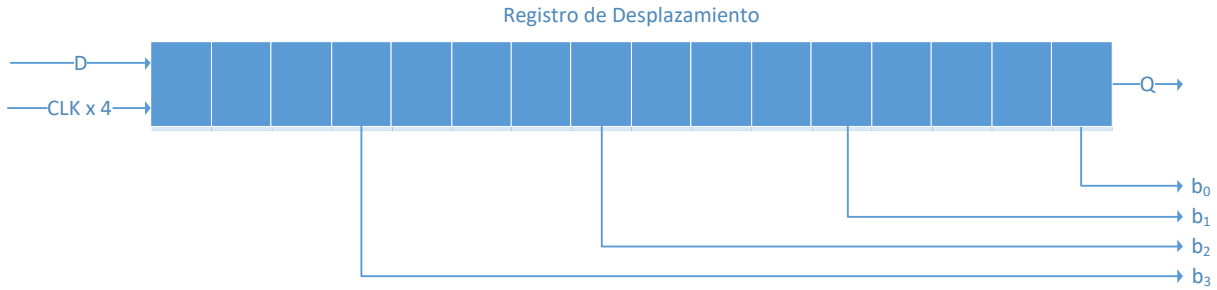


Figura 48. Principio de funcionamiento del sincronizador

Una vez que sucede esta primera ocurrencia se deberá esperar N/2 pulsos de “clock” para garantizar que cada bit sobre muestreado se posicione en la mitad del bit real. Después de ese momento la salida serie ya se logra sincronizar, y el “clock” de salida se debe regenerar con un divisor N veces del “clock” interno. La ventaja de trabajar con un multiplicador cada vez mayor está en que permite encontrar con mayor precisión el punto central de cada bit, minimizando las probabilidades de error, esto se puede ver en la Figura 48.

No existe sincronismo entre la señal de entrada y la frecuencia de muestreo (“clock” multiplicado por N) por lo es importante que se tome la muestra del medio como la más representativa de todo el bit ya que es la que asegura que los períodos transitorios de la señal ya terminaron. El sobre muestreo disminuye el error de encontrar dicho punto medio, pero un sobre muestreo excesivo incrementa el uso de recursos de la FPGA innecesariamente. Como se ve en la siguiente figura, al incrementar el nivel de sobre muestreo se consigue que la muestra N/2 esté cada vez más centrada con respecto a los flancos detectados:

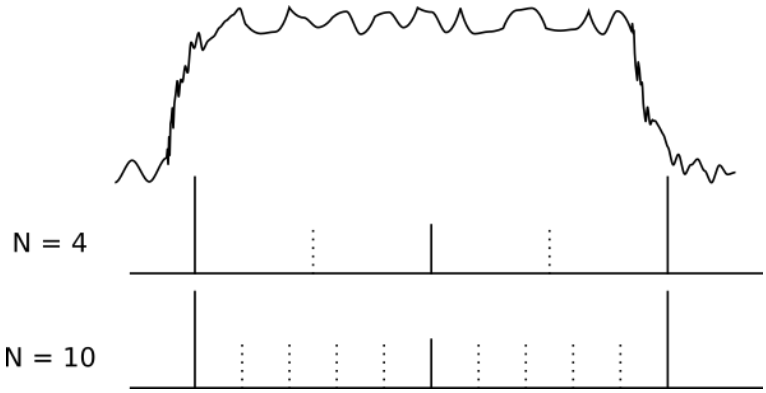


Figura 49. Señal ruidosa sobre muestreada a 4 y 10 veces la tasa de bit

La probabilidad de que la muestra central coincida con el punto central del bit está dada por la ecuación (6).

$$P = \frac{1}{N} \tag{6}$$

Para N = 4, el punto central tiene un 25% de probabilidades de coincidir con la muestra, mientras que para un N = 10 las probabilidades de coincidir son de 10%. Incrementar por encima de éste valor pierde practicidad ya que implica trabajar con frecuencias cada vez más altas y dificulta su obtención de la misma en las plaquetas de desarrollo que se utilizan en este estudio.

3.2.1. VENTAJAS Y LIMITACIONES

Una de las principales ventajas del método aplicado para la resincronización de la señal es que se puede acomodar a cualquier tamaño o palabra de sincronismo. Esto le aporta flexibilidad al sistema desarrollado y permite que sea utilizado en otro contexto.

Una limitación que se desprende de su forma de operación es que para poder reconfigurar el sistema para que trabaje con otra tasa de transferencia u otra palabra el mismo debe ser recompilado. La flexibilidad que aporta no es al momento operativo del mismo, generalmente se adopta una tasa de datos y durante el desarrollo de una aplicación determinada no se modifica.

El acercamiento modular del desarrollo permite que se puedan incorporar más bloques a la cadena, como es el caso de un filtro a la entrada con el fin de mejorar la relación señal a ruido de la señal telemétrica.

3.2.2. BLOQUES FUNDAMENTALES

El desarrollo del adquisidor se separó en cuatro bloques fundamentales cada uno con una función específica dentro de la cadena de procesamiento. De ésta forma, cada módulo se pudo evaluar por separado, también agrega la posibilidad de actualizar cada parte para futuros desarrollos.

El desafío del desarrollo se encuentra en poder detectar el inicio de cada paquete para regenerar los datos de forma sincrónica para su posterior procesamiento. El modelo completo se puede ver en la Figura 50.

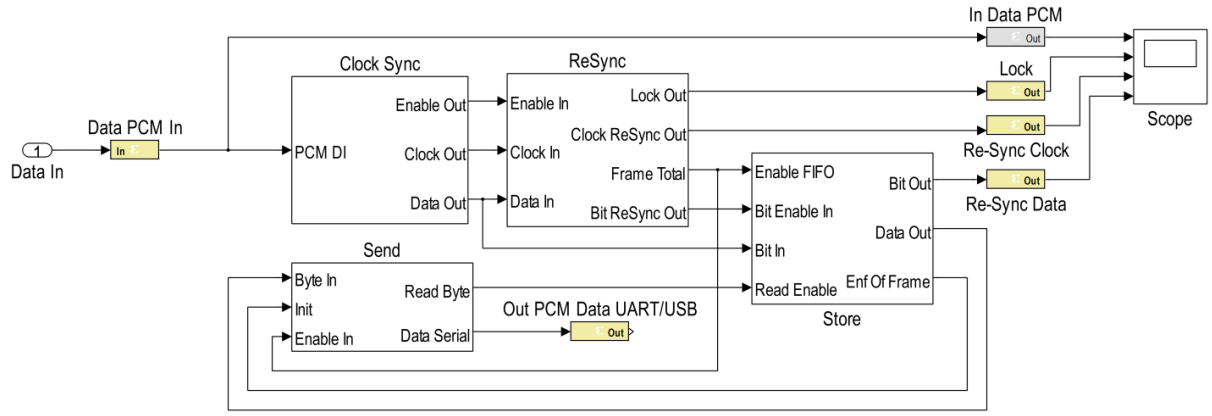


Figura 50. Diagrama en bloques del adquisidor

Los bloques principales que componen el Adquisidor son:

- ✓ Clock Sync
- ✓ ReSync
- ✓ Store
- ✓ Send

A continuación, se detallan los requisitos de cada uno de los bloques.

3.2.2.1. CLOCK SYNC

Es una primera instancia de sincronización. Para ello se utiliza una técnica de detección de flancos ascendentes de la señal que ingresa al sistema y un contador asociado. Este conjunto de sub-bloques logra una primera reconstrucción del "clock".

El algoritmo desarrollado para el bloque de resincronización toma como base la detección de flancos ascendentes de la señal entrante de datos PCM. Este sub-sistema (Figura 51) realiza una generación de "clock" PCM continuamente y se sincroniza automáticamente cuando se detecta un flanco de subida en la señal entrante. Debe configurarse con la velocidad del canal PCM en el contador donde se indica la cuenta.

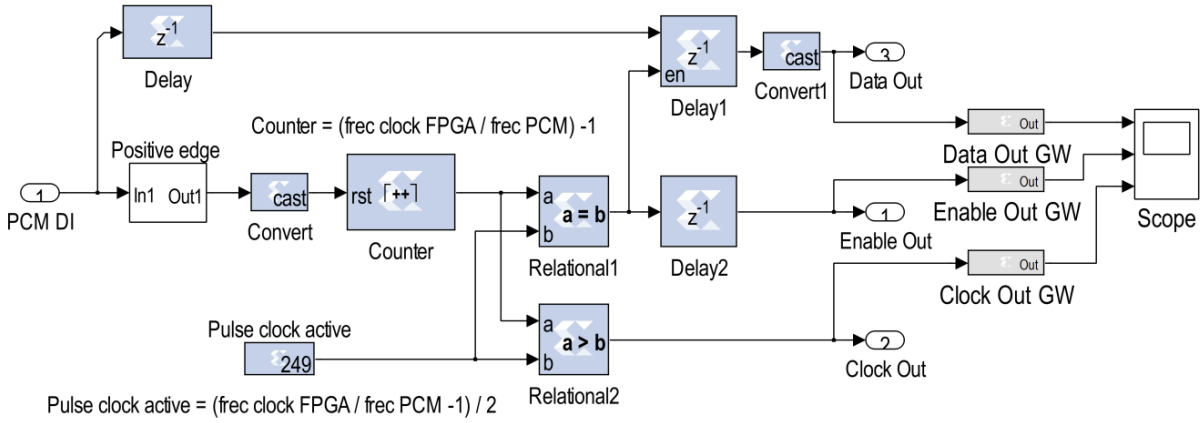


Figura 51. Modelo del sub-sistema "Clock Sync"

Los únicos valores que deben configurarse son el límite de la cuenta para el bloque "Counter" para que el "clock" se auto-resetee si no se encuentra un flanco en la señal entrante (ecuación (7)), y el valor de la constante "Pulse clock active" para establecer el ciclo activo del "clock" en un 50% (ecuación (8)). Para este caso de 100kbps:

$$Límite\ contador = \frac{Frecuencia\ de\ clock\ de\ FPGA}{velocidad\ del\ canal\ PCM} - 1 = \frac{50\ MHz}{100kbps} - 1 = 499 \qquad (7)$$

$$Pulse\ clock\ active = \frac{\frac{Frecuencia\ de\ clock\ de\ FPGA}{velocidad\ del\ canal\ PCM} - 1}{2} = \frac{\frac{50\ MHz}{100kbps} - 1}{2} \cong 249 \qquad (8)$$

Como resultado de la simulación en Simulink, podemos ver en la Figura 52 como se corrige al detectar un flanco positivo de la señal PCM entrante.

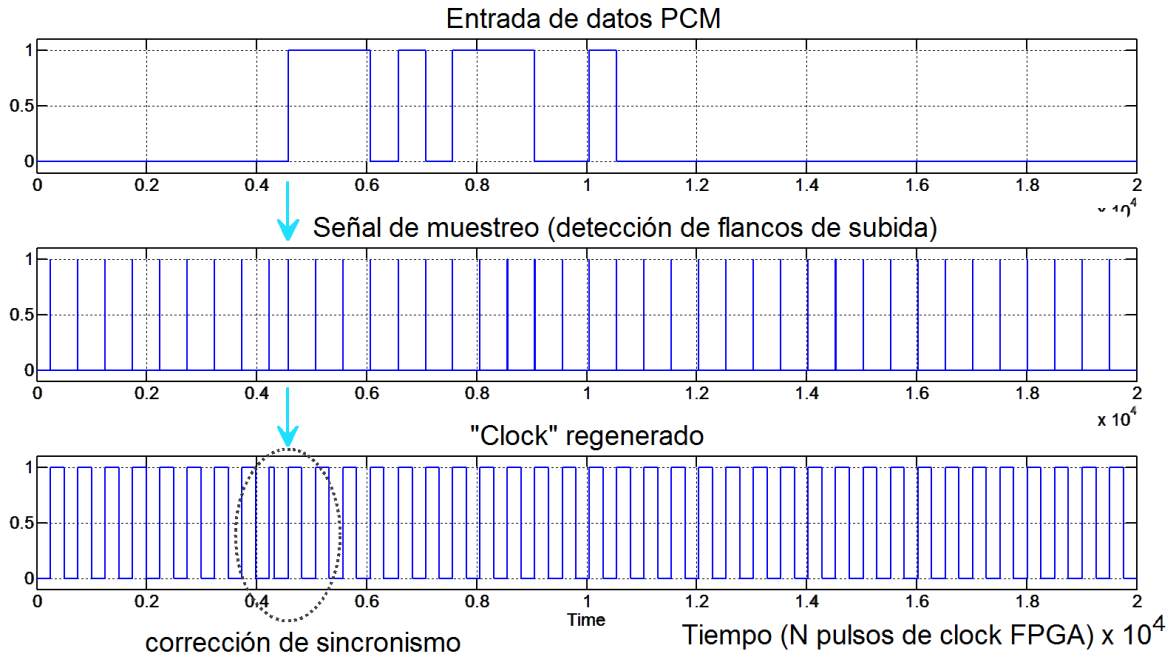


Figura 52. Simulación del sub-sistema "Clock Sync"

3.2.2.2. RESYNC

Este bloque genera una segunda instancia de sincronización, recordando que las aplicaciones de este adquisidor son generalmente misiones críticas. La importancia de este bloque radica en la detección de la palabra de sincronismo utilizando dos bloques de máquinas de estado trabajando en tándem.

El sub-sistema ReSync, ver Figura 53, realiza una búsqueda de la palabra de sincronismo para engancharse con la señal entrante y re-sincroniza el “clock” nuevamente. Para lograr esto se utilizan dos bloques en tándem. El primero posee una máquina de estado finita y un registro de desplazamiento, para encontrar bit a bit, la palabra de sincronismo. El segundo, genera señales adicionales que alimentan los demás bloques.

En la Figura 54 se pueden ver los resultados de la simulación. En el último canal podemos ver la señal de datos PCM entrante al sub-sistema, y al ocurrir el primer flanco de subida, empieza a correr el reloj, y el indicador de trama válida se activa indicando que la posible aparición de la misma. Desde ese momento, el bloque “Lock”, posee un algoritmo de búsqueda de un patrón conocido (la palabra de sincronismo) y se puede ver en las mediciones que luego de 16 bits, se activa su salida indicando que encontró una palabra de sincronismo válida.

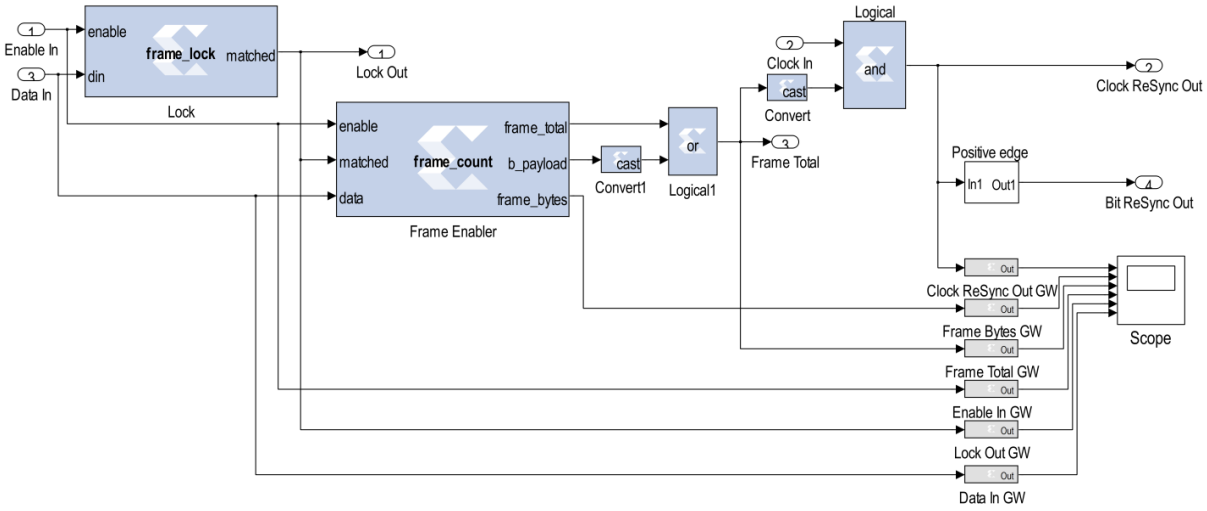


Figura 53. Modelo del sub-sistema "ReSync"

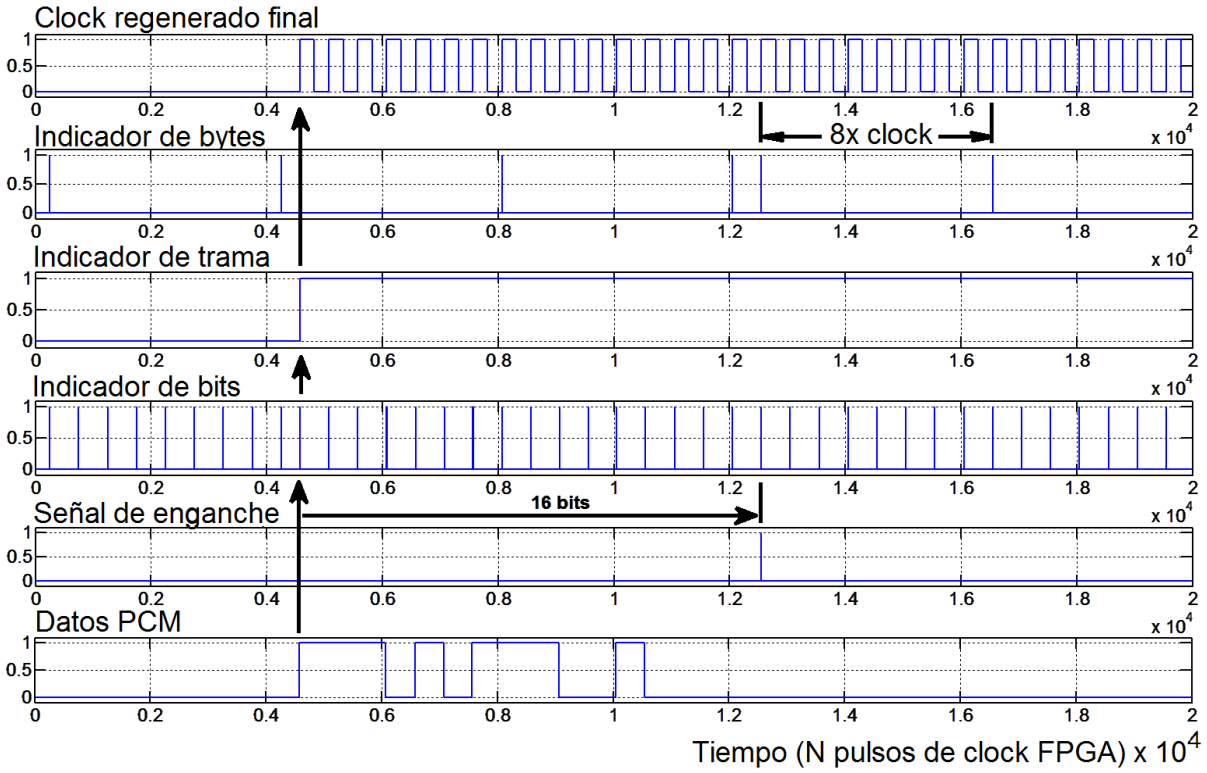


Figura 54. Resultados del sub-sistema "ReSync"

3.2.2.3. STORE

Posee dos funciones principales, la primera es de convertir los bits sincrónicos en los bytes que componen la trama. La segunda, es de almacenar los bytes mencionado en una memoria del tipo FIFO para generar un vector con todos los bytes que componente la trama completa.

El almacenamiento de bytes en el sistema, es esencial para poder realizar una conversión de protocolo. Para el caso que se trata, necesitamos convertir los datos PCM que ingresan al sistema

y realizar su adquisición para posteriormente transformarlos en un protocolo UART vía una conexión física USB. Este sub-sistema en combina un bloque de M-Code para tomar una serie de bits y entrantes, y a partir de la marca de enganche (Lock), generar los bytes de los datos contenidos en la trama como se puede ver en el modelo de la Figura 55. Estos datos en bytes son almacenados en una memoria FIFO configurada como una memoria distribuida (ver Figura 56) para poder disminuir los recursos y área de la FPGA, especialmente para dispositivos de bajo costo donde son más limitados los recursos de hardware.

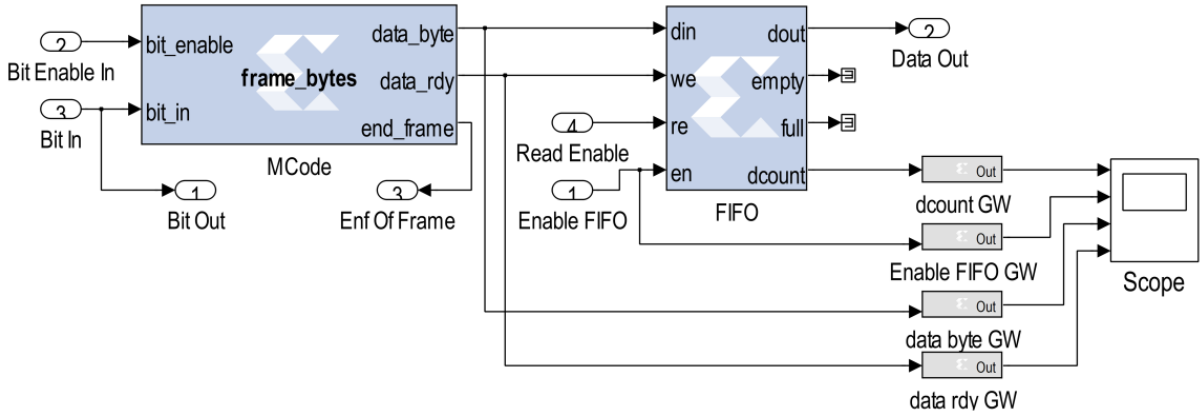


Figura 55. Modelo del sub-sistema "Store"

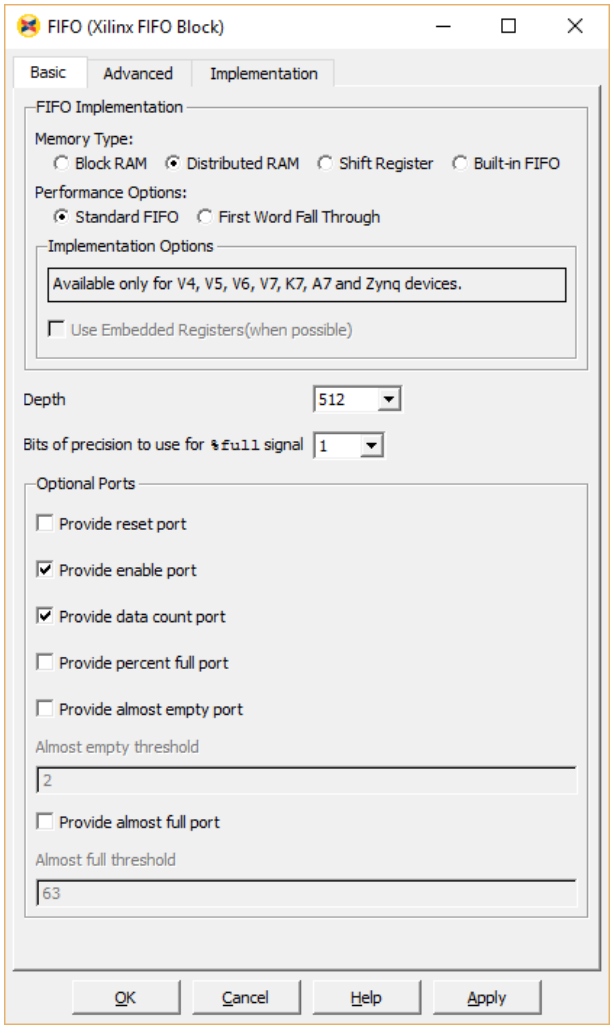


Figura 56. Cuadro de configuración del bloque FIFO

Los resultados de la simulación se muestran en la Figura 57 y se puede visualizar como el sub-sistema toma el byte válido al inicio de la trama, recordando que la palabra de sincronismo es EB90 en hexadecimal, es decir 235 (EB) y 144 (90).

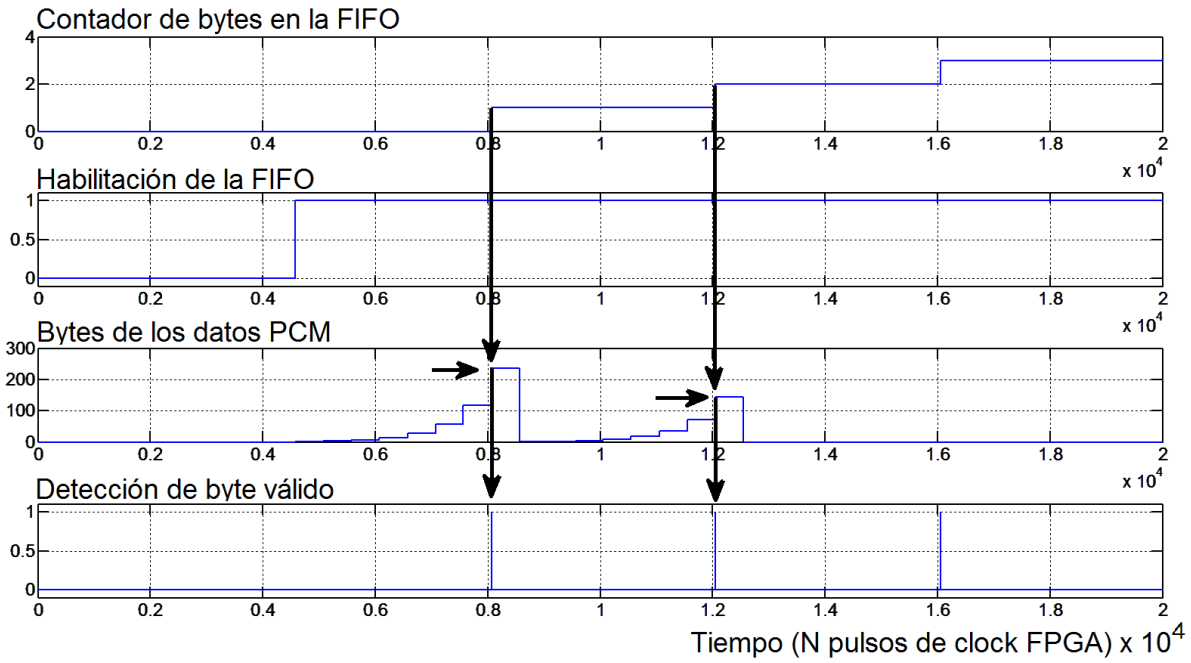


Figura 57. Resultados de la simulación del sub-sistema "Store"

3.2.2.4. SEND

La función de este bloque es retransmitir los datos recibidos a las computadoras que componen el puesto de tierra por medio una interfaz serie USB.

El sub-sistema “Send” se encarga de recolectar los datos almacenados en el bloque “Store” y serializarlos con un protocolo UART 8N1 (es decir, 1 bit de comienzo de palabra, 8 bits de datos y un bit de parada). En la Figura 58 se aprecia el modelo a implementar del sub-sistema en cuestión y en la Figura 59 se muestra un resultado de simulación de un sector de la trama PCM convertida.

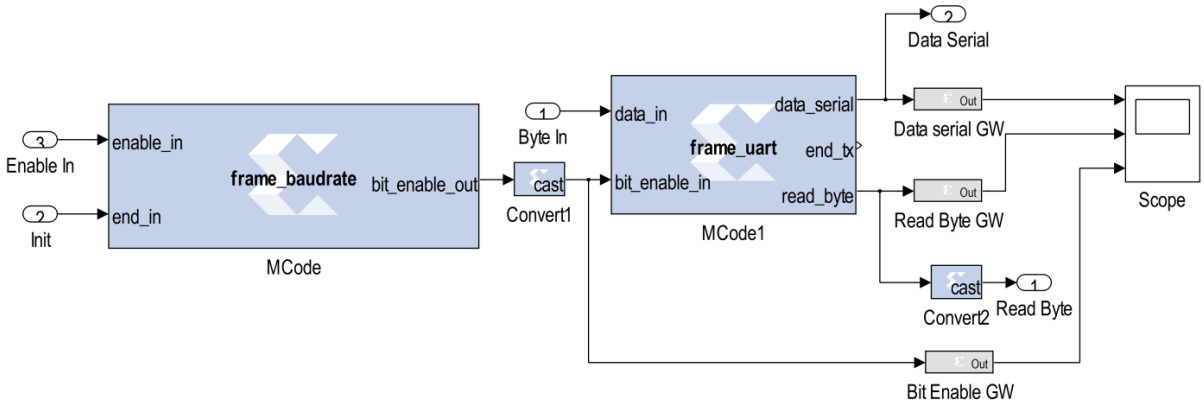


Figura 58. Modelo del sub-sistema "Send"

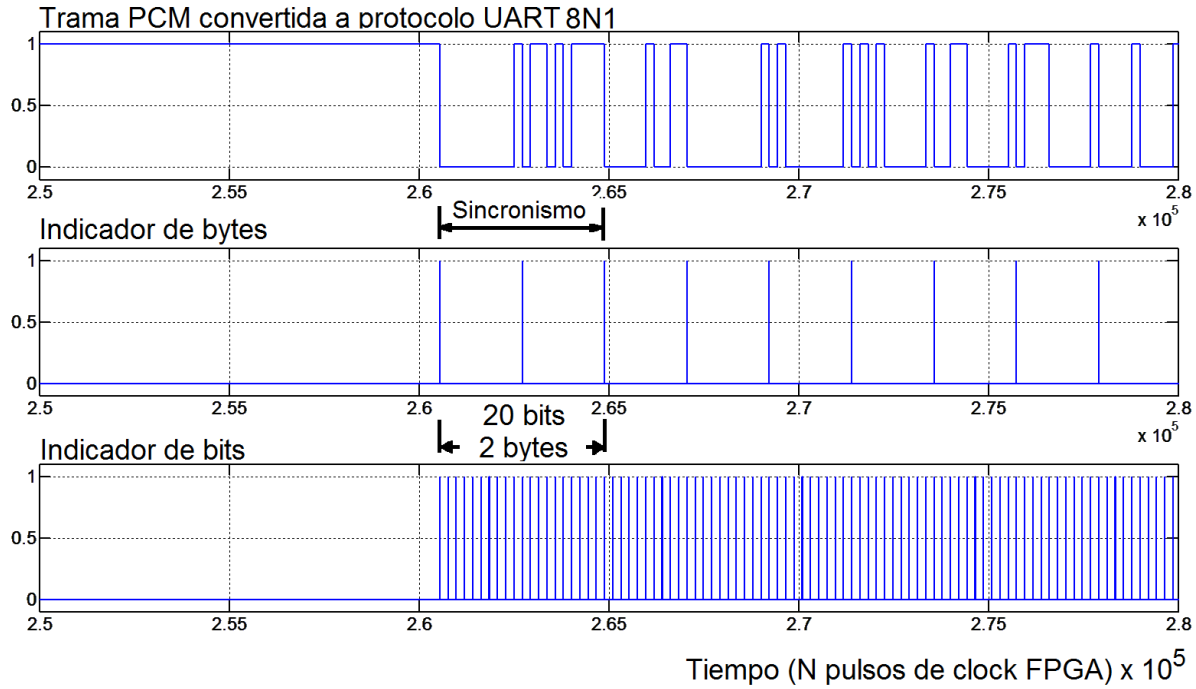


Figura 59. Resultados de la simulación del sub-sistema "Send"

A continuación, se muestra una tabla comparativa con los valores esperados para cada una de las frecuencias de operación, que dependen de las ecuaciones vistas anteriormente relacionadas con los parámetros de la transmisión PCM y el tiempo de bit. El sub-sistema “Send” es totalmente reconfigurable.

Velocidad de Transmisión	Tiempo entre Tramas	Tiempo de bit	Frecuencia de “clock”
250 Kbps	2048 µs	4 µs	250 KHz
100 Kbps	5.12 ms	10 µs	100 KHz
50 Kbps	10.24 ms	20 µs	50 KHz

Tabla 2. Parámetros en función de la velocidad de transmisión

La transmisión serie se realiza a 3 velocidades diferentes, es importante corroborar que estén dentro de los márgenes aceptables que establece la norma EIA/TIA-232-F. La misma especifica

un desvío máximo del 4% en el periodo de bit a transmitir (Texas Instruments, 2002). Para las velocidades empleadas, se muestra a continuación una tabla con los valores máximos y mínimos aceptables:

Tasa Real	Valor Ideal	Valor Mínimo	Valor Máximo
460800	2.17 µs	2.26 µs	2.08 µs
230400	4.34 µs	4.52 µs	4.17 µs
115200	8.68 µs	9.04 µs	8.34 µs

Tabla 3. Tiempos mínimos y máximos de bit para transmisión serie

Medición del Transmisor serie a 460800 bps:

En este caso, utilizamos una tasa de velocidad serie teórica de 460800bps (es la única tasa que permite trasmitir paquetes de 64bytes en 2,048ms) y utilizamos un osciloscopio (los modelos utilizado para todas las mediciones es el TBS1202 de Tektronix (Tektronix, s.f.) y el de Agilent (Keysight, 2012)) para medir el ancho de un bit de datos de la salida serie. Con esta medición se determinó que el ancho de bit de datos es de 2,24µs, lo cual da una tasa de 446428bps. Este valor real, introduce un error de 3,11%, el cual se admite por la norma. Ver Figura 60.

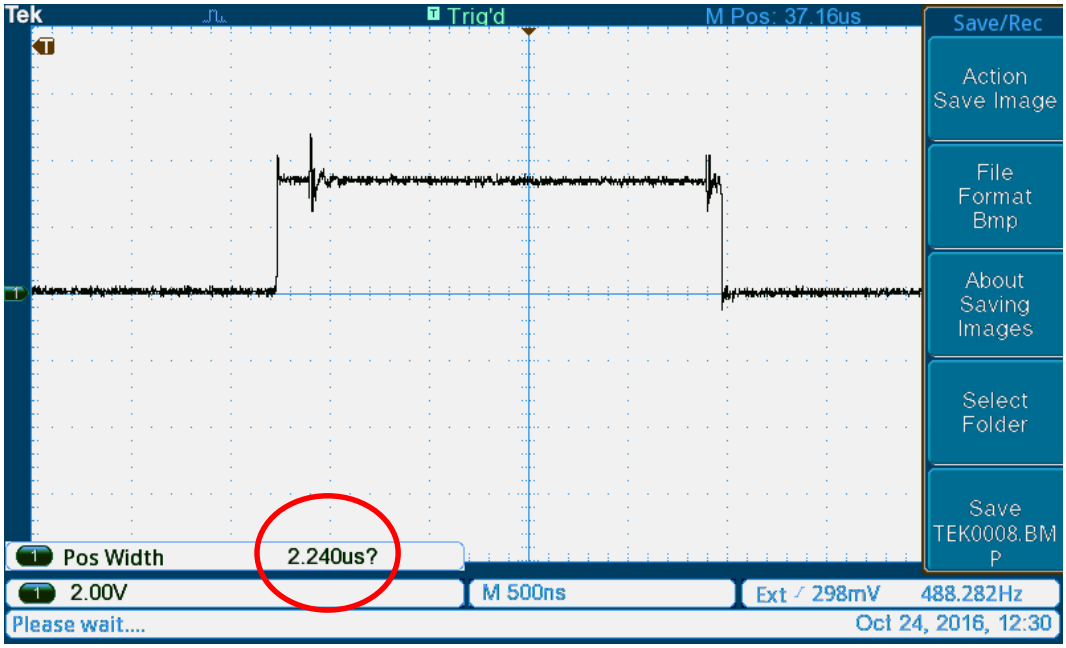


Figura 60. Medición del tiempo de bit del Transmisor serie a 460800 bps

Medición del Transmisor serie a 230400 bps:

Con una configuración de 230400 baudios (ideales) en el puerto se obtiene una tasa de 223164 baudios (reales). El error en la tasa de transferencia es de 3,14%. Ver Figura 61.

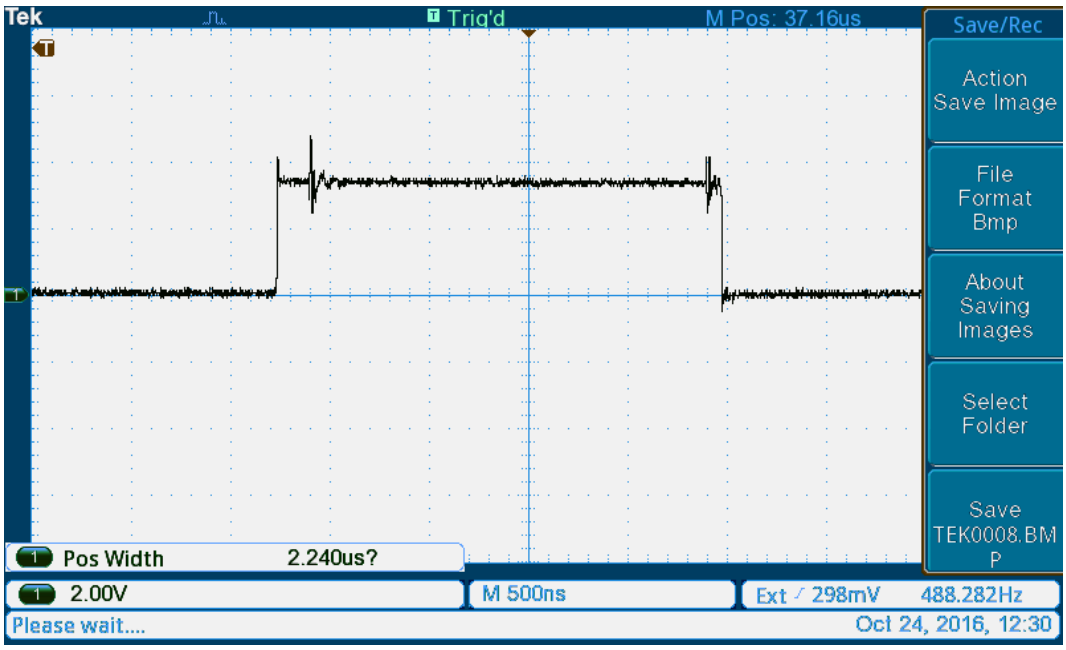


Figura 61. Medición del tiempo de bit del Transmisor serie a 230400 bps

Medición del Transmisor serie a 115200:

Para una tasa de transferencia de 115200, el valor obtenido real de tasa de transferencia es 111607 baudios. El error entre la tasa real y la ideal es de 3,11%. Ver Figura 62.

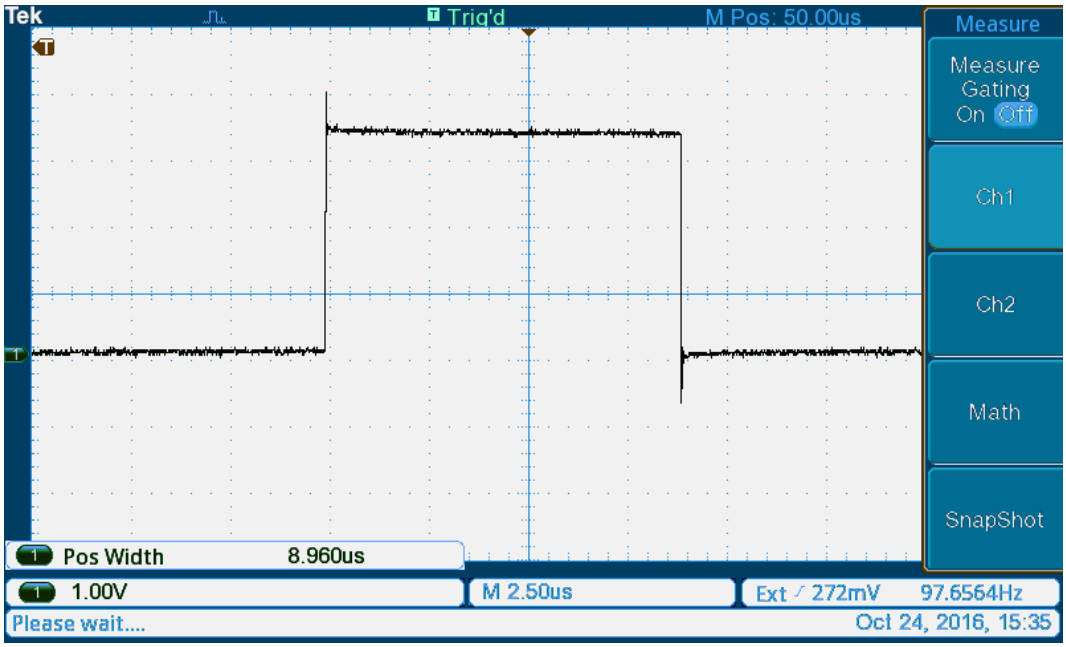


Figura 62. Medición del tiempo de bit del Transmisor serie a 115200 bps

3.3. ENSAYOS EN HARDWARE

En la Figura 63 se puede visualizar las salidas resultantes de una simulación (breve) en el simulador de datos PCM. En la misma podemos visualizar “Data Byte” en el primer canal, que corresponden a los datos (bytes) del vector generado por el bloque “Buffer”. En segundo y tercer lugar se encuentran las señales de marca de bytes y bits, que son generadas por el bloque “Enable Generator” y que alimentan a los demás bloques. En los últimos dos canales se muestran las señales finales de datos y “clock” PCM.

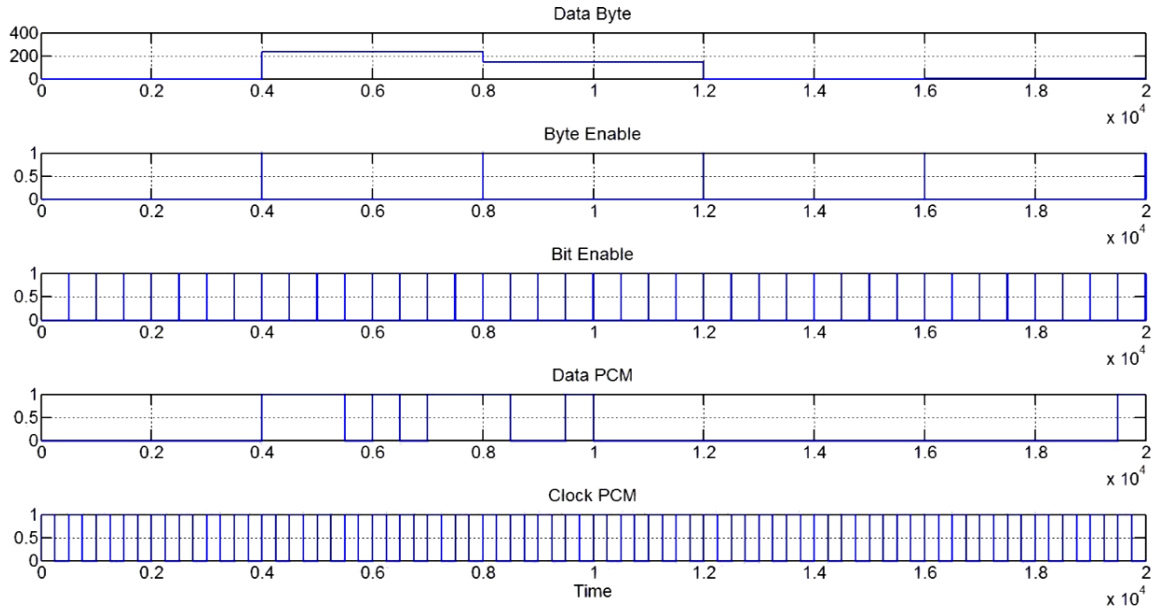


Figura 63. Señales resultantes de la simulación del sistema simulador PCM

Para la implementación del simulador PCM, se utilizó el siguiente software base y el dispositivo FPGA de la Tabla 4. Los resultados de implementación final, es decir, una vez generado el ruteo definitivo de hardware luego de la síntesis, se pueden analizar en la Figura 64 y Figura 65 que el área que ocupa el diseño dentro de la FPGA es muy pequeña.

Tabla 4. Versión de software para síntesis y dispositivo FPGA

Software Version and Target Device			
Product Version:	ISE:14.2 (WebPack) - P.28xd	Target Family:	Spartan6
OS Platform:	NT64	Target Device:	xc6slx25
Project ID (random number)	77e28071ac21441e93c1698e6151e328.80e021dc65bf46ee9c92c184c3cfb5e3.1	Target Package:	ftg256
Registration ID	__0_0_0	Target Speed:	-3
Date Generated	2017-09-13T14:54:14	Tool Flow	CommandLine

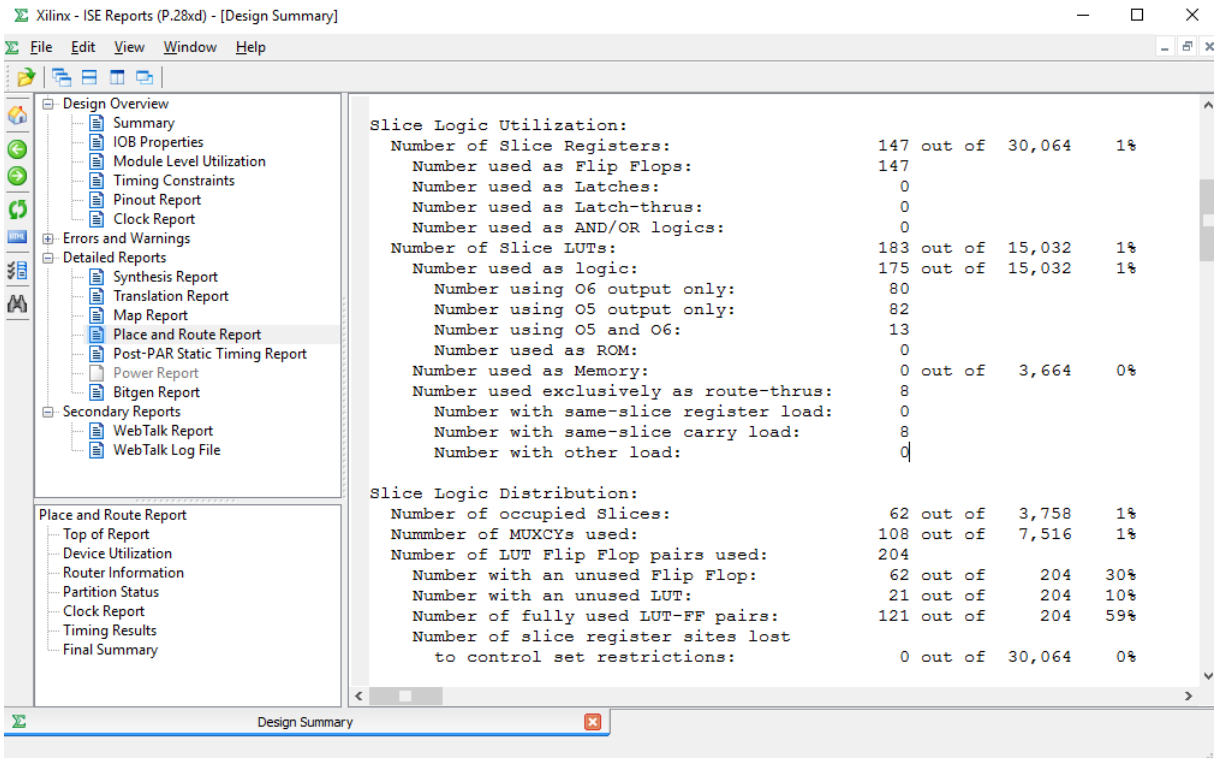


Figura 64. Resultados del "Place and Route" del simulador PCM

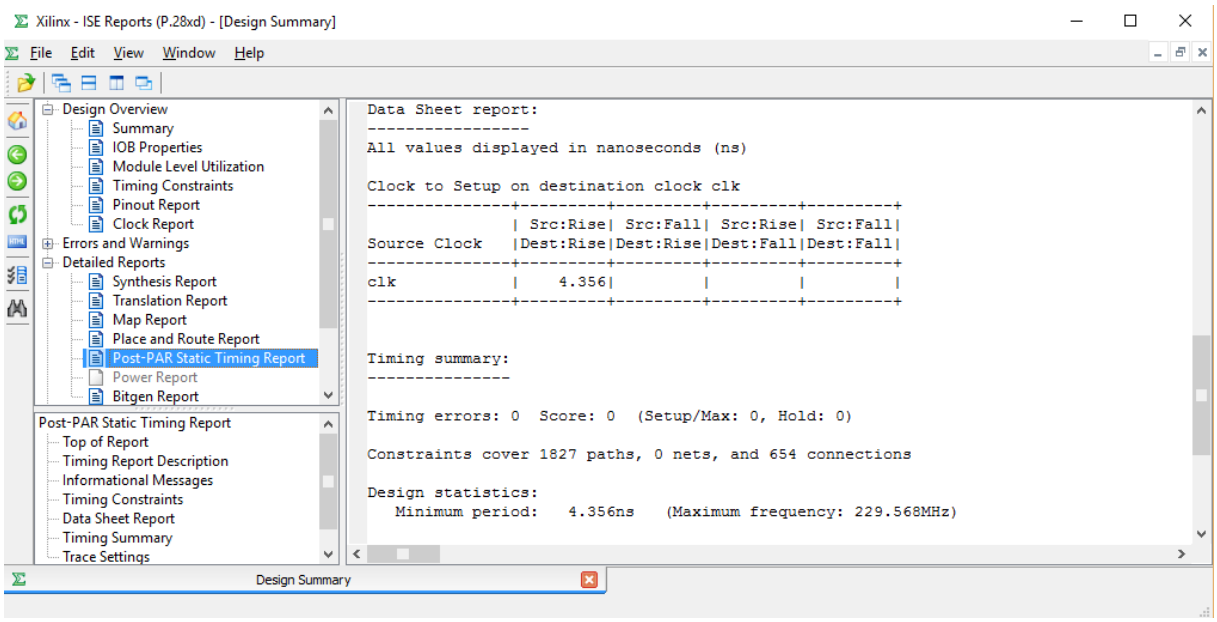


Figura 65. Resultados de tiempos del "Place and Route" del simulador PCM

Una vez que se graba el diseño final en el hardware, se analizan los resultados con mediciones realizadas a través de un osciloscopio (Figura 66), y se muestra una señal de “clock” estable de 100KHz, con un desvío de 3,6pHz, es decir, una señal realmente estable en frecuencia. También se muestra su línea de datos PCM y su valor decodificado en heaxadecimal en la parte inferior de la misma figura, ya que este osciloscopio posee un adquisidor embebido de señales serie, y se corrobora (ver datos del vector en “Anexos - Código Frame Generator”) con el modelo que son idénticos.

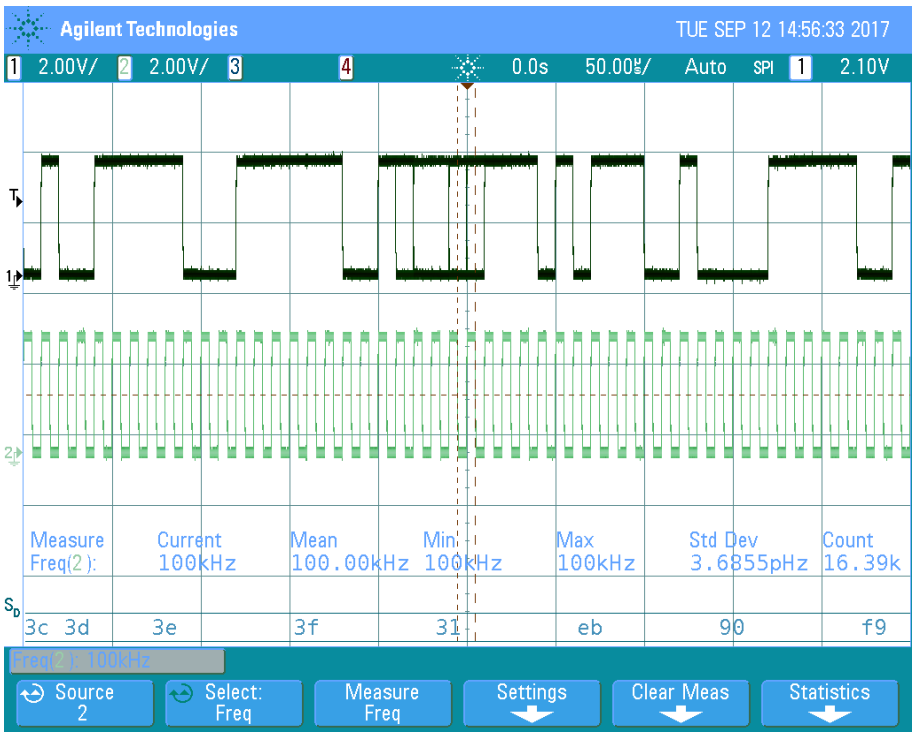


Figura 66. Medición con osciloscopio del simulador PCM

Por otro lado, como hemos mencionado anteriormente, este diseño y desarrollo está concebido para aplicaciones de misión crítica y una de las cuestiones fundamentales para el análisis de su funcionamiento radican en la estabilidad del "clock". La medición efectuada con un frecuencímetro (el modelo utilizado para todas las mediciones es el FCA3000 de la firma Tektronix (Tektronix, s.f.)) arroja los resultados de la Figura 67 y se visualiza que la frecuencia media medida corresponde a 99,9998 KHz con una desviación estándar de 1,1248 mHz con 125 muestras por segundo en un total de 10 segundos de duración, por lo tanto el error es mínimo y se desprecia.

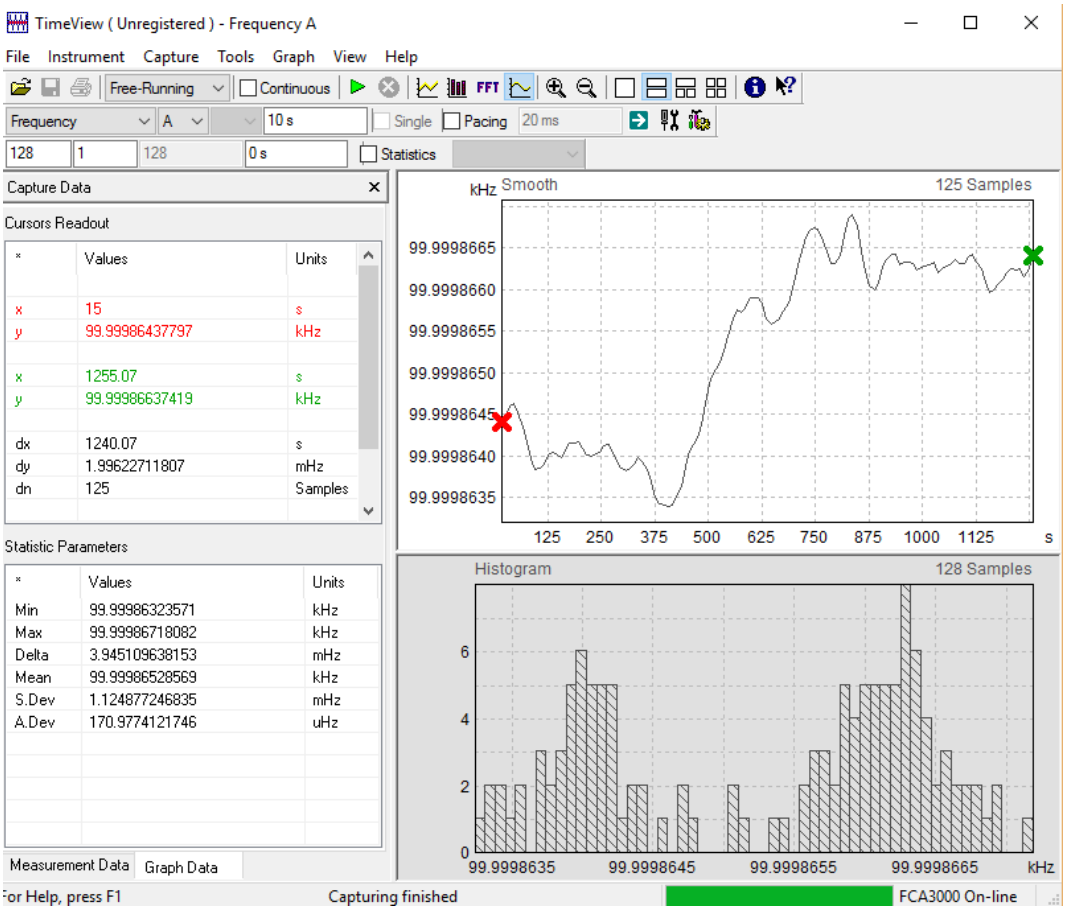


Figura 67. Medición de la estabilidad del "Clock PCM"

Por último, se utiliza un analizador lógico que nos permite visualizar y medir las caracteristicas de las señales de datos y “clock” PCM. En la Figura 68 se muestra el resultado de la medición de frecuecia del clock y los datos en binario de la señal PCM, y al decodificarse los bits, se puede ver la palabra de sincronismo (EB90 en hexadecimal) en los dos primeros bytes, por lo tanto puede concluirse que los datos generados por el simulador son correctos.

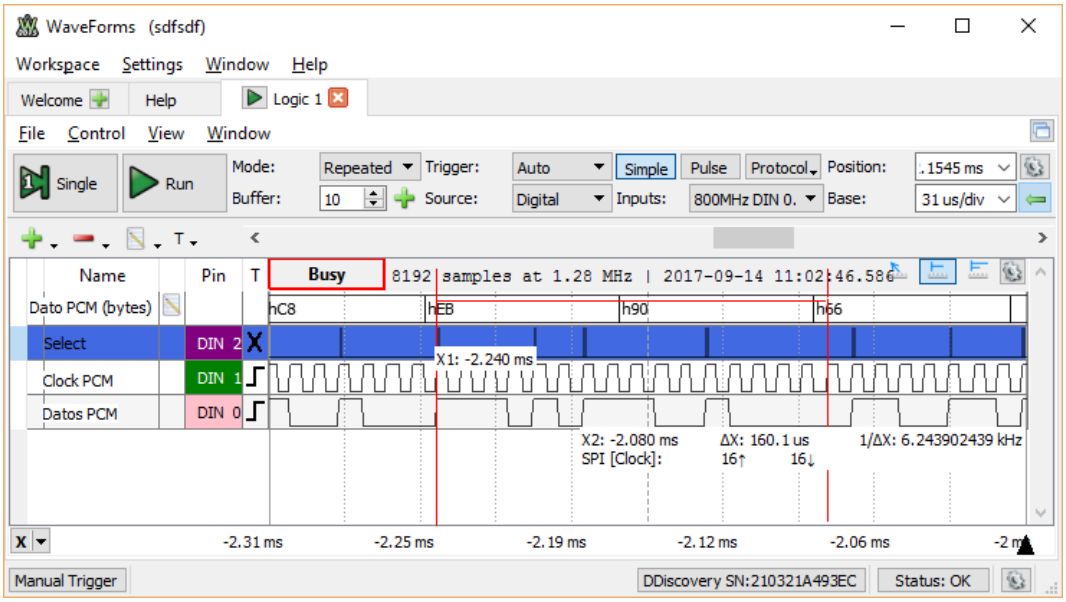


Figura 68. Medición lógica de los canales de Datos y Clock PCM

3.4. IMPLEMENTACIÓN DEL FILTRO DIGITAL PARA PCM

Una señal PCM real trae incorporado un ruido adicionado. Por este motivo se implementa un filtro digital como un bloque por separado para el pre-procesamiento de la señal antes de ingresar al sistema adquisidor.

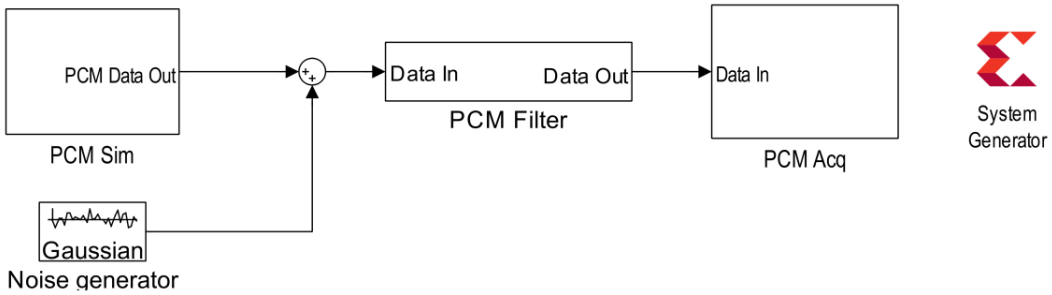


Figura 69. Implementación del modelo completo

Con el fin de asegurar una condición extrema de ruido externo que influya a la señal PCM, la condición elegida para el diseño es que el filtrado sea para condiciones críticas donde el ruido podría superar el 50% del nivel de señal. Para lograr esto realizamos la configuración del generador de ruido gausiano ("Noise generator") como establece la Figura 70 y realizamos la medición de ruido mediante la ecuación (9).

$$SNR = \frac{Signal}{Noise} = \left| \frac{Valor\ RMS\ de\ las\ muestras\ del\ dato\ PCM}{Valor\ RMS\ de\ las\ muestras\ del\ ruido\ generado} \right| \qquad (9)$$

El modelo de medición de ruido se muestra en la Figura 71 y los resultados arrojan un valor de 0,6 aproximadamente, es decir que el ruido en este caso está configurado para ser de -4dB (un 60%

del nivel de señal) con lo cual la condición de diseño está completamente asegurada (Figura 72). Este modelo mencionado para realizar la medición SNR, tiene la particularidad de hacerlo en tiempo real (en inglés, “live”) lo cual es ventajoso para probar diferentes escenarios de simulación con diferentes tipos de tramas y tiempos.

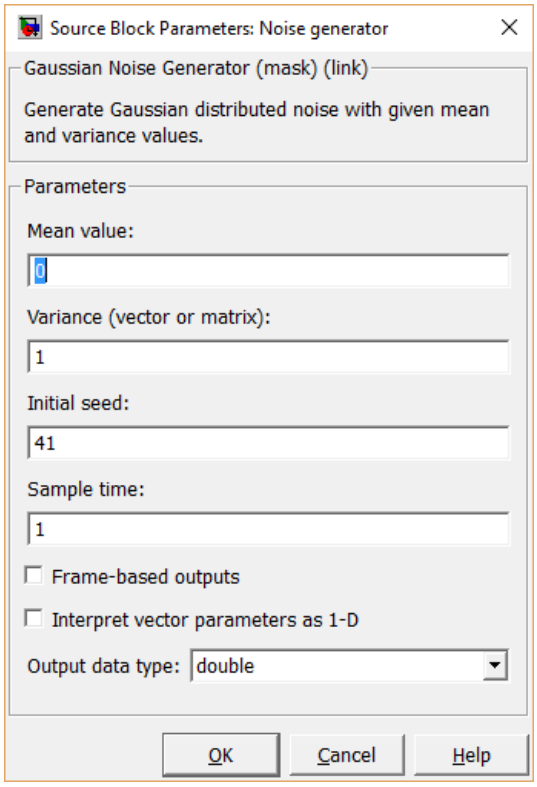


Figura 70. Valores de "Noise generator"

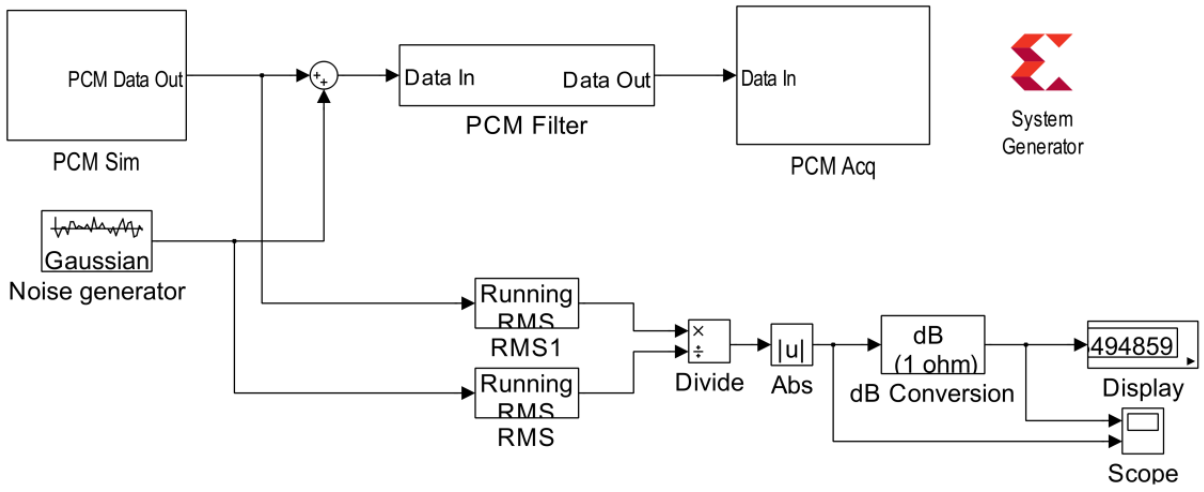


Figura 71. Modelo para la medición de SNR

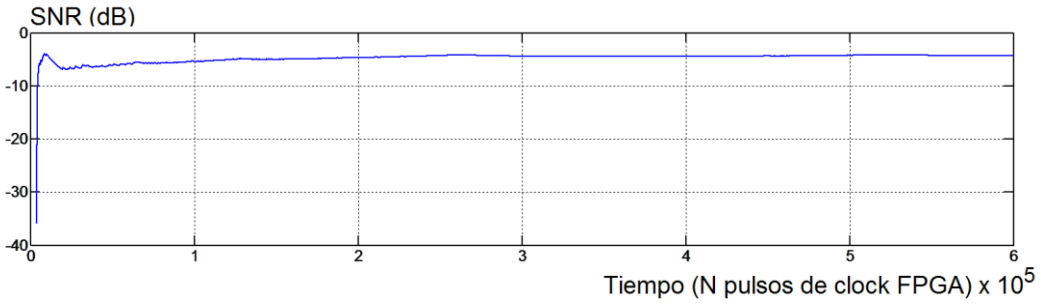


Figura 72. Medición del SNR a la salida del simulador PCM

3.4.1. FILTRO PCM

La etapa anterior al sistema adquisidor, debe realizar un filtrado ya que como se describe en el capítulo de sistemas de telemetría, los datos PCM llegan sumergidos en ruido térmico o gaussiano. Por otro lado, esta plaqueta (y la mayoría que con FPGA de la familia Spartan 6) no tiene incluido un conversor analógico-digital, por ser de bajo costo. Estas dos condiciones limitan el diseño y aumentan los recursos utilizados de la FPGA para su implementación.

Este sub-sistema de filtro PCM, implementa un filtro CIC (filtros integradores y peines combinados en cascada. En inglés, "Cascaded integrator-comb"). Esta decisión es fundamental ya que al ser un filtro que no utiliza multiplicadores, no bajan los recursos y la complejidad computacional para su implementación. Su funcionamiento se basa en sumadores-restadores, y registros de desplazamiento (Hogenauer, 1981).

3.4.2. TEORIA DEL FILTRO CIC

Existen dos bloques básicos que conforman el filtro CIC, el integrador y el comb (o peine). El integrador, ver Figura 73, es básicamente un filtro IIR con un solo polo y un coeficiente de realimentación unitario que puede verse en la ecuación (10). La función trasferencia para un integrador en el plano "z" puede verse en la ecuación (11).

$$y[n] = y[n-1] + x[] \tag{10}$$

$$H_I(z) = \frac{1}{1 - Z^{-1}} \tag{11}$$

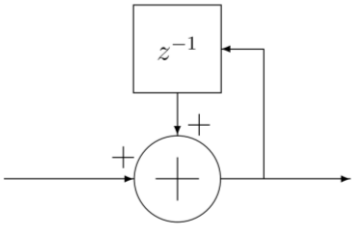


Figura 73. Diagrama básico de un integrador

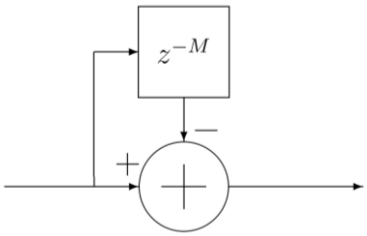


Figura 74. Diagrama básico de un comb

Un filtro comb a una tasa alta de muestro, f_s, para cambios de velocidades "R" es un filtro FIR de simetría impar descrito por la ecuación (12).

$$y[n] = x[n] - x[n - RM] \tag{12}$$

En esta ecuación (12) "M" representa un parámetro de diseño denominado "differential delay", es decir, un retardo diferencial. "M" puede ser un número entero positivo, generalmente es 1 o 2. La transferencia correspondiente a f_s está dada por la ecuación (13).

$$H_C[z] = 1 - z^{-RM} \tag{13}$$

Dónde:

- R = relación de diezmado o interpolación
- M = número de muestras por fase
- N = número de etapas de filtro

Cuando R=1 y M=1, la respuesta corresponde a una función de un filtro pasa alto con una ganancia de 20dB por década (6dB por octava), es decir, la inversa de un integrador.

Al construir un filtro CIC, lo que se hace es conectar en cascada la salida a la entrada, con N secciones de integradores y N secciones de filtros comb, sincronizados a f_s. Ver diagrama en bloques de la Figura 75.

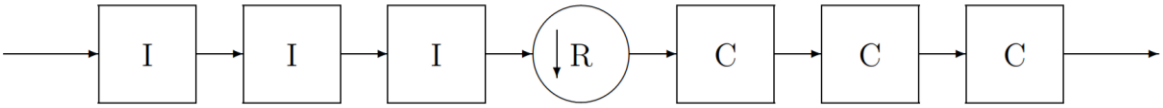


Figura 75. Filtro CIC de 3 etapas

La ecuación (14) nos muestra la función de trasferencia de un filtro CIC a fs (Crochiere & Rabiner, 1983). También nos pone de manifiesto que, aunque un CIC tenga integradores (que por sí mismos tienen una respuesta infinita al impulso), este filtro CIC es equivalente a N filtros FIR (donde cada uno de los cuales tiene una respuesta rectangular al impulso.) Dado que todos los coeficientes de estos filtros FIR son unitarios, y por lo tanto simétricos, un filtro CIC también tiene una respuesta de fase lineal y un retardo de grupo constante. El valor del módulo a la salida del filtro puede verse en (15), utilizando la aproximación del seno para ángulos pequeños $\sin(x) \approx x$ y realizando algunas operaciones algébricas podemos aproximar la ecuación (15) a la (14) para valores grandes de R.

$$H(z) = H_I^N(z).H_C^N(z) = \frac{(1 - Z^{-RM})^N}{(1 - Z^{-1})^N} = \left(\sum_{k=0}^{RM-1} Z^{-k}\right)^N \tag{14}$$

$$|H(f)| = \left|\frac{sin\pi Mf}{sin\frac{\pi f}{R}}\right|^N \tag{15}$$

$$|H(f)| \approx \left|RM\frac{sin\pi Mf}{\pi Mf}\right|^N \; para\; 0 \leq f < \frac{1}{M} \tag{16}$$

De la ecuación (16) podemos notar algunas características acerca de la respuesta del CIC. Una es que el espectro de salida tiene valores nulos (ceros) en múltiplos de $f = \frac{1}{M}$. Además, la región alrededor del cero es donde se produce "aliasing". Si definimos f_c como la frecuencia de corte pasabanda, entonces las regiones donde se produce el aliasing/imaging son $(i - f_c) \leq f \leq (i + f_c)$ para $f \leq \frac{1}{2}$ y $i = 1,2, \ldots, \left[\frac{R}{2}\right]$. Si $f_c \leq \frac{M}{2}$, entonces el máximo ocurrirá en el borde inferior de la primera banda, $1 - f_c$. El diseño del sistema debe tener esto en cuenta y ajustar R, M y N según sea necesario.

Otra condición de trabajo operativa que podemos notar es que la atenuación de la banda de paso es función del número de etapas. Como resultado, mientras aumentamos el número de etapas, mejora el rechazo de alias, también aumenta la selectividad de la banda de paso. También podemos

ver que la ganancia de continua del filtro es una función de los cambios de velocidad (Frerking, 1994).

Para el análisis de bits necesarios, primero determinamos la ganancia G a la salida del filtro CIC decimador según la ecuación (17).

$$G = (RM)^N \tag{17}$$

Asumiendo una aritmética de complemento a dos, podemos usar este resultado para calcular el número de bits requeridos para el último filtro peine debido al crecimiento de los bits. Si B_{in} es el número de bits de entrada, el número de bits de salida B_{out}, es:

$$B_{out} = \lceil N log_2 RM + B_{in} \rceil \tag{18}$$

También resulta que los bits B_{out} son necesarios para cada integrador y etapa de peine. La entrada debe ser extendida a los bits B_{out}, aunque los bits menos significativos pueden ser truncados o redondeados en etapas posteriores.

Debido a la pendiente de la banda de paso y, por lo que se obtendrá, la banda de paso utilizable estrecha, muchos diseños de CIC utilizan un filtro FIR adicional a la baja velocidad de muestreo. Este filtro igualará la pendiente de la banda de paso y realizará un cambio de velocidad baja, usualmente por un factor de dos a ocho. En muchos diseños de CIC, el cambio de velocidad R es programable. Dado que el crecimiento de los bits es una función del cambio de velocidad, el filtro debe diseñarse para procesar tanto los cambios de velocidad más grandes como los más pequeños. El cambio de velocidad más grande dictará el ancho total de los bits de las etapas, y el cambio de velocidad más pequeño determinará cuántos bits deben mantenerse en la etapa final.

3.4.3. IMPLEMENTACIÓN DEL FILTRO CIC

Recordando que los filtros CIC son usados para realizar grandes cambios en la velocidad de muestreo de sistemas digitales. Se basan tanto en las estructuras de decimación como en las de interpolación. Los filtros CIC no contienen multiplicadores; consisten sólo en sumadores, sustractores y registros. Se emplean típicamente en aplicaciones que tengan una frecuencia de muestreo alta; es decir, la velocidad de muestreo del sistema es mucho mayor que el ancho de banda ocupado por la señal (MathWorks, s.f.).

El bloque CIC posee una solo puerto de entrada y un solo puerto de salida, x_n e y_n, M es el retraso (latencia) diferencial. En la configuración de decimador, se reduce la tasa de muestreo por un factor de R tomando sub-muestras de la salida de la última etapa del integrador.

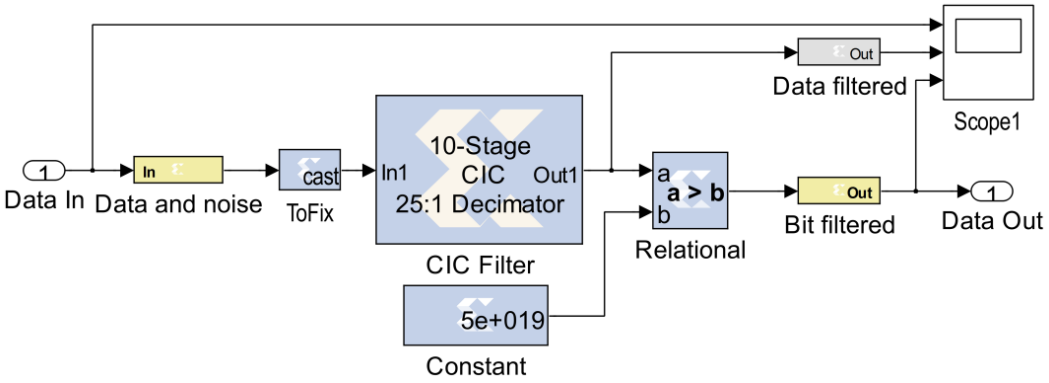


Figura 76. Modelo del filtro PCM implementado

Como se puede ver en la Figura 76, el modelo implementado consta de un filtro CIC decimador de 10 etapas, que si bien consume grandes recursos y área de la FPGA, nos asegura un buen filtrado que cumple con un alto nivel de exigencia en el diseño. Por otro lado, debemos garantizar que el ancho de cada bit se mantenga igual a la salida del filtro en comparación con su entrada, por este motivo adaptamos de manera empírica un filtro 25:1, lo cual nos genera datos a la salida en complemento a dos de 68 bits. Este último dato (número de 68 bits) es comparado con un umbral para generar un pulso digital a la salida del sub-sistema. Mediante pruebas empíricas y varias simulaciones en paralelo, se pudo ver que el valor máximo a la salida del filtro corresponde a 7.10^{19} cuentas y un valor mínimo de 3.10^{19} cuentas. Como resultado, se estableció un umbral de 5.10^{19} cuentas en la media geométrica entre el máximo y el mínimo. En otras palabras, toda señal de salida del filtro que supere al umbral se corresponde con un "1" lógico y toda señal por debajo del mismo se corresponde a un valor de "0" lógico.

En las Figura 77, Figura 78, Figura 79 Figura 80 se pueden ver las señales intervinientes en el sistema (Datos PCM ideales sin ruido, Datos PCM con ruido adicionado, Datos PCM a la salida del filtro CIC y finalmente Datos PCM reconstruidos mediante la comparación con una constante) tanto en el dominio del tiempo como en el dominio de la frecuencia, y las mediciones finales del modelo se pueden ver en la Figura 81 donde se puede ver el efecto del filtrado y el efecto sobre la comparación final para reconstruir la señal de datos PCM.

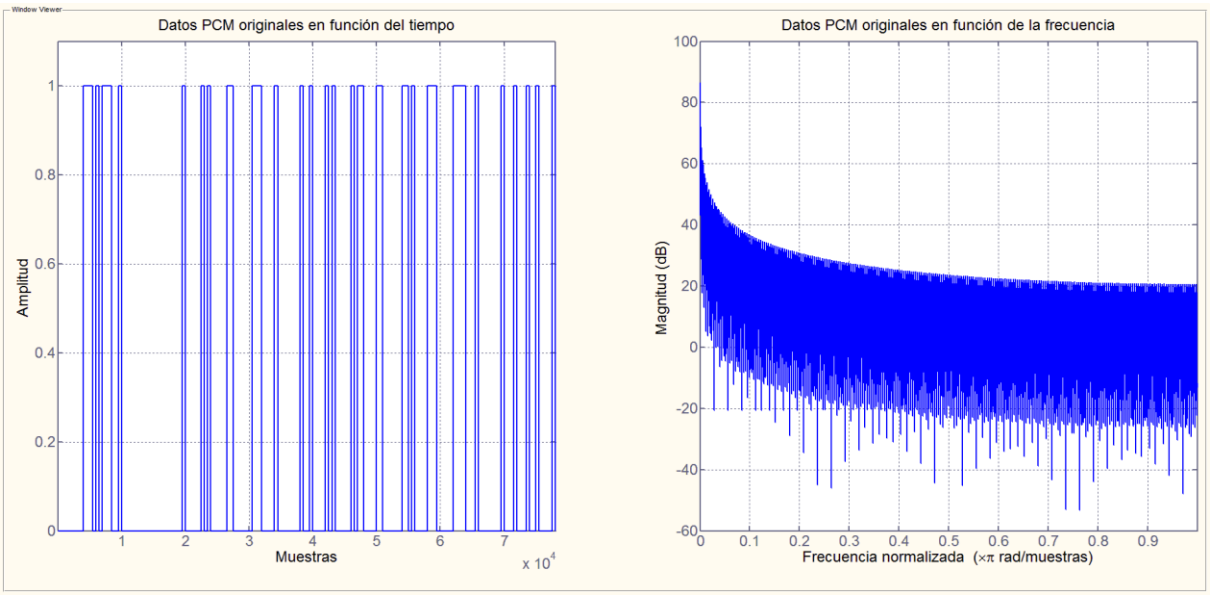


Figura 77. Análisis espectral de la señal PCM ideal del simulador

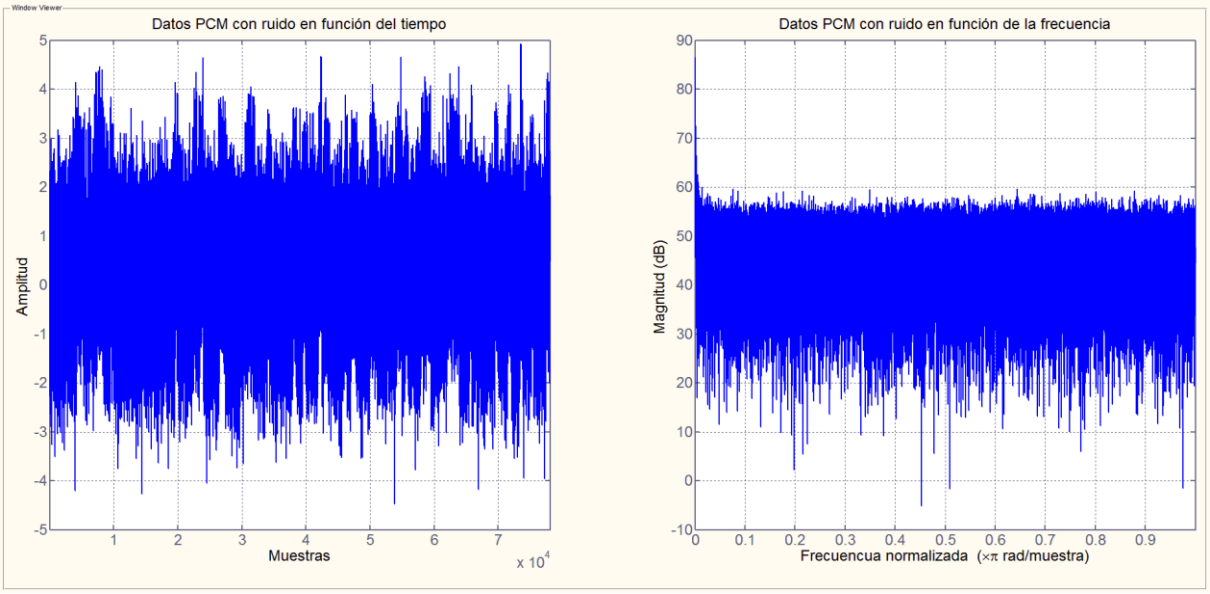


Figura 78. Análisis espectral de la señal PCM del simulador con ruido gaussiano adicionado

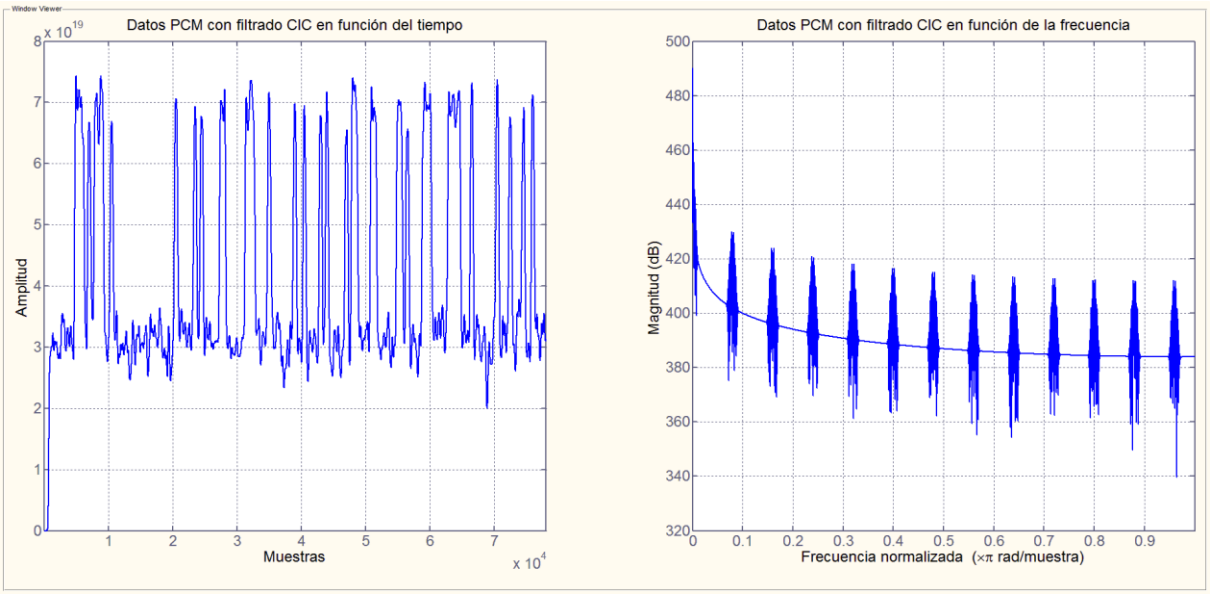


Figura 79. Análisis espectral de la señal PCM a la salida del filtro CIC

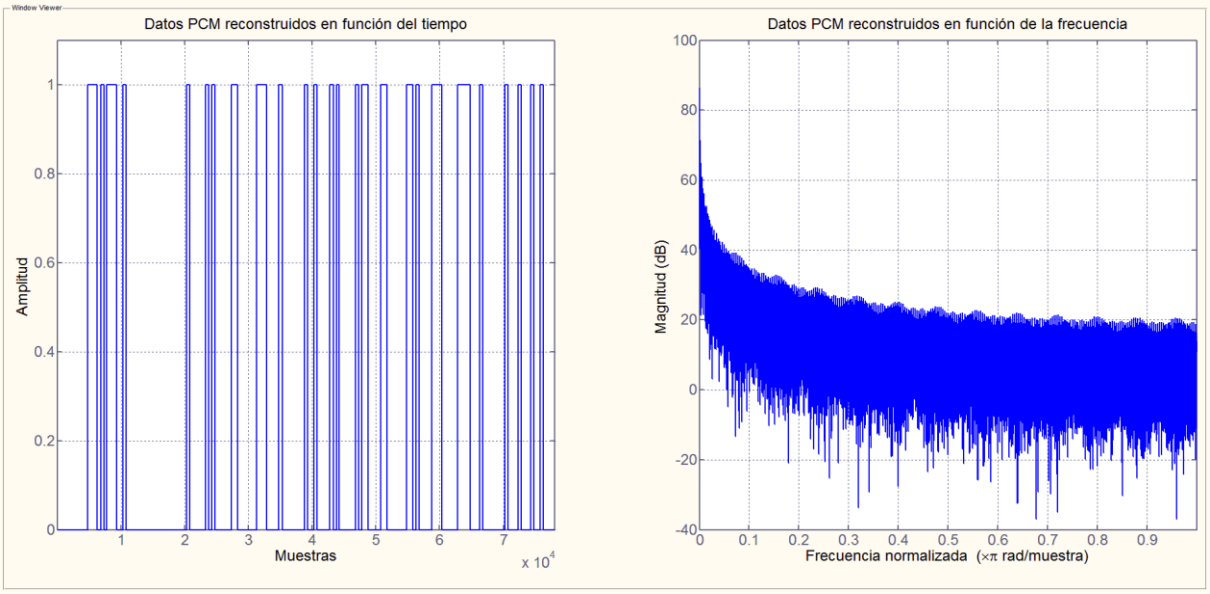


Figura 80. Análisis espectral de la señal PCM reconstruida

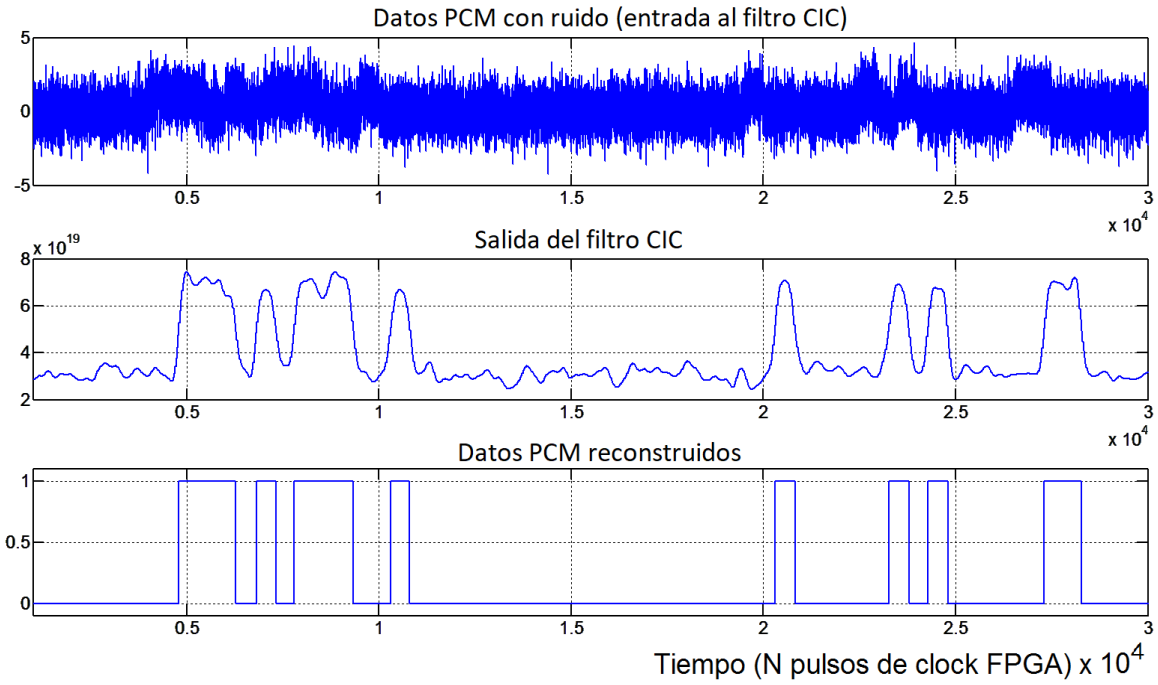


Figura 81. Medición de la respuesta del filtro PCM en función del tiempo

3.5. VERIFICACIONES DEL SISTEMA

Inicialmente, se programó al simulador de datos PCM con una trama conocida y fija para utilizarla como patrón o señal de referencia y analizar el comportamiento del adquisidor.

El hardware FPGA se lo programa con el modelo de la Figura 82, y la trama de referencia ingresa a través del gateway "Data PCM In". La salida del sistema "Out PCM Data" es inyectada directamente en un puerto serie de una computadora y dichos datos capturados fueron almacenados en un vector denominado "serialdata" (ver Figura 83).

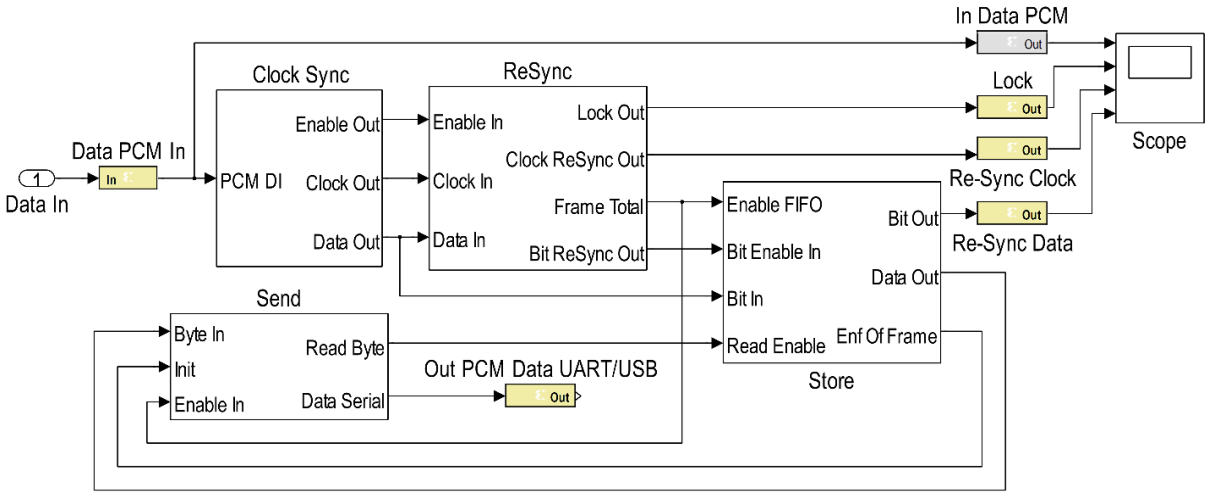


Figura 82. Modelo de adquisidor

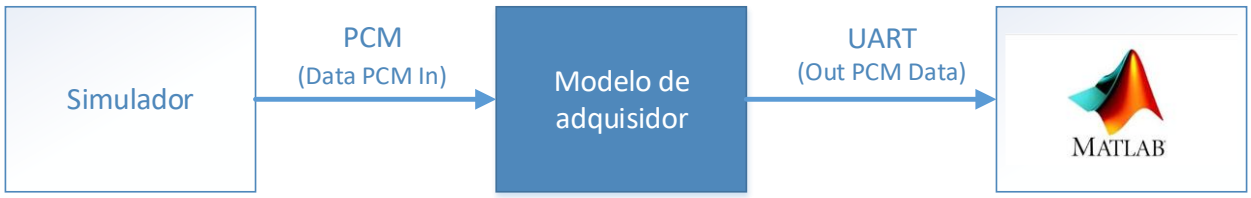


Figura 83. Diagrama en bloques simplificado de la prueba del Modelo de adquisidor

La idea general de este análisis consistió en generar dos matrices de datos donde las columnas representan el número de cada trama, una a continuación de la otra; y las filas son cada canal de esa misma trama PCM. La primera matriz es para los datos capturados por el adquisidor, y la segunda matriz es la de datos del simulador (patrón o referencia).

De esta manera, se pudo realizar una comparación entre estas dos matrices, la adquirida (la salida del sistema adquisidor) y la simulada (la entrada del sistema adquisidor), y así se pudieron analizar los errores que se producen en el modelo propuesto.

A su vez, estas matrices fueron convertidas a imágenes utilizando una escala de color HSV[4] para tener una visualización general de lo que ocurre en todo el proceso de adquisición.

Los datos adquiridos se almacenaron en un vector de una única fila llamado "serialdata" y los convertimos a la matriz propuesta denominada "serialdatamatrix. Esta matriz se la convirtió a imagen y el resultado se muestra en la Figura 84.

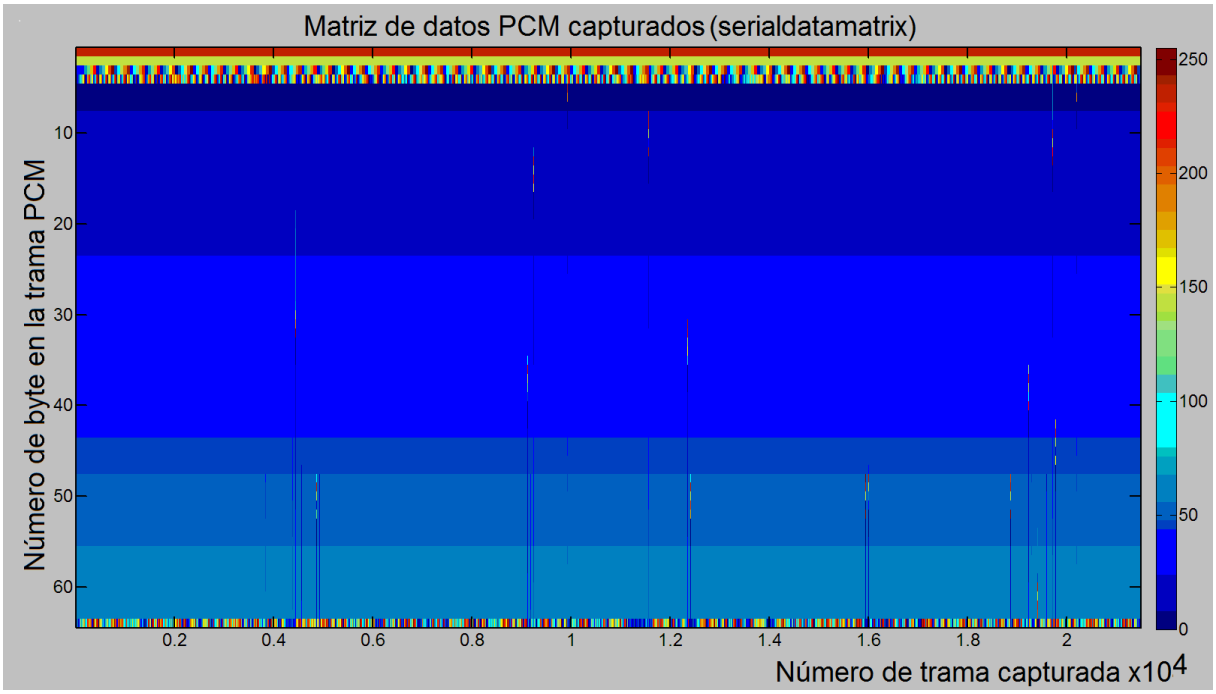


Figura 84. Imagen de la matriz de datos PCM con escala de colores HSV

[4] El modelo HSV (del inglés Hue, Saturation, Value – Matiz, Saturación, Valor), define un modelo de color en términos de sus componentes.

Recodando la naturaleza de la trama PCM, esta contiene información de sincronismo e información en la carga útil, la que es de interés es esta última, ya que la información de sincronismo no trae datos en sí. Por esta razón, a esta matriz/imagen se le recortaron los primeros 4 bytes (es decir las cuatro primeras filas, donde se hallan los 2 primeros bytes de la palabra de sincronismo y un contador de 16 bits) y el último byte (última fila, que representa un cálculo de comprobación de errores), y así trabajar solamente con los datos de interés. En la Figura 85 podemos ver el resultado del recorte mencionado.

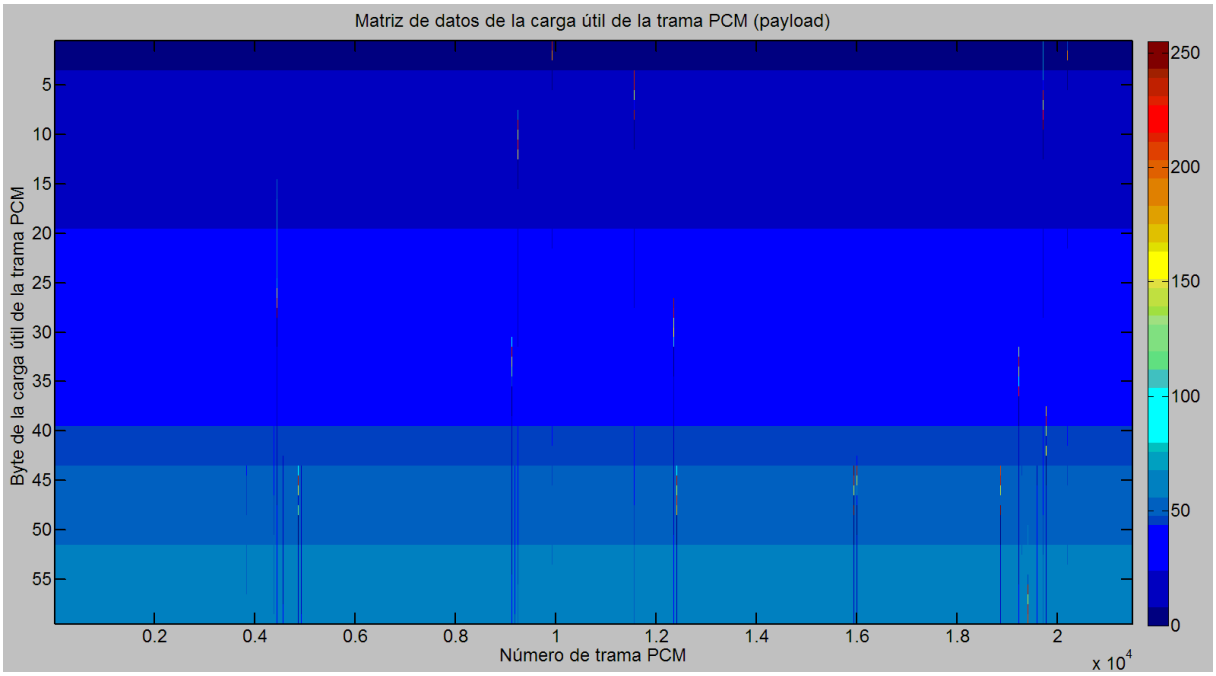


Figura 85. Imagen de la matriz de la carga útil con escala de colores HSV

Estos mismos pasos para la generación de la imagen/matriz se repitieron con los datos de entrada (el patrón de salida del simulador) y así obtener una imagen de referencia. Esa matriz se llamó "datamatrix".

Una vez que se obtuvieron ambas matrices, la imagen patrón y la imagen adquirida, para hallar diferencias entre lo adquirido y lo simulado se restan las matrices y de esta manera podemos determinar que bytes se recibieron sin problema (resultado 0) o si hubo errores de interpretación en el adquisidor (el resultado es distinto de 0). Esta nueva matriz resta la llamamos "dif".

Finalmente, a dicha matriz diferencia la convertimos en valores lógicos: es decir que sea "0" cuando la diferencia es nula entre los datos adquiridos reales y el patrón de la simulación, y "1" si alguna diferencia no es nula. Entonces podemos visualizar una matriz que nos da información de cómo y dónde se introducen errores en el sistema de adquisición (marcas de color en la Figura 86).

$$dif(m,n) = serialdatamatrix(m,n) - datamatrix(m,n)$$

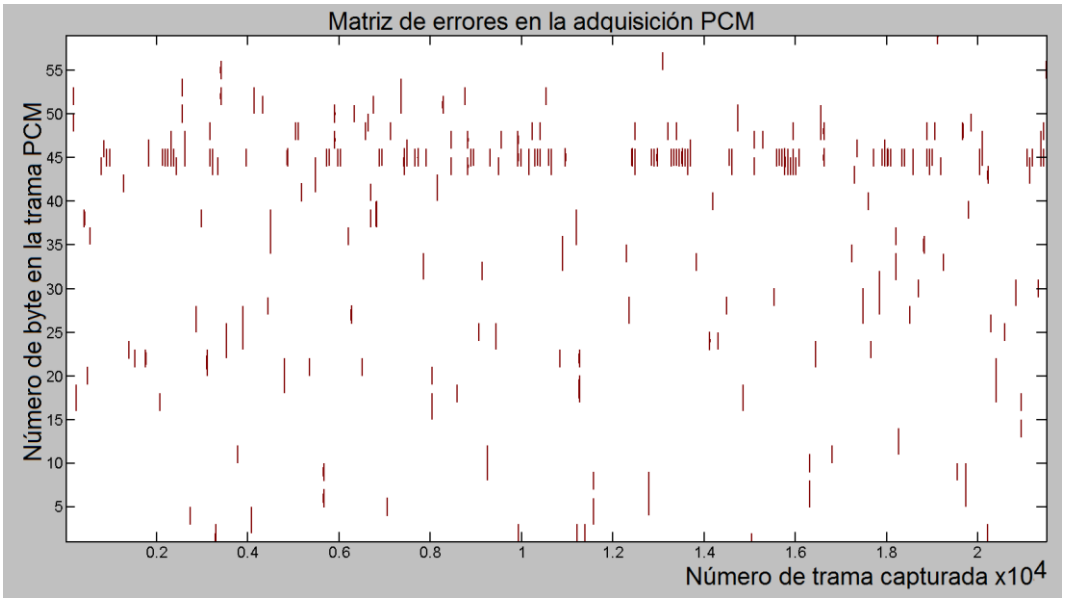


Figura 86. Mapa de errores encontrados

Analizando el total de datos de la matriz, de un total de 1.375.232 bytes (21.488 tramas con una tasa de 5ms por trama y un tamaño de trama de 64 bytes) se hallaron 2.102 errores (bytes corruptos).

Si al total de datos adquiridos le restamos los primeros 4 bytes y el último (tal como se realizó en el análisis de las imágenes), teniendo un total de 1.267.792 bytes (21488 tramas con 59 bytes en la carga útil). Si calculamos un porcentaje de bytes erróneos con la cantidad de bytes (de la carga útil de la trama PCM), nos da una eficacia del sistema adquisidor del 99,83%.

El error máximo introducido por el sistema adquisidor es de 0,165%, en el mayor de los casos, y estamos suponiendo que el error se produce en un byte entero, es decir que los 8 bits son erróneos.

Esto se debe a que el sistema adquisidor se engancha constantemente por cada trama recibida y no en una única vez como en muchos sistemas comerciales. Por lo tanto, si se produce un error de adquisición en los datos de la mitad de una trama, al encontrar la palabra de sincronismo de la próxima, se vuelve a acomodar. Esto comprueba que, si bien el sistema introduce un pequeño error en la cadena de telemetría, es mínimo y se auto-corrige, obteniéndose una mayor robustez para las aplicaciones de misión crítica.

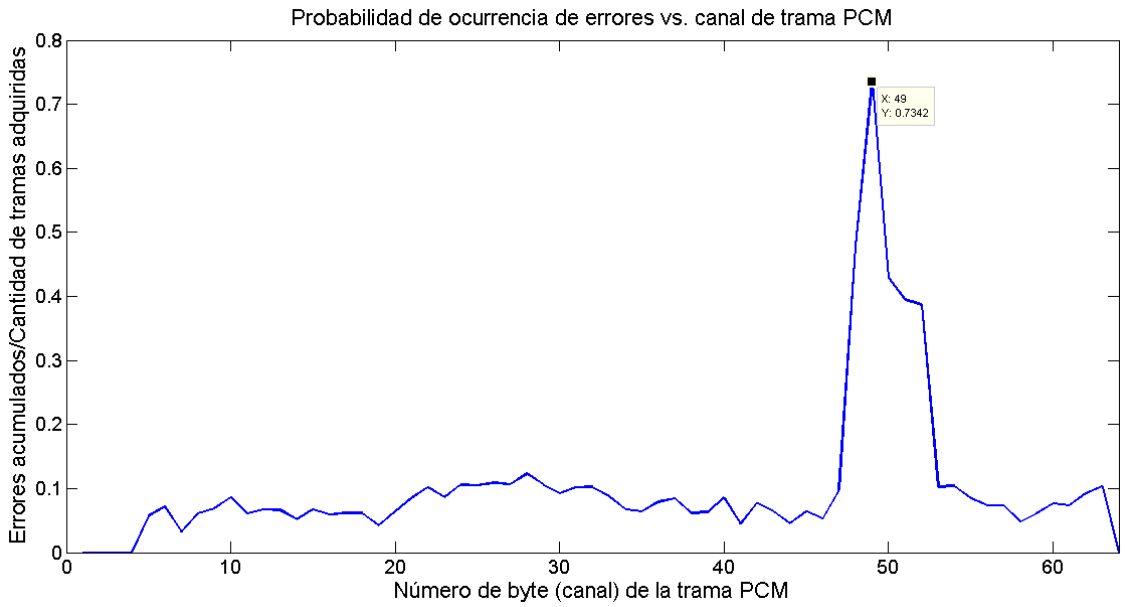


Figura 87. Probabilidad de ocurrencia de errores en el sistema de adquisición

En total se hallaron 230 tramas con errores, y un máximo de 5 errores por trama. Si se aplica una política de descarte de trama entera con algún error, en total sería un 1,07% de tramas descartadas.

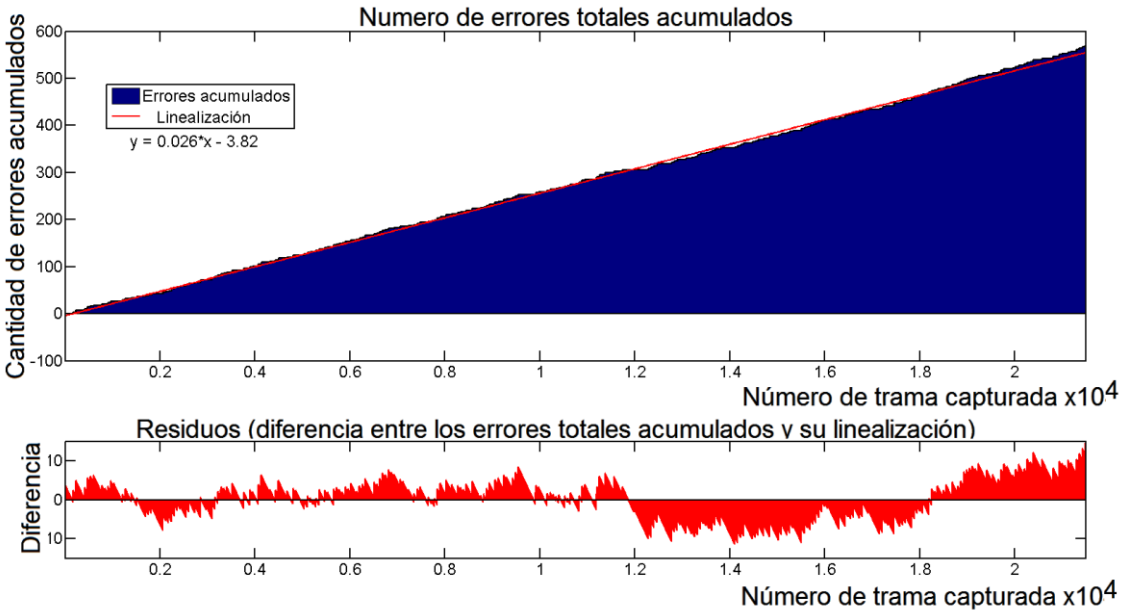


Figura 88. Errores acumulados totales en la adquisición

Por otro lado, a través de la matriz se puede realizar una sumatoria de todos los errores en cada canal a través del tiempo, y determinar que canal de la trama PCM tendrá una mayor sensibilidad a errores. En este caso podemos ver en la Figura 87 que los datos de los bytes 49, 50 y 51 son los que mayores errores tendrán a través del tiempo, o son más difíciles de interpretar por este sistema adquisidor, así que en estos canales se guardarán los datos menos críticos.

Otro análisis de importancia fue corroborar la cantidad de datos erróneos en función al tiempo, es decir, al número de trama PCM adquirida. Si se realiza un ajuste lineal automático, el mismo

software computacional nos proporciona la ecuación de una recta con la cantidad de errores totales acumulados “y” en función al número de tramas PCM adquiridas.

3.6. RESULTADOS

Luego de simular el sistema con Simulink, de realizar los primeros ensayos en hardware y de corregir errores encontrados, se compiló el modelo con System Generator el cual traduce dicho modelo al lenguaje de descripción elegido de Verilog. Esto no solamente crea el archivo binario que será grabado en el FPGA, sino que también genera un proyecto completo de ISE (Integrated System Environment). Con este proyecto el usuario puede corroborar la sintaxis de los archivos Verilog generados y caracterizar el sistema que se va a ejecutar. Finalmente, la implementación del desarrollo en la FPGA permitió analizar tiempos y comportamiento de todo el sistema, para ello se armó un banco de pruebas (ver Figura 89) con instrumentos de laboratorio como ser:

- Fuente de alimentación: GwInstek GPE-3323
- Generador de funciones: Picotest G5100A
- Osciloscopio: Agilent Techonologies DSO7054A
- Analizador lógico: Digilent Digital Discovery 2

A su vez, se utilizaron las siguientes aplicaciones de software para realizar la extracción y el análisis de datos.

- Matlab/Simulink R2012A (modificaciones en el modelo y generación del archivo BIT)
- Adept de Digilent (para grabar el archio BIT en el kit Atlys)
- Papilio Loader (para grabar el archivo BIT en el kit 3PX1 de Emtech)
- Termite 3.0 (monitor de puertos COM y RS232)

A su vez, fue necesario desarrollar un software que automatice el análisis de tramas que ingresan a la computadora. Este software se denominó “Verificador de Tramas” y posee las siguientes características:

- ✓ Identificación de palabra de sincronización
- ✓ Detección de contador
- ✓ Verificación de tramas contiguas
- ✓ Corroboración de longitud de tramas
- ✓ Almacenamiento en disco

Antes de comenzar estas pruebas, se grabó el modelo en otra FPGA independiente del sistema a probar. En este caso se tomó la placa de desarrollo ATLYS ya comentada anteriormente, Figura 90. Con esto se logra una independencia de los ensayos de realizar.

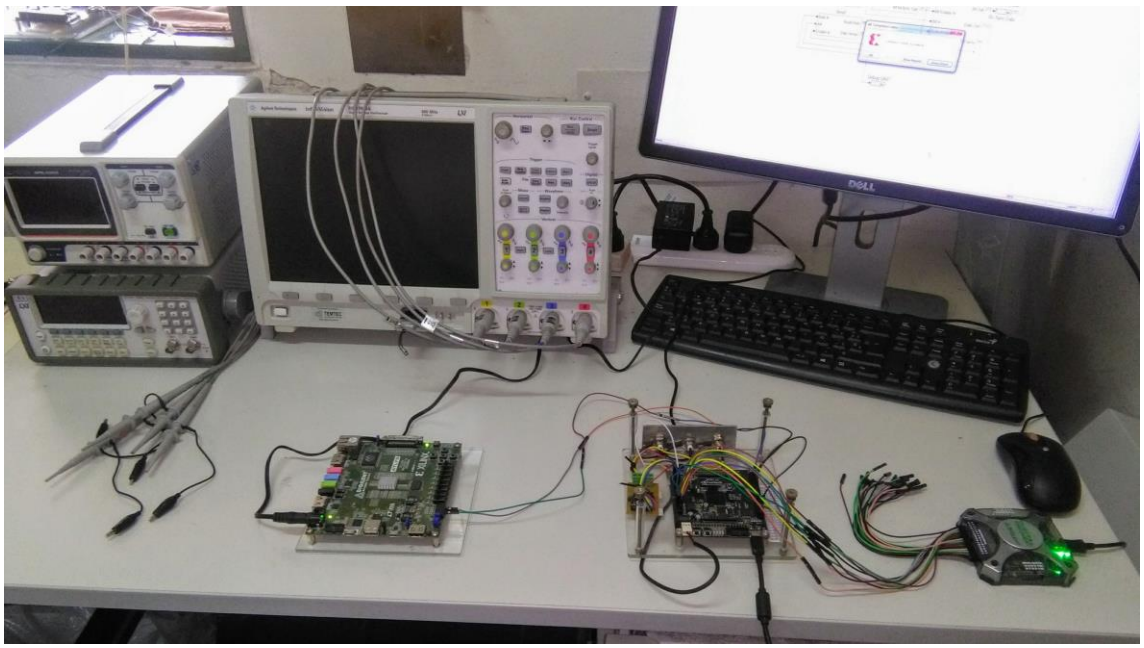


Figura 89. Banco de pruebas del laboratorio

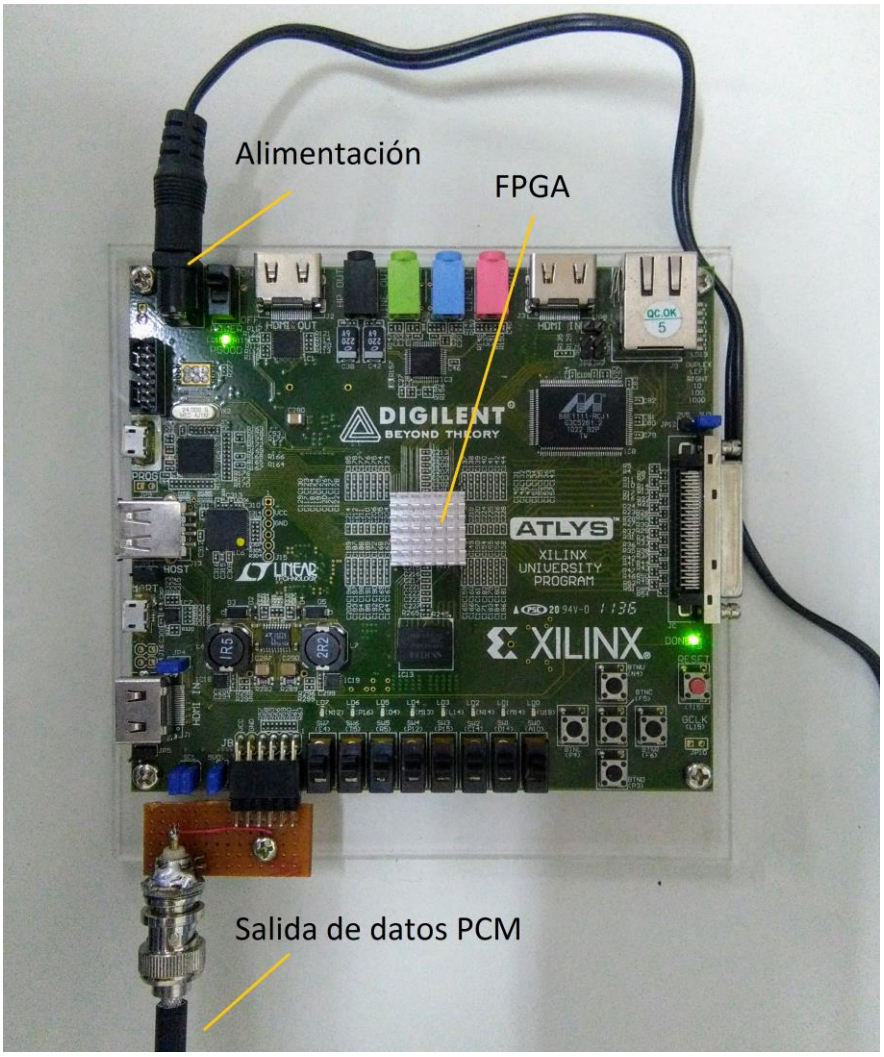


Figura 90. Generador/Simulador de datos PCM implementado en el kit de desarrollo Atlys de Digilent

Esta plaqueta simuladora de datos PCM se implementó en el kit de desarrollo Atlys de Digilent, y en su salida de datos trasmite una trama de longitud fija de 64bytes. Este simulador se lo conecta al adquisidor donde el modelo está implementado en el kit de desarrollo 3PX1 de Emtech, por medio de un solo cable de datos (y tierra) como se puede ver en la Figura 91.

La información de la trama simulada fija es la siguiente:

- Primer Byte = $235_{(10)}$ en decimal, es decir $EB_{(16)}$ en hexadecimal. Parte alta del sincronismo.
- Segundo Byte = $144_{(10)}$ en decimal, es decir $90_{(16)}$ en hexadecimal. Parte baja del sincronismo.
- Tercer y Cuarto Byte, parte alta y parte baja de un contador de 16 bits (*free running counter*).
- Del quinto al sesentaicuatroavo byte, número en decimal de la posición en la trama de ese mismo byte.

En esta trama no se utiliza información adicional para la comprobación de errores.

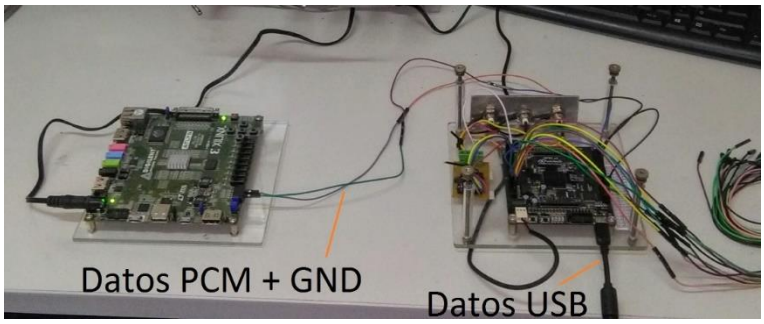


Figura 91 Conexión a través de una única línea de datos entre el Simulador implementado con el kit de desarrollo Atlys de Digilent y el adquisidor implementado en el kit 3PX1 de Emtech

La salida del adquisidor es USB, y está conectado a una computadora quien toma dicha conexión como un puerto serie virtual (Vitual COM), y se ejecuta la aplicación de software Verificadora de Datos para analizar que en adquisidor funcione correctamente.

A su vez, en el adquisidor existe una salida de datos adicional derivada de la entrada “Byte In” del bloque “Send” (Figura 50). Esta es una salida de datos de 8 bits en forma paralela, y es utilizada para poder conectar un analizador lógico (Digital Discovery de la firma Digilent, (Digilent, s.f.)). En señal “Bus” de la Figura 92 observamos como el ultimo byte (número 64) se acopla con la siguiente trama que empieza con su palabra de sincronismo (números 235 y 144 respectivamente). Este tipo de medición se la realizo aleatoriamente durante un tiempo prolongado con la finalidad de encontrar algún tipo de anomalía y no se ha detectado ninguna.

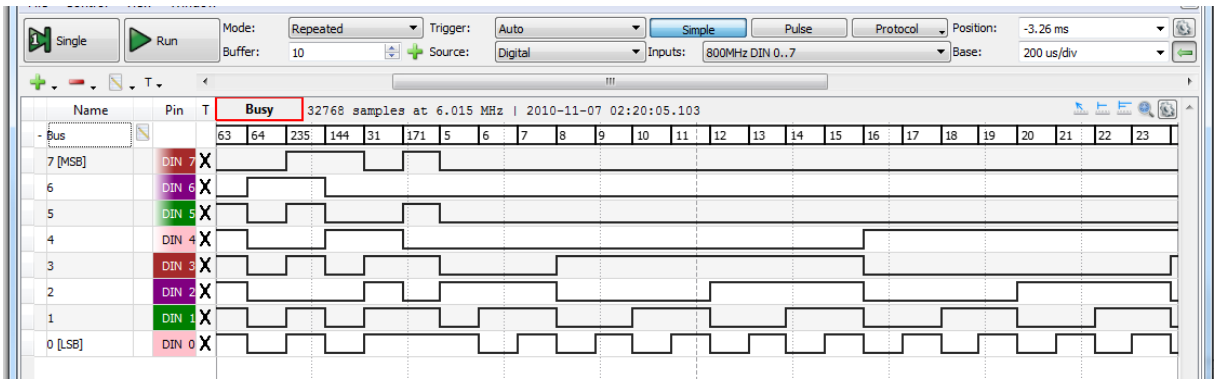


Figura 92. Análisis del adquisidor con un analizador lógico

Finalmente, en el modelo del adquisidor se agregaron algunos puntos de medición para extraer con un osciloscopio las formas de onda y verificar la ausencia de anomalías. En esta serie de mediciones se analiza primeramente que la marca de muestreo se encuentre a la mitad del bit PCM recibido y a partir de ello se regenere el "clock" interno (ver Figura 93). Seguidamente, se mide y comprueba que se detecte la palabra de sincronismo y se puede medir cuanto tiempo le demora al algoritmo llevar a cabo esta tarea (ver Figura 94). En tercer lugar, se verifica que la detección de sincronismo sea continua, es decir, que no se pierdan de tramas por la no detección del sincronismo (ver Figura 95). Luego se verifica que finalmente el reloj regenerado responda a lo esperado (ver Figura 96). Por último se analiza la trama convertida a formato UART 8N1 con una velocidad de 210400bps (ver Figura 97).

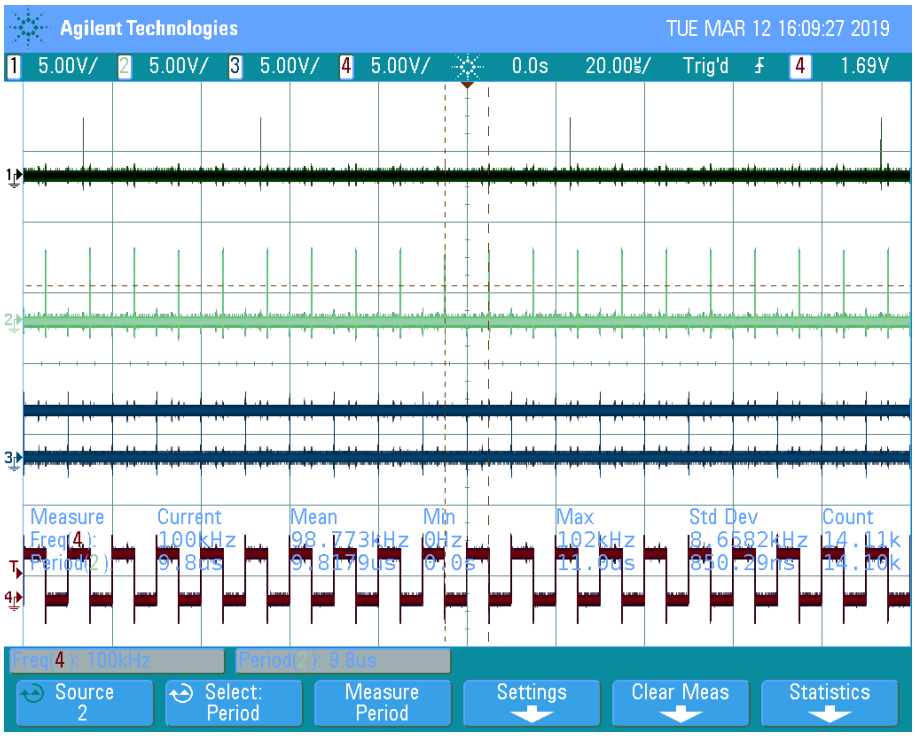


Figura 93. Medición de la marca de muestreo y el reloj regenerado.

Canal 1: NC marca de byte, Canal 2: Marca de bit, Canal 3: Datos PCM filtrados, Canal 4: Reloj regenerado.

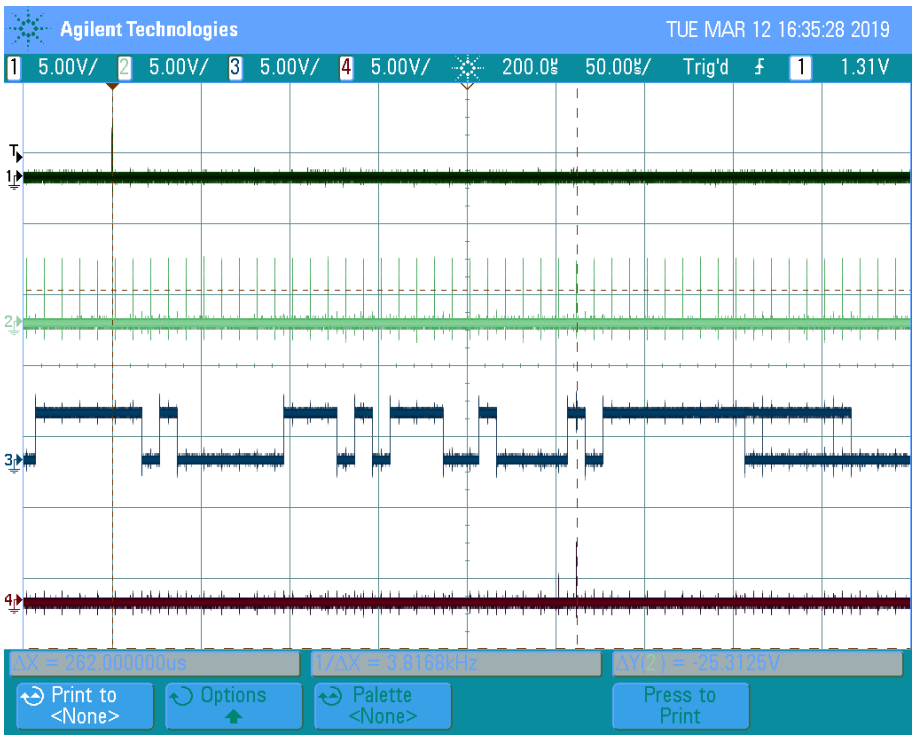


Figura 94. Medición del tiempo de detección de la palabra de sincronismo

Canal 1: Disparo, Canal 2: Marca de bit, Canal 3: Datos PCM entrantes, Canal 4: Sincronismo detectado.

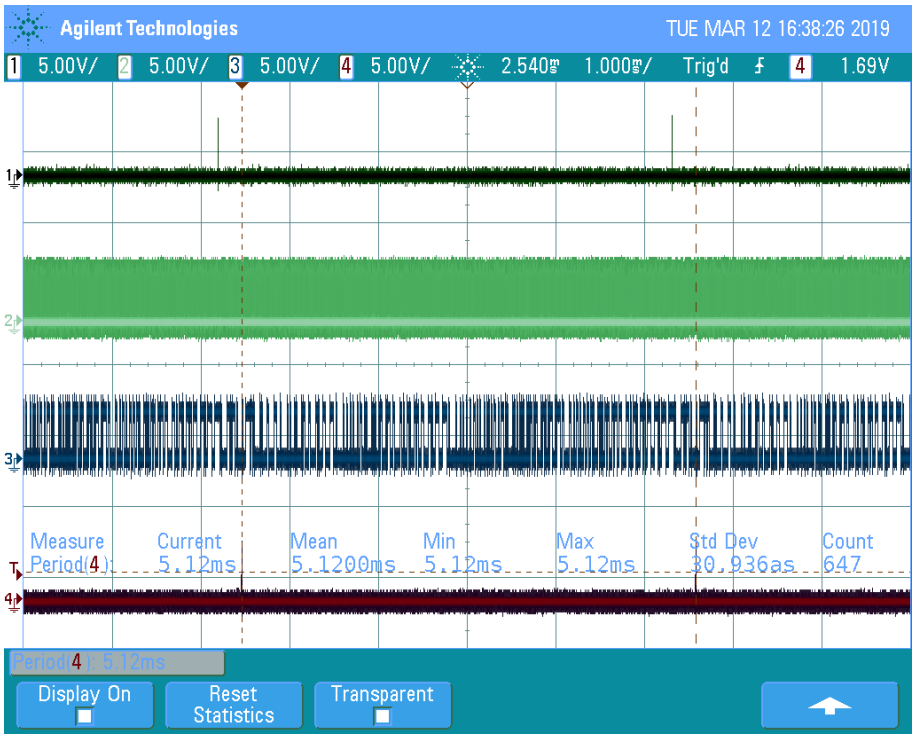


Figura 95. Medición del período entre detecciones de sincronismo

Canal 1: Disparo, Canal 2: Marca de bit, Canal 3: Datos PCM entrantes, Canal 4: Sincronismos detectado.

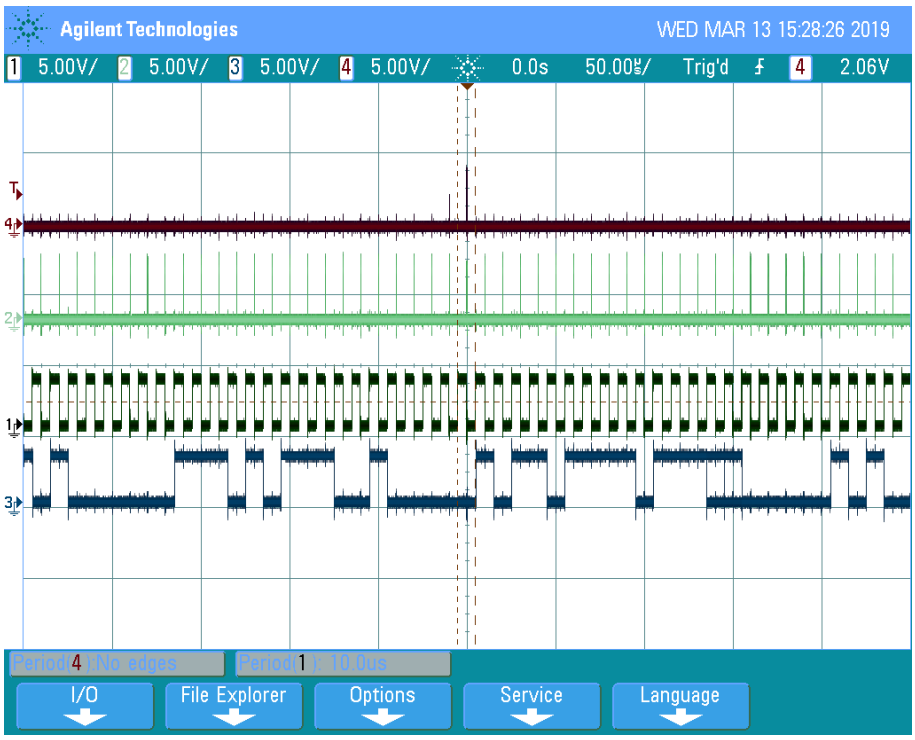


Figura 96. Medición del clock regenerado

Canal 1: Reloj regenerado, Canal 2: Marca de bit, Canal 3: Datos PCM, Canal 4: Sincronismo detectado.

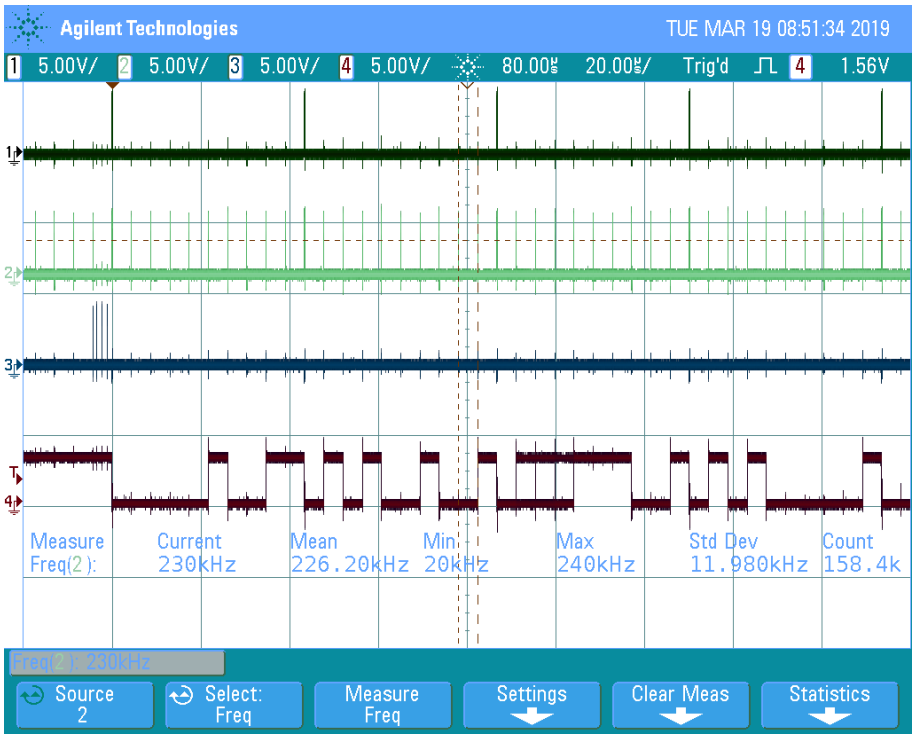


Figura 97. Medición de la trama UART

Canal 1: Marca de byte, Canal 2: Marca de bit, Canal 3: Inicio de transmisión, Canal 4: Trama UART.

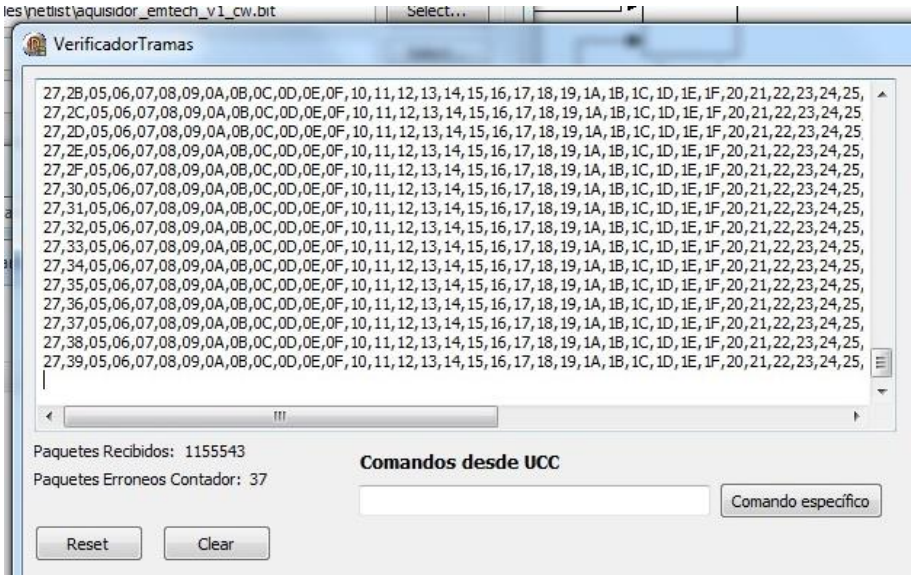


Figura 98. Verificación de tramas. El simulador se conecta directamente al adquisidor.

Según la Figura 98, se puede ver que se adquirieron 1155543 tramas completas de 64 bytes y se han recibido 37 tramas erróneas, esto significa que el adquisidor tuvo un 99,997% de efectividad (0,003% errores) en su funcionamiento.

Por otro lado, se tomó el desarrollo de una plaqueta específica con sensores de a bordo para vuelos con UAV (para ser evaluado en un avión de ala alta y un dron de 6 motores), que, si bien esto no forma parte del estudio específico de esta tesis, es necesario para probar toda la cadena de procesamiento de datos del sistema, ver Figura 99. A esta plaqueta se le adiciona una segunda plaqueta sumadora de señales digitales y ruido desarrollada para generar una señal PCM real, utilizando sumadores y mezcladores con amplificadores operacionales.

La misma posee las siguientes características:

- ✓ Procesador Cortex M4 de la firma STMicroelectronics
- ✓ Sensor integrado inercial con giróscopo, magnetómetro y acelerómetro
- ✓ Sensor de temperatura
- ✓ Sensor de aceleraciones en tres ejes
- ✓ Memoria SDRAM
- ✓ Manejo de memorias SD
- ✓ Conectividad con factor de forma PC/104

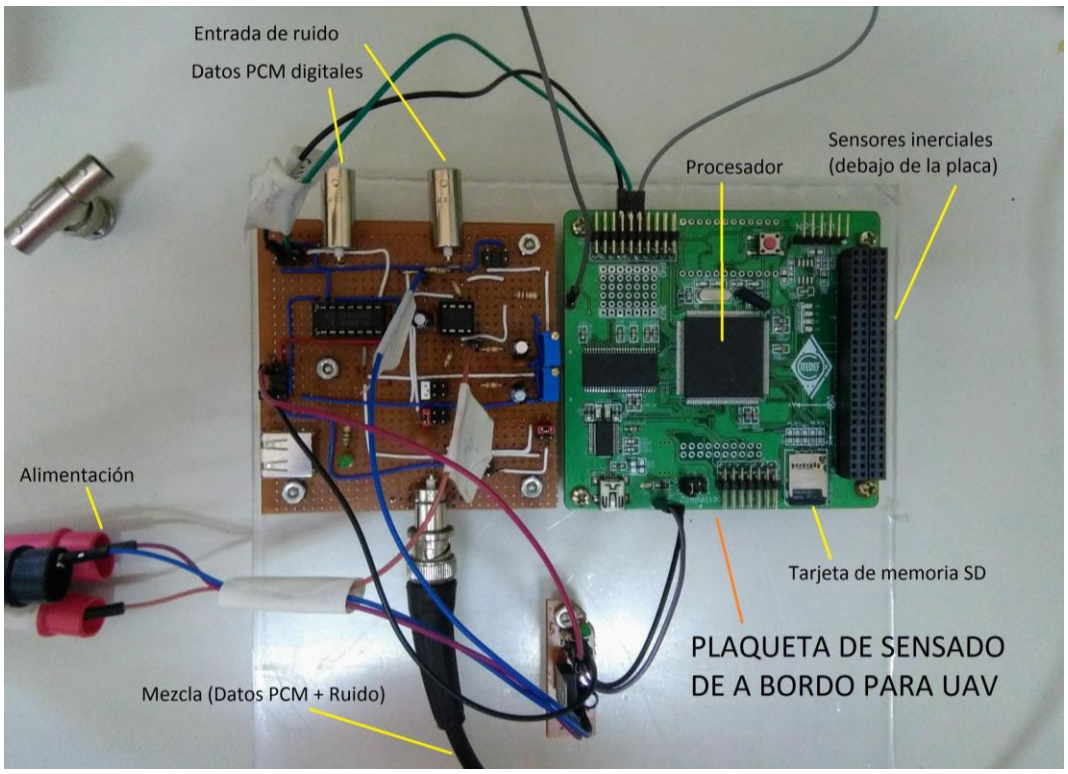


Figura 99. Plaqueta generadora de datos PCM con sensores integrados para aplicación con UAVs

Esta plaqueta posee tres funcionalidades principales, simular una trama fija de datos PCM (cumple la misma función que el kit Atlys mencionado anteriormente), simular una trama pseudo-aleatoria y trasmitir una trama real de datos de sus sensores internos.

A todas esas características se le suma el ruido y reducción del ancho de banda, efectos que se han descripto en el capítulo Descripción general de un sistema de telemetría - Reconstrucción de la trama PCM con la finalidad de reproducir todos los efectos de un sistema de telemetría real.

Lo primero a medir es la entrada de datos PCM, asegurarse que esta contenga un nivel de ruido adecuado para una correcta evaluación del sistema adquisidor. En esta medición los cuatro canales del osciloscopio están adquiriendo las siguientes señales:

1) Sincronismo de la trama simulada
2) Canal PCM puramente digital
3) Ruido térmico simulado por el generador de señales
4) Suma de PCM con ruido

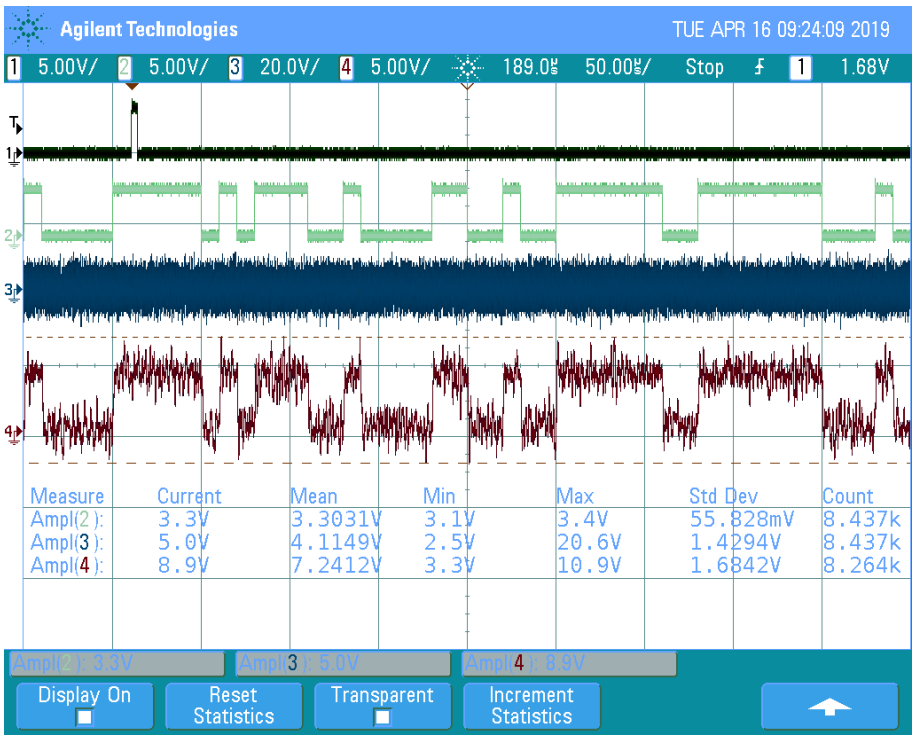


Figura 100. Adición de ruido en los datos PCM.

Con el osciloscopio se realizó la medición de la señal, el ruido por separado y ambas señales combinadas, las cuales se pueden visualizar en la Figura 100 obteniéndose los siguientes resultados:

- Nivel de señal PCM total: 7,68 V
- Nivel de ruido: 3,75 V
- SNR: -6,22 dB

La señal combinada, datos PCM digitales y ruido, representa una señal PCM realista. Esta señal PCM ingresa al adquisidor y se analizó nuevamente con el software, obteniéndose los siguientes resultados.

- ✓ Con el generador de ruidos apagado y datos fijos (Figura 101): 2919909 tramas recibidas y 98 errores, es decir una efectividad de 99,997%.
- ✓ Con el generador de ruidos encendido y datos fijos (Figura 102): 1086145 tramas recibidas y 869 errores, es decir una efectividad de 99,919%.
- ✓ Con el generador de ruidos encendido y datos pseudo-aleatorios (Figura 102): 2036510 tramas recibidas y 1700 errores, es decir una efectividad de 99,916%.
- ✓ Con datos reales de sensores y ruido: 510663 tramas recibidas y 14407 errores, es decir es decir una efectividad de 97,178%.

Existe una mínima pérdida en la efectividad en el adquisidor de datos PCM debido a la introducción de ruido en el mismo, pero esta no afecta la performance del mismo.

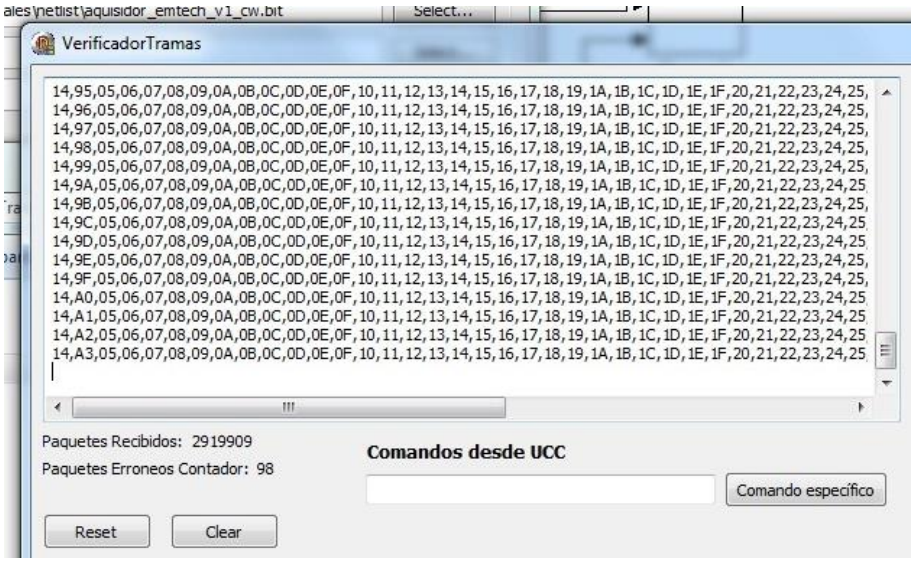


Figura 101. Verificación de tramas. La plaqueta de abordo con tramas simuladas y sin presencia de ruido se conecta al adquisidor.

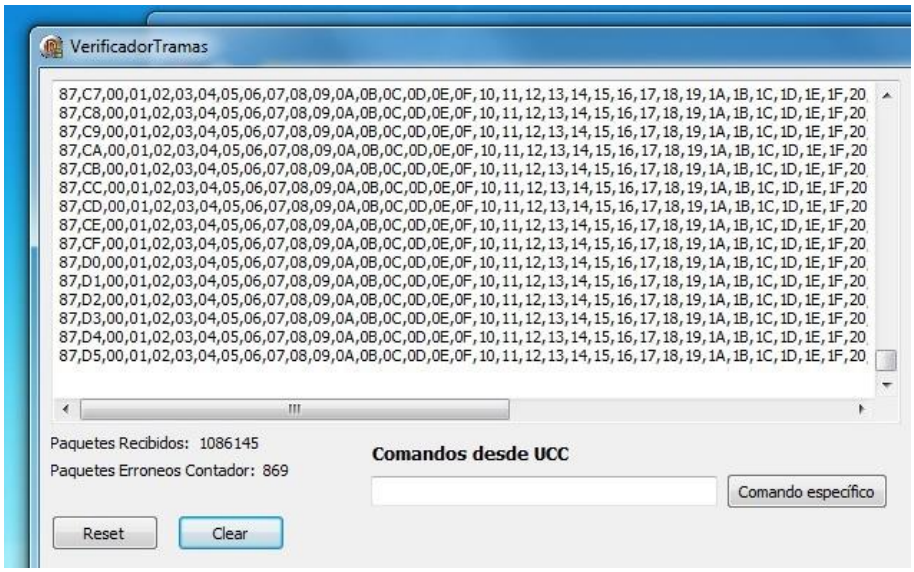


Figura 102. Verificación de tramas. La plaqueta de abordo con tramas simuladas y con la presencia de ruido se conecta al adquisidor.

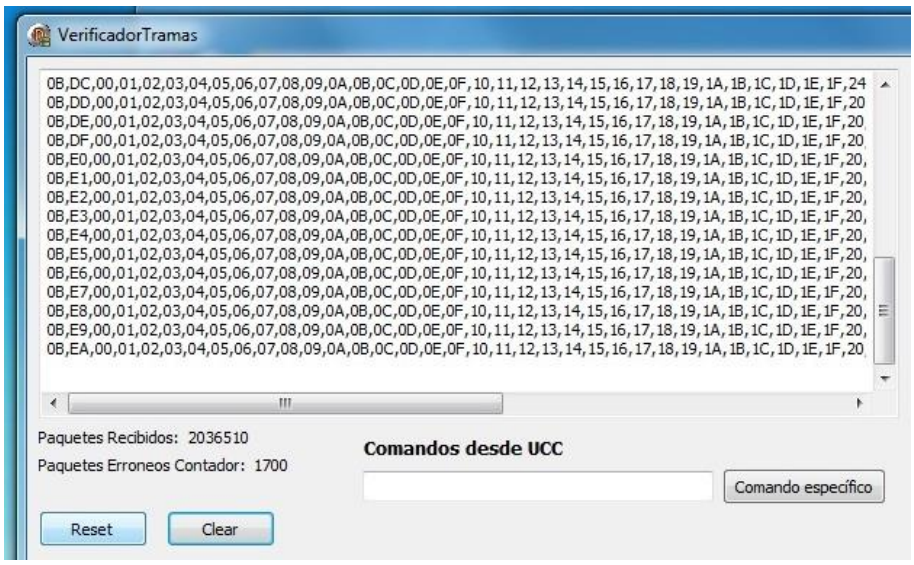


Figura 103. Verificación de tramas. La plaqueta de abordo con tramas simuladas y datos pseudo-aleatorios y con presencia de ruido se conecta al adquisidor.

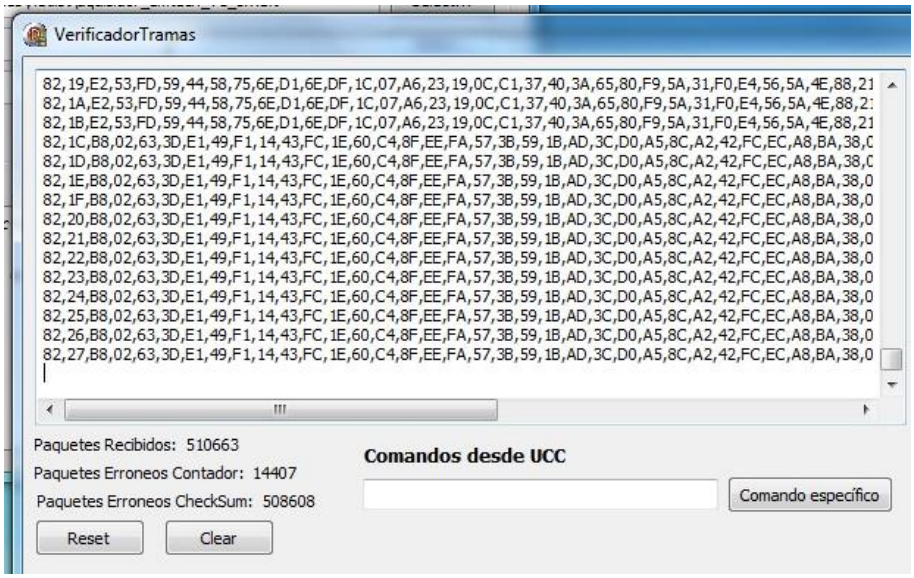


Figura 104. Verificación de tramas. La plaqueta de abordo con datos reales de sus sensores internos y con presencia de ruido se conecta al adquisidor.

Estos dispositivos programables han demostrado un gran rendimiento y facilidad al momento de implementar grandes sistemas digitales. En cuanto a los dispositivos FPGA y la implementación de modelos digitales en el entorno MATLAB / SIMULINK hace que el programador tenga una amplia versatilidad al momento del diseño y desarrollo, tanto en el ámbito de investigación profesional como en el académico.

- ✓ La firma Xilinx posee las soluciones más completas y de mejor desempeño en soluciones de lógica programable ya que es reconocida por inventar los FPGA. Además, cuenta con un gran número de herramientas de diseño, núcleos de propiedad intelectual (IP), servicios de diseño, ingeniería de campo y soporte técnico. Xilinx también tiene un programa de servicios y formación a nivel global.

- ✓ Mediante el lenguaje de descripción de hardware VHDL, la herramienta de diseño System Generator y el software de simulación Simulink, se diseñaron y analizaron los modelos digitales los cuales fueron examinados y comparados con otros sistemas, donde se concluyó que la tecnología FGPA es una alternativa para el entrenamiento y para el acondicionamiento de los laboratorios de CITEDEF.

- ✓ Los resultados obtenidos de la implementación de las modulaciones digitales en placa de desarrollo de EMTECH nos indica la factibilidad técnica y operativa para la realización de las practicas demostrativos referentes a sistemas de adquisición digitales, como así también las herramientas de software para diseño y desarrollo; tanto en VHDL, VERILOG y lenguaje MATLAB son muy prácticos y confiables.

- ✓ La tecnología FPGA es un elemento esencial para el análisis de las modelos digitales.

- ✓ La sincronización a nivel de bit y de tramas ha sido uno de los primeros problemas a resolver desde que las redes de computadoras se han desarrollado. En este trabajo se presentó una técnica derivada de técnicas ya conocidas, pero utilizando tecnología de la actualidad.

CONCLUSIONES

El dispositivo desarrollado ha cumplido con creces las expectativas iniciales en cuanto a su performance, adaptabilidad y re-programación del sistema, tanto en la etapa de simulación como en las pruebas de campo realizadas en condiciones reales con un dron hexacóptero. Esta experiencia fue realizada en los suburbios de Buenos Aires donde hay una presencia importante de interferencia radioeléctrica, debido mayoritariamente a la presencia en sus cercanías de antenas de transmisión de estaciones de radio AM y FM.

Los dispositivos programables utilizados han demostrado poseer, frente a las exigencias de uso, gran rendimiento operacional y relativa facilidad en su programación al momento de implementar Sistemas Digitales Dinámicos de envergadura. En cuanto a la programación de los dispositivos FPGA seleccionados para su uso, éstos permitieron la programación en bloques con lo cual se pudieron implementar modelos digitales en el entorno MATLAB / SIMULINK; de esta manera, el programador cuenta con una herramienta poderosa que le permite tener una amplia versatilidad al momento del diseño y desarrollo, tanto en el ámbito de investigación aplicada, en el ámbito profesional o en el entorno académico.

Realizamos una comparación con la placa L3-COM MTF800 disponible en el mercado internacional (insisto que este tipo de placa se vende llave en mano, es decir no permite ser re-programada, salvo por el fabricante). Cabe destacar que nuestros resultados están a la altura de dicha placa con la gran ventaja que ésta puede ser re-programada y adaptada a cualquier tipo de situación donde se requiera un Sistema Digital Dinámico de alta velocidad de adquisición de datos remotos.

Sistema desarrollado	**Sistema comercial**
Una entrada PCM de hasta 1Mbps con bit-synchonizer [5]	Dos entradas PCM de hasta 30Mbps con dos bit-synchonizer
No utiliza ADC	Si utiliza ADC
Conexión USB/UART	Conexión PCIe
Deconmutación de canales por software[6]	Deconmutación de canales por hardware
Hardware reprogramable para futuras modificaciones. No depende de una	Hardware con llave en mano, depende de una computadora de escritorio. Depende de un

[5] Expandible a “n” entradas dependiendo de la FPGA a utilizar y el área de la síntesis. En este caso se estima que pueden tenerse hasta 10 entradas independientes.

[6] En desarrollo al día de la fecha

computadora de escritorio, se puede utilizar en una notebook o laptop.[7]	Sistema Operativo y posee un software propietario lincenciado.[8]
Palabra de sincronismo de 16 bits programable	Palabra de sincronismo de 64 bits configurable
Precio en Argentina: USD 1000USD[9]	Precio en Argentina: USD 26503
Soporta código: NRZ-L	Soporta códigos: NRZ-L, NRZ-M, NRZ-S, BiØ-L, BiØ-M, BiØ-S, RZ, randomized NRZ-L (11, 15, 17)

La tecnología FPGA resulta ser un elemento esencial para el análisis de los modelos digitales; posee la velocidad requerida y la capacidad y manejo de datos en forma dinámica adquiriendo, procesando y resguardando la información que recibe por la red RF.

Uno de los primeros problemas que se logró resolver, que históricamente existe desde que las redes de computadoras se han desarrollado, es la sincronización de tramas a nivel bit (bit a bit). En el presente trabajo se utilizó una técnica derivada de técnicas ya conocidas (filtrado digital, registro de desplazamiento y búsqueda de patrones), pero desarrollando dispositivos programables de alta performance combinando tecnología de vanguardia: FPGA + MATLAB/SIMULINK + XILINX SYSTEM GENRATOR.

Durante el presente trabajo, y luego de un estudio exhaustivo de hardware, programación y software disponible en el mercado que no tuviera impedimentos para su adquisición, se diseñaron y analizaron los modelos digitales mediante el lenguaje de descripción de hardware VHDL, la herramienta de diseño System Generator y el software de simulación Simulink de MATLAB. Las cuales fueron examinadas y comparadas con otros sistemas (comerciales - llave en mano y/o implementados con electrónica discreta); donde se concluyó que la tecnología FGPA es la mejor alternativa para el entrenamiento de programadores y para el acondicionamiento de los laboratorios de CITEDEF.

Por otro lado, comparado con las unidades comerciales, éste resulta ser de bajo costo; lo cual representa una enorme ventaja dad nuestra situación socio-económica y de fortalecimiento de la industria nacional.

Los resultados obtenidos con la implementación de las Señales Digitales Moduladas realizadas con la placa de desarrollo de EMTECH, nos muestra la factibilidad técnica y operativa cuando se

[7] Se puede utilizar un simple monitor RS232 en cualquier tipo de computadora portátil, no es necesario ningún software.

[8] Utiliza el software VTS 7.x baseline

[9] Precio estimado considerando la implementación de un nuevo prototipo.

requiere la realización de prácticas demostrativos con referencia a los sistemas de adquisición de señales digitales; como así también las herramientas de software para diseño y desarrollo; tanto en VHDL, VERILOG para la programación de hardware FPGA; como la implementación de modelos digitales en lenguaje SYSTEM GENERATOR / MATLAB. Todas estas herramientas resultaron ser muy prácticas tanto en simulación como en uso en el campo, y durante el desarrollo del sistema hasta las pruebas finales resultaron ser muy confiables.

Finalmente, continuaremos nuestra labor de desarrollo y construcción de este tipo de unidades con tecnología moderna, que será implementada con unidades de la firma Xilinx, ya que sus productos, poseen las soluciones más completas y de mejor desempeño cuando se trata de lógica programable. Esta firma es reconocida a nivel mundial por sus sistemas FPGA y, además, cuenta con un gran número de herramientas de diseño, núcleos de propiedad intelectual (IP), servicios de diseño, ingeniería de campo y soporte técnico. Xilinx también tiene un programa de servicios y formación a nivel global con disponibilidad a todos los usuarios de sus productos.

A lo largo de estos años de investigación y desarrollo se han participado de los siguientes congresos y seminarios, y se han publicado los siguientes artículos:

Publicaciones en revistas internacionales:

- FILTERING AND ACQUISITION OF PCM FRAMES USING SYSTEM GENERATOR, se publicó en la revista "International Journal of Electronics and Communication Engineering (IJECE)" (IASET, s.f.) con ISSN(P): 2278-9901; ISSN(E): 2278-991X en el Vol. 8, Issue 1, Dec - Jan 2019; 1-10 en la ciudad de Tamil Nadu, India.

International Academy of Science, Engineering and Technology

IASET

Connecting Researchers; Nurturing Innovations

Paper Id: IJECEJAN201901 Date: 17.12.2018

Certificate of Publication

This is to certify that the research paper title **"FILTERING AND ACQUISITION OF PCM FRAMES USING SYSTEM GENERATOR"** authored by **"ADRIAN STACUL & EDGARDO COMAS"** had been reviewed by the board and published in **"International Journal of Electronics and Communication Engineering (IJECE); ISSN Print:2278-9901; ISSN Online:2278-991X; VOL-8, ISSUE-1; DEC--JAN2019; Impact Factor (JCC): 5.2039; NAAS Rating: 3.80; ICV : 49.71; Impact Factor(Research Bib): .9372 / 2012".**

Deputy Editor-IASET

Editor- In- Chief-IASET

Artículos aceptados en revistas internacionales:

- FILTERING AND ACQUISITION OF SERIAL DATA FRAMES USING XILINX SYSTEM GENERATOR. El autor es Adrián Stacul y fue aceptado en Enero del 2020 y se publicará en Marzo del 2020 en la revista "International Journal of Reconfigurable and Embedded Systems (IJRES)" (IAES, s.f.)con ISSN 2089-4864 en la ciudad de Yogyakarta, Indonesia.
- A HARDWARE SYSTEM WITH ARM-BASED DATA PROCESSING FOR NANO-SATELLITES. Los autores son Adrián Stacul, Daniel Pastafiglia, Ariel Dalmas Di Giovanni, Martín Morales, Sergio Saluzzi, Gerardo García, Agustín Gadea y Ramiro Puga. Fue aceptado en Enero del 2020 y se publicará en Julio del 2020 en la revista "International Journal of Reconfigurable and Embedded Systems (IJRES)" con ISSN 2089-4864 en la ciudad de Yogyakarta, Indonesia.
- ROBUST OBJECT TRACKING IN INFRARED VIDEO VIA PARTICLE FILTERS", los autores son Edgardo Comas , Adrián Stacul y Claudio Delrieux. Fue aceptado en la revista "Electronic Letters on Computer Vision and Image Analysis (ELCVIA)" (ELCVIA , s.f.) con ISSN 1577-5097 de la ciudad de Barcelona, España.

Publicaciones en congresos nacionales:

- Dalmas Di Giovanni, Ariel; Igareta, Diego; Morales, Martín; Saluzzi, Sergio; Alvarez, Sebastián; García, Gerardo; Stacul, Adrián; Pastafiglia, Daniel; Comas, Edgardo. INSTRUMENTACIÓN DE A BORDO, NAVEGACIÓN E INTERPRETACIÓN DE LA INFORMACIÓN PARA LOS VUELOS REALIZADOS POR EL PLANEADOR ESTRATOSFÉRICO PERLAN II. Argentina. Capital Federal. 2019. Revista. Artículo Completo. Congreso. X Congreso Argentino de Tecnología Espacial. Asociación Argentina de Tecnología Espacial
- Adrián Stacul; Sebastián Alvarez. APLICACIONES DE SOFTWARE PARA VISUALIZACIÓN Y MONITOREO APLICABLES A VEHÍCULOS NO TRIPULADOS Y COHETES SONDA. Argentina. Buenos Aires. 2017. Libro. Artículo Completo. Congreso. Noveno Congreso Argentino de Tecnología Espacial.
- Adrián Stacul; Cristian Bruña. DESARROLLO E IMPLEMENTACIÓN DE UN SIMULADOR DE TRAMAS PCM UTILIZANDO SYSTEM GENERATOR DE XILINX. Argentina. Buenos Aires. 2017. Libro. Artículo Completo. Congreso. Noveno Congreso Argentino de Tecnología Espacial.
- Adrián Stacul; Sebastián Alvarez. APLICACIONES DE SOFTWARE PARA VISUALIZACIÓN Y MONITOREO APLICABLES A VEHÍCULOS NO TRIPULADOS

Y COHETES SONDA. ARGENTINA. Ciudad Autónoma de Buenos Aires. 2017. Libro. Artículo Completo. Congreso. Congreso Argentino deSsistemas Embebidos CASE 2017.

- Adrián Stacul; Cristian Bruña. DESARROLLO E IMPLEMENTACIÓN DE GENERADOR Y CODIFICADOR NRZ-L DE TRAMAS PCM UTILIZANDO SYSTEM GENERATOR DE XILINX. Argentina. Ciudad Autónoma de Buenos Aires. 2017. Libro. Artículo Breve. Congreso. Congreso Argentino de Sistemas Embebidos CASE 2017.
- Adrián Stacul; Felipe Diniello. BIT-SYNCHRONIZER IMPLEMENTATION IN A LOW-COST FPGA FOR UAV. Argentina. Ciudad Autónoma de Buenos Aires. 2017. Libro. Artículo Breve. Congreso. Congreso Argentino de Sistemas Embebidos CASE 2017.
- Edgardo Comas; Daniel Pastafiglia; Adrián Stacul; Cristian Bruña; Ariel Dalmas Di Giovanni; Martín Morales; Mauricio Burgos; Sergio Saluzzi. DISEÑO, DESARROLLO E IMPLEMENTACIÓN DE UNA ESTACIÓN TERRENA PARA COHETES SONDA. Argentina. San Justo. 2015. Libro. Artículo Completo. Congreso. VI Congreso de Microelectrónica Aplicada.
- Edgardo Comas; Daniel Pastafiglia; Cristian Bruña; Adrián Stacul; Ariel Dalmas Di Giovanni; Martín Morales; Sergio Saluzzi; Mauricio Burgos. DISEÑO, DESARROLLO E IMPLEMENTACIÓN DE UNA ESTACIÓN TERRENA PARA COHETES SONDA. Argentina. Buenos Aires. 2015. Libro. Artículo Completo. Congreso. VIII Congreso Argentino de Tecnología Espacial. Congreso Argentino de Tecnología Espacial
- Edgardo Comas; Daniel Pastafiglia; Cristian Bruña; Adrián Stacul; Sergio Saluzzi; Mauricio Burgos. MODERNIZACIÓN, SIMULACIÓN E IMPLEMENTACIÓN EN UNA COMPUTADORA ROBUSTA, DE UN SISTEMA DE MEDICIÓN ANGULAR MEDIANTE UN SENSOR SINCRÓNICO. Argentina. Buenos Aires. 2013. Libro. Artículo Completo. Congreso. Congreso Argentino de Sistemas Embebidos 2013. FIUBA

Libros

- Edgardo Comas; Adrián Stacul. CONTROL EN ESPACIO DE ESTADOS. Editorial CEIT. 2016. pag.181. isbn 978-987-1978-28-1

BIBLIOGRAFÍA

Brown. (s.f.). *Fundamentos de Logica Digital con Diseño VHDL* . Mc Graw Hill.

Bryant, J. (2013). Multipliers vs. Modulators. Obtenido de https://www.analog.com/media/en/analog-dialogue/volume-47/number-2/articles/multipliers-vs-modulators.pdf

Comas, E., Pastafiglia, D., Bruña, C., Dalmas Di Giovanni, A., & Stacul, A. (2015). *Diseño, Desarrollo e Implementación de una Estación Terrena para Cohetes Sonda.* Buenos Aires, Argentina: VIII Congreso Argentino de Tecnología Espacial.

Comas, E., Vescovo, E., & Legnani, W. (s.f.). Análisis del comportamiento caótico del flujo hipersónico en el vector GRADICOM II. *Congreso Argentino de la Tecnología Espacial, At Córdoba, Argentina, Volume: 9*.

Consultative Committee for Space Data Systems. (2000). *Radio frequency and modulation systems - part 1: Earth stations and spacecraft.* Washington, DC, USA : CCSDS Secretariat .

Crochiere, R., & Rabiner, L. (1983). *Multirate Digital Signal Processing.* Prentice Hall.

Digilent. (2015). *History of the FPGA.* Obtenido de History of the FPGA: https://blog.digilentinc.com/history-of-the-fpga/

Digilent. (s.f.). *Digital Discovery: Portable USB Logic Analyzer and Digital Pattern Generator*. Obtenido de https://store.digilentinc.com/digital-discovery-portable-usb-logic-analyzer-and-digital-pattern-generator/

Digilent. (s.f.). *FPGA – Configurable Logic Block*. Obtenido de FPGA – Configurable Logic Block: https://blog.digilentinc.com/fpga-configurable-logic-block/

Dostis, I. (s.f.). The effects of digital errors on PCM transmission of compandored speech. *The Bell System Technical Journal*, vol. 44, no. 10, pp. 2227-2243.

ELCVIA . (s.f.). *ELCVIA Electronic Letters on Computer Vision and Image Analysis*. Obtenido de ELCVIA Electronic Letters on Computer Vision and Image Analysis: https://elcvia.cvc.uab.es/

Emtech. (s.f.). *Kit de desarrollo 3PX1*. Obtenido de http://www.emtech.com.ar/producto/placa-3px1

Frerking, M. (1994). *Digital Signal Processing in Communication Systems.* Boston: Kluwer Academic Publishers.

Halsall, F. (1995). *Data Communications, Computer Networks and Open Systems.* (W. P. Inc., Ed.)

Hogenauer, E. (1981). An ecnomical class of digital filters for decimation and interpolation. *IEEE Transactions on Acoustics, Speech and Signal Processing*.

IAES. (s.f.). *International Journal of Reconfigurable and Embedded Systems (IJRES)*. Obtenido de International Journal of Reconfigurable and Embedded Systems (IJRES): http://ijres.iaescore.com/index.php/IJRES/index

IASET. (s.f.). *International Journal of Electronics and Communication Engineering (IJECE)*. Obtenido de https://www.iaset.us/journals/international-journals/international-journal-of-electronics-and-communication-engineering

IEEE SA . (17 de 3 de 2001). *1364-2001 - IEEE Standard Verilog Hardware Description Language*. Obtenido de standards.ieee.org/standard/1364-2001.html

IEEE SA. (2019). *1076-2019 - IEEE Standard for VHDL Language Reference Manual*. Obtenido de https://standards.ieee.org/standard/1076-2019.html

Keysight. (2012). Obtenido de Agilent Technologies - InfiniiVision 7000A Series Oscilloscopes: http://literature.cdn.keysight.com/litweb/pdf/5989-7736EN.pdf?id=1373609

MathWorks. (s.f.). *Matlab - The Language of Technical Computing*. Obtenido de https://www.mathworks.com/products/matlab.html

Maxfield, C. (2004). *The Design Warrior's Guide to FPGAs.* (M. G. Corp, Ed.)

McKinney, D. (2001). *Impact of Commercial Off-The-Shelf (COTS) Software and Technology on Systems Engineering.* Obtenido de http://www.incose.org/northstar/2001Slides/McKinney%20Charts.pdf

Oppenheim, A., & Schafer, R. (1989). *Discrete-Time Signal Processing.* Prentice Hall.

Rojas, L., Franco M, Z. E., & Pateti M, A. S. (2009). Diseños de circuitos electrónicos digitales utilizando la tecnología FPGA. *Universidad, Ciencia y Tecnología, 13(52), 250-258*. Obtenido de http://ve.scielo.org/scielo.php?script=sci_arttext&pid=S1316-48212009000300008

Schwartz, M. (s.f.). *Information, Transmition, Modulation & Noise.* Mc. Graw Hill.

ScienceDirect - ELSEVIER. (s.f.). *Nyquist Frequency*. Obtenido de Nyquist Frequency: https://www.sciencedirect.com/topics/engineering/nyquist-frequency

Stauth, J., & Sanders, S. (2008). Pulse-Density Modulation for RF Applications: The Radio-Frequency Power Amplifier (RF PA) as a Power Converter. *IEEE Annual Power Electronics Specialists Conference*.

Tektronix. (s.f.). *Frecuencímetro FCA3000*. Obtenido de http://www.tek.com/frequency-counter/fca3000-3100

Tektronix. (s.f.). *Osciloscopio TBS1202B*. Obtenido de http://www.tek.com/datasheet/digital-storage-oscilloscopes-12

Telemetry Standards. (2013). *IRIG Standard 106-13 (Part 1) - Chapter 4.*

Telemetry Standars. (2015). *IRIG Standard 106-15 (Part 1) - Appendix C - Pulse Code Modulation Standards* .

Texas Instruments. (2002). *TIA/EIA-232-F - Design notes.* Dallas - Texas: Texas Instruments.

Xilinx. (2020). *Xilinx*. Obtenido de Xilinx: www.xilinx.com

Xilinx Inc. (s.f.). *ISE Design Suite*. Obtenido de https://www.xilinx.com/products/design-tools/ise-design-suite.html

Xilinx Inc. (s.f.). *Spartan-6 Family Overview*. Obtenido de https://www.xilinx.com/support/documentation/data_sheets/ds160.pdf

Xilinx Inc. (s.f.). *Xilinx Applications*. Obtenido de https://www.xilinx.com/applications.html

Xilinx Inc. (s.f.). *Xilinx System Generator for DSP*. Obtenido de https://www.mathworks.com/products/connections/product_detail/product_35567.html

ANEXOS

4.1. FORMATO DE TRAMA PCM

La Trama PCM propuesta en el proyecto cumple con un formato interno, la disposición de los datos se puede ver en la Tabla 5 y Tabla 6. La misma es utilizada para enviar mensajes desde electrónica de a bordo ubicada dentro de un vector del tipo sonda y otra electrónica de a bordo para instrumentar un vehículo aéreo no tripulado.

Nº Canal	Nombre del canal Canal	Bytes	Observación	Posición
0	Sincronismo	2	EB90h	0
1	Contador	2	0 a 65535	2
2	IMU	24	Sensores inerciales y canales adicionales.	4
3	GPS	2	Mensaje RMC, GGA	28
4	Señal PPS	1	Señal de 1PPS de GPS	30
5	Altímetro ultrasonido	2	Altura medida por ultrasonido	31
6	Altímetro barométrico	2	Altura medida por presión.	33
7	Temperatura barométrica	2	Datos intermedios de temperatura.	35
8	Presión barométrica	2	Datos intermedios de presión.	37
9	Velocidad del aire	2	Tubo de pitot	39
10	Reservado para Temperatura	2	Sensor de temperatura adicional	41
11	Estados de Instrumentación	2	Información de "salud" de la instrumentación a bordo.	43
12	Reservado Telebar	2	Expansión Telebar.	45
13	Reservado Analógicas	2	Expansión entradas analógicas	47
14	Reservado Estados del sistema	2	Expansión entradas Digitales	49
15	Reservados	12	Canales no utilizados.	51
16	Checksum	1	Código de control de errores, XOR de todo menos del header	63

Tabla 5. Formato de datos de la Trama PCM

Canal	Bytes	Observación
IMU	2	Giro X
IMU	2	Giro Y
IMU	2	Giro Z
IMU	2	Aceleración X
IMU	2	Aceleración Y
IMU	2	Aceleración Z
IMU	2	Magnetómetro X
IMU	2	Magnetómetro Y
IMU	2	Magnetómetro Z
IMU	2	Tensión

IMU	2	Temperatura
IMU	2	ADC

Tabla 6. Detalle del canal de IMU

4.2. CONFIGURACIÓN DE LA FPGA

Los puertos físicos de entradas y salidas digitales asignados en la plaqueta 3PX1 para este desarrollo son:

Tabla 7. Pines del modelo asignados a puertos físicos del 3PX1

Nombre de linea	Terminal conector	Terminal de FPGA	Nombre en el modelo	Detalle
GND	P2-1	-		
GND	P2-2	-		
5v	P2-3	-		
3,3v	P2-4	-		
P2_1	P2-5	D3	Clock Out PCM	Salida de "clock" del simulador PCM
P2_2	P2-6	D1	Data PCM Out	Salida de datos del simulador PCM
P2_3	P2-7	C1	Frame Enable Out	Salida de marca por cada trama PCM enviada
P2_4	P2-8	B1	Data and noise	Entrada de datos con ruido al filtro PCM
P2_5	P2-9	F6	Data filtered	Salida del filtro PCM
P2_6	P2-10	F5	Data PCM In	Entrada de datos al adquisidor PCM
P2_7	P2-11	B2	Lock	Enganche de la palabra de sincronismo
P2_8	P2-12	A2	Re-Sync Clock	"Clock" regenerado
P2_9	P2-13	C3	Re-Sync Data	Datos resincronizados
P2_10	P2-14	C2		
P2_11	P2-15	B3		
P2_12	P2-16	A3		
P2_13	P2-17	D5		
P2_14	P2-18	C5		
P2_15	P2-19	B5		
P2_16	P2-20	A5		
P2_17	P2-21	D6		
P2_18	P2-22	C6		
P2_19	P2-23	B6		
P2_20	P2-24	A6		
P2_21	P2-25	C7		
P2_22	P2-26	A7		
P2_23	P2-27	F7		
P2_24	P2-28	E6		
P2_25	P2-29	D8		
P2_26	P2-30	C8		
P2_27	P2-31	B8		
P2_28	P2-32	A8		
P2_29	P2-33	E7		
P2_30	P2-34	E8		
P2_31	P2-35	C9		

P2_32	P2-36	A9		
3,3v	P2-37	-		

4.3. BLOQUES FUNDAMENTALES DEL SYSTEM GENERATOR

El set de bloques del System Generator de Xilinx contiene bloques para construir sistemas digitales y de procesamiento de señales para FPGA utilizando Simulink. A continuación se detallan algunos de los bloques (ver Figura 105) del sistema desarrollado para esta tesis.

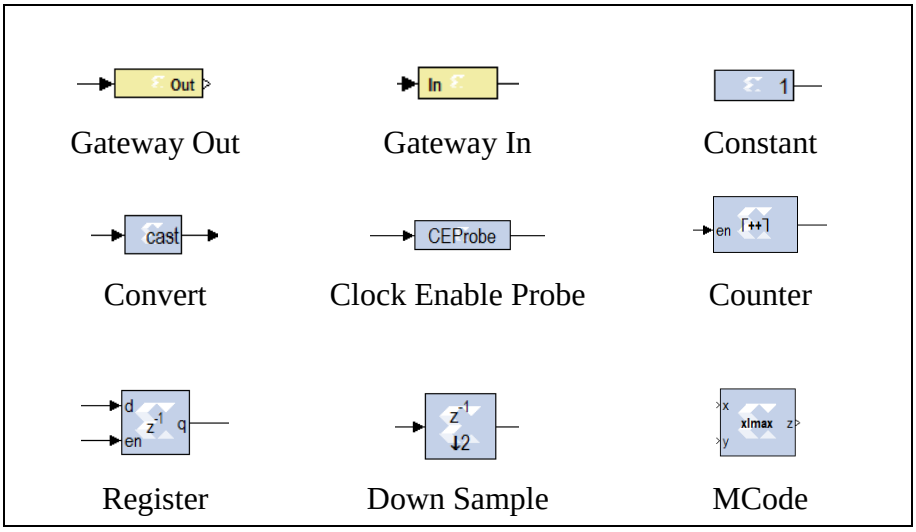


Figura 105. Bloques System Generator de Xilinx para Simulink

Gateway In / Gateway Out:

Los "Gateways" (puertas de entrada o salida) son interfaces de conexión entre elementos del System Generator (que serán que finalmente estén dentro de la FPGA) y los elementos de Simulink que serán elementos externos al desarrollo y servirán para visualizar señales, excitar entradas, etc.

Las funciones específicas de los gateways son:

- Convertir tipos datos de punto flotante o punto fijo de System Generator a tipo de datos de Simulink como ser: enteros, dobles o de punto fijo.
- Definir los puertos de entradas y salidas (en inglés I/O) en el nivel superior del desarrollo (top-level) generados por el System Generator. Estos serán puertos físicos cuando el desarrollo esté finalmente implementado en la FPGA.
- Definir el banco de pruebas (en inglés test-bench) cuando la opción esté seleccionada. En caso afirmativo, durante la generación de código HDL, los valores resultantes de las entradas y salidas del bloque serán almacenados en un archivo de datos.
- Etiquetar las entradas y salidas de la entidad superior definida en el HDL.

Constant:

El bloque de Xilinx “Constant” genera una constante que puede ser un valor de punto fijo, punto flotante o un valor booleano entre otras. El bloque puede ser configurado con el número de bits, el tipo de dato, la posición del punto binario, la precisión (simple, doble o personalizado), etc.

Convert:

El bloque “Convert” convierte el valor de su entrada en un número de un tipo aritmético deseado. Por ejemplo, un número se puede convertir a un valor signado (en complemento a dos) o sin signo. También se puede configurar la acción deseada si se produce un desbordamiento en el valor a convertir.

Clock Enable Probe:

El Clock Enable Probe (CE Probe) proporciona un mecanismo que extrae las señales de habilitación de “clock” de los modelos de System Generator.

El CE Probe acepta cualquier tipo de señal Xilinx como entrada, y produce una señal de salida del tipo booleano. La salida del bloque es un pulso de “clock” que imita el comportamiento de una señal de habilitación de “clock” ideal, que es utilizada en la implementación de hardware de un circuito multirate (es decir, múltiples velocidades de “clock” en un mismo sistema). La frecuencia del pulso a proporcional al período de muestreo de la señal de entrada.

Counter:

El bloque “Counter” de Xilinx implementa un contador con conteo limitado o libre, ascendente o descendente. La salida del mismo puede ser configurada como un número signado o sin signo.

Un contador limitado en cuenta es implementado combinando un contador de cuenta libre con un comparador. Los contadores con límite de cuenta tienen una salida de una precisión máxima de 64 bits. Es posible configurar el “paso” entre la cuenta inicial y el valor final.

Según especificaciones, en el caso de un contador ascendente con cuenta limitada, la salida se calcula de la siguiente manera:

$$out(n) = \begin{cases} InitialValue & if\ n = 0\ or\ out(n-1) = CountLimit \\ (out(n-1) + Step) mod 2^N & otherwise \end{cases} \quad (3)$$

Donde N denota la cantidad de bits en el contador.

Register:

El bloque “Register” modela un registro basado en un flip-flop tipo D, con una latencia de 1 período de muestreo. La salida inicial se puede especificar en el cuadro de díalogo de configuración del bloque. El bloque posee un puerto de entrada de datos y opcionalmente una entrada de “reset” y una entrada de habilitación.

Down Sample:

El bloque “Down Sample” reduce la tasa de muestreo en el lugar del diseño donde se coloca el bloque. El valor muestreado se presenta en el puerto de salida y se mantiene constante hasta que se toma la siguiente muestra.

MCode:

El bloque “MCode” de Xilinx es un contendor para poder ejecutar funciones escritas en código de MATLAB. Este bloque ejecuta el código MATLAB (M-code) para calcular los datos de salida durante una simulación en Simulink. Este mismo código se traduce directamente en VHDL o Verilog cuando se genera la síntesis de hardware para que tenga el mismo comportamiento.

Las interfaces de entrada y salida de este bloque están en concordancia con los parámetros de la función de MATLAB. En otras palabras, hay un puerto de entrada por cada parámetro de la función, y hay un puerto de salida por cada valor que devuelve la misma, con el mismo orden y el mismo orden en que fueron declaradas.

El “MCode” soporta un set limitado de funciones y parámetros de MATLAB y es ampliamente utilizado para la implementación de funciones aritméticas, máquinas de estado finitas y lógica de control.

Se deben seguir las 3 siguientes directivas para la escritura de una función en M-Code:

- Todas las entradas y salidas deben ser del tipo punto fijo de Xilinx.
- Debe tener una salida como mínimo.
- El código para el bloque debe estar en el mismo directorio que el archivo del modelo.

Las declaraciones y operadores soportados por el bloque “MCode” son:

- Declaraciones de asignación.
- Declaraciones if/else/eleif.
- Instrucciones de switch/case.
- Sumas/Restas/Multiplicaciones/Divisiones por potencia de dos

- Operaciones de relación:

<	Menor a
<=	Menor o igual a
>	Mayor a
>=	Mayor o igual a
==	Igual a
~=	No igual a

- Operaciones lógicas:

&	and
|	or
~	not

Las funciones soportadas por el bloque “MCode” son:

- Conversiones de tipo de datos: Soporta una única función que convierte al tipo de datos de punto fijo de Xilinx es la denominada *xfix()*.
- Funciones que devuelven una propiedad de un tipo de datos *xfix*:

xl_nbits()	and
xl_binpt()	or
xl_arith()	not

- Funciones de operaciones lógicas a nivel de bit:

xl_and()	and
xl_or()	or
xl_xor()	xor
xl_not()	not

- Operaciones de desplazamiento: *xl_lsh()* y *xl_rsh()*.
- Función de corte de datos: *xl_slice()*
- Función para concatenar datos: *xl_concat()*
- Función para reinterpretar datos: *xl_force()*
- Función de estados internos: *xl_state()*
- Funciones propias de MATLAB:

disp()	Visualizar el valor de la variable
error()	Muestra el mensaje y aborta la función
isnan()	Prueba si un número es NaN
NaN()	Devuelve un No-Número (Not-A-Number)
num2str()	Convierte un número a cadena de caracteres (string)
ones(1,N)	Retorna un vector de unos de 1xN

pi()	Devuelve el valor de pi
zeros(1,N)	Retorna un vector de ceros de 1xN

4.4. CÓDIGO FRAME GENERATOR

Esta función tiene como parámetros de entrada y sus salidas:

- addr: la dirección del vector, es decir la posición de la trama del dato de interés.
- enable: un pulso de habilitación cuando se quiere hacer efectivo el cambio del dato a la salida.
- dout: es del tipo xfix8,0 (dato de punto fijo no signado de 8bits enteros y 0 bits fraccionarios, es decir un byte) con el dato de salida.
- c: es del tipo xfix16,0 (dato de punto fijo no signado de 8bits enteros y 0 bits fraccionarios, es decir un byte) con el dato del contador de la trama. Es opcional al diseño, se utiliza para fines de depuración del código.

Dentro del código tenemos las siguientes características configurables:

- len: tamaño del vector (largo de la trama).
- header_H y header_L: Dos bytes que representan la palabra de sincronismo en decimal (235_{10}=EB_{16} y 144_{10}=90_{16}), también es configurable.
- vector: son los datos de la carga útil, en este caso a modo de realizar una depuración del código de manera más sencilla, se adoptó un vector de números enteros consecutivos.

```
function [dout,c] = frame_gen(addr,enable)

  len=64;

  header_H = 235;
  header_L = 144;

  vector = 1:len;

  proto8b = {xlUnsigned,8,0};
  proto16b = {xlUnsigned,16,0};

  persistent mem, mem = xl_state(vector, proto8b,len);
  persistent count, count = xl_state(0, proto16b);
  persistent s, s = xl_state(0, proto8b);
  persistent xor, xor = xl_state(0, proto8b);

  if enable
      switch addr

          case 0
              s=header_H;
              count=count+1;
              xor=0;

          case 1
              s=header_L;

          case 2
              s=xl_slice(count,15,8);
              xor=s;

          case 3
              s=xl_slice(count,7,0);
              xor=xl_xor(xor,s);

          case len-1
              s=xor;

          otherwise
              s=mem(addr);
              xor=xl_xor(xor,s);

      end;
 end;

 dout = s;
 c=count;
```