# INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE INGENIERÍA MECÁNICA Y ELÉCTRICA

SECCIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

## EVALUACIÓN E IMPLEMENTACIÓN DE ALGORITMOS DE ADQUISICIÓN DE UNA SEÑAL GPS, EN UN DISPOSITIVO RECONFIGURABLE FPGA

TESIS
QUE PARA OBTENER EL GRADO DE MAESTRO EN CIENCIAS EN INGENIERÍA EN TELECOMUNICACIONES

PRESENTA
IVÁN JESÚS SÁNCHEZ CUAPIO

DIRECTOR DE TESIS:

M. EN C. MIGUEL SÁNCHEZ MERÁZ

*MÉXICO D.F.* *JUNIO 2009*

# INSTITUTO POLITÉCNICO NACIONAL

## SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

SIP-14

*ACTA DE REVISIÓN DE TESIS*

En la Ciudad de México, D. F. siendo las **16:00** horas del día **26** del mes de **JUNIO** del **2009** se reunieron los miembros de la Comisión Revisora de Tesis designada por el Colegio de Profesores de Estudios de Posgrado e Investigación de E.S.I.M.E. para examinar la tesis titulada:

**"EVALUACIÓN E IMPLEMENTACIÓN DE ALGORITMOS DE ADQUISICIÓN DE UNA SEÑAL GPS, EN UN DISPOSITIVO RECONFIGURABLE FPGA"**

Presentada por el alumno:

| **SÁNCHEZ** | **CUAPIO** | **IVÁN JESÚS** |
|---|---|---|
| Apellido paterno | Apellido materno | Nombre(s) |

Con registro:

| A | 0 | 7 | 0 | 2 | 9 | 1 |
|---|---|---|---|---|---|---|

aspirante de:

**MAESTRO EN CIENCIAS EN INGENIERÍA DE TELECOMUNICACIONES**

Después de intercambiar opiniones los miembros de la Comisión manifestaron ***SU APROBACIÓN DE LA TESIS***, en virtud de que satisface los requisitos señalados por las disposiciones reglamentarias vigentes.

LA COMISIÓN REVISORA

El Director de tesis

**M. EN C. MIGUEL SÁNCHEZ MERAZ**

Presidente

**DR. VLADIMIR KAZAKOV**

Segundo Vocal

**DR. AMADEO JOSÉ ARGÜELLES CRUZ**

Tercer Vocal

**M. EM C. FRANCISCO RUBÉN CASTILLO SORIA**

Secretario

**DR. JORGE ROBERTO SOSA PEDROZA**

Suplente

**M. EN C. MARCO ANTONIO ACEVEDO MOSQUEDA**

El Presidente del Colegio

**DR. JAIME ROBLES GARCÍA**

I. P. N.
SECCIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

# *INSTITUTO POLITÉCNICO NACIONAL*

***SECRETARÍA DE INVESTIGACIÓN Y POSGRADO***

## *CARTA CESIÓN DE DERECHOS*

En la Ciudad de México D.F. el día 26 del mes Junio del año 2009, el (la) que suscribe Iván Jesús Sánchez Cuapio alumno (a) del Programa de Maestría en Ingeniería en Telecomunicaciones con número de registro A070291, adscrito a la sección de estudios y posgrado e Investigación de la E.S.I.M.E. Unidad Zacatenco, manifiesta que es autor (a) intelectual del presente trabajo de Tesis bajo la dirección de M. en C. Miguel Sánchez Meráz cede los derechos del trabajo intitulado Evaluación e Implementación de Algoritmos de Adquisición de una Señal GPS, en un Dispositivo reconfigurable FPGA, al Instituto Politécnico Nacional para su difusión, con fines académicos y de investigación.

Los usuarios de la información no deben reproducir el contenido textual, gráficas o datos del trabajo sin el permiso expreso del autor y/o director del trabajo. Este puede ser obtenido escribiendo a la siguiente dirección isanchezc0604@ipn.mx. Si el permiso se otorga, el usuario deberá dar el agradecimiento correspondiente y citar la fuente del mismo.

Iván Jesús Sánchez Cuapio
Nombre y firma

# ÍNDICE

**Páginas**

# Índice de Figuras

## Capítulo 1

**Figura**



## Capítulo 2

**Figura**



## Capítulo 3

**Figura**

# Capítulo 4

**Figura**

# Capítulo 5

**Figura**



# Índice de Tablas

**Tabla**

# OBJETIVO

Realizar la simulación, implementación sobre un FPGA y evaluación de desempeño de algoritmos de adquisición de señales aplicados a la señal L1 del sistema GPS.

**Objetivos Particulares:**

- Documentar los principales métodos utilizados en la adquisición de señales L1 del sistema GPS.
- Desarrollar programas en MatLab y Lenguaje C para simular los algoritmos de adquisición de señales GPS.
- Implementar sistema para adquisición de señales GPS basado en un dispositivo reconfigurable FPGA.
- Hacer un análisis en desempeño de los algoritmos de adquisición de señales GPS mostrando las características que los distinguen.

# JUSTIFICACIÓN

Debido a la proliferación de sistemas de comunicación inalámbrica y móvil se ha dado gran impulso a la tecnología de radio definido por software (SDR) que busca implementar diferentes estándares de comunicación sobre plataformas comunes de hardware. Elementos básicos para la implementación práctica de este tipo de sistemas, son los dispositivos de electrónica reconfigurable. Haciendo uso de los FPGA se puede actualizar fácilmente el funcionamiento de algún diseño de radio modificando el software de los dispositivos sin alterar su arquitectura de hardware.

El presente trabajo de tesis pretende generar una base de desarrollo para la implementación de arquitecturas de radio definido por software. En particular se enfoca a implementar algoritmos de adquisición de señales GPS sobre un FPGA, aunque esta etapa forma parte de una arquitectura más amplia que conjuntada a las etapas adicionales, permitirá establecer la base para la generación de un receptor GPS.

Se desarrolla un receptor de señales GPS debido a que las mejoras a este sistema, así como los diferentes sistemas de posicionamiento como Galileo operan con señales muy parecidas a las de la señal GPS. Entonces un receptor de señales GPS L1 podría adaptarse fácilmente, mediante modificaciones en el software que ejecuta, para convertirse en un receptor de las nuevas señales de GPS o bien de algún otro sistema de posicionamiento existentes en el mundo.

De forma adicional la implementación de una arquitectura de radio definido por software como la desarrollada en este trabajo permitirá disponer de una plataforma de prueba para investigar el desempeño en hardware para diferentes algoritmos de procesamiento de señales como códigos de control de error o algunas técnicas adaptativas de procesamiento de señales.

# RESUMEN

En este trabajo se presenta el desarrollo de módulos de software y hardware para la adquisición de señales GPS. En primer lugar se estudian las características del sistema de posicionamiento global GPS, realizando una descripción de los componentes que integran a este sistema. Se analiza a detalle la estructura básica de un receptor para éste sistema de posicionamiento, donde se describe y estudia a la señal procedente de los satélites que intervienen en su funcionamiento. Así mismo se aborda el modo en que se realiza la adquisición de las señales, inicialmente de manera general como señales de espectro expandido para posteriormente implementar y evaluar algoritmos que realizan esta tarea. Se hace una descripción de los sistemas embebidos y sus particularidades, debido a que el sistema desarrollado se encuentra bajo ese tipo de arquitectura, esencialmente para su evaluación sobre una plataforma de dispositivos lógicos programables FPGA; para lo que es necesario mostrar sus características y modo de operación en la implementación del receptor de GPS. El análisis se efectúa sobre diversos ambientes de trabajo, la etapa de implementación en los programas MatLab y su adaptación en Lenguaje C, conforman la sección de simulación; para concluir con una evaluación en un ambiente de tiempo real, empleando las herramientas que ofrece el software ISE de Xilinx. Finalmente se presentan los resultados del trabajo realizado, identificando en detalle las características de desempeño de los algoritmos implementados y analizados.

# ABSTRACT

This work presents the development of software and hardware modules to realize the acquisition of GPS signals. First the characteristics of the Global Positioning System GPS are presented, giving a brief description of their components. The basic structure of a receiver for this type of system is analyzed in detail, where the signal coming from the satellites that take part in the operation of the GPS is described and studied. Also is revised the way in which the acquisition of GPS signals, initially in a general way like signals of spread spectrum and later implementing and evaluating algorithms that realize this task. Also it is presented a general description of embedded systems because this thesis work is developed under that type of architecture, essentially for its evaluation on a FPGA platform. They are showed this characteristics and way of operation in the implementation of the GPS receiver. The analysis takes place on different work environments. The stage of implementation in the MatLab and its adaptation in C Language conform the simulation section. The test of the developed system concludes with an evaluation in a real time environment. Finally the results of this work are showed, identifying in detail the characteristics of performance of the implemented and analyzed algorithms.

# INTRODUCCIÓN

Los receptores del Global Position System (GPS) o Sistema de Posicionamiento Global (siendo su nombre completo NAVSTAR-GPS) han estado disponibles en el mercado por más de 20 años, sin embargo como sucede con otros tipos de radio receptores, estaban basados fundamentalmente en una electrónica de tipo analógica, resultando en receptores voluminosos y con mucho consumo de potencia. Con el lanzamiento de la fase II de GPS[1] todo el procesamiento de señales analógicas fue remplazado por microprocesadores y algunos circuitos integrados adicionales. En la actualidad los receptores GPS están basados en el uso de circuitos integrados de aplicación especifica o ASIC por su acrónimo en inglés, para el procesamiento de señales y microprocesadores rápidos para ejecutar cálculos[2]. Los ASIC resultan ser muy efectivos para el procesamiento de señales, sin embargo resulta muy costoso el rediseño de sistemas basados en este tipo de dispositivos ya que no pueden ser reprogramados.

El Radio Definido por Software (SDR, por sus siglas en inglés) es un concepto que está revolucionando la tecnología de las comunicaciones, incorporándose y fortaleciendo a las tendencias para la integración de múltiples arquitecturas de redes de telecomunicaciones inalámbricas. El término "Software Radio" fue utilizado por Joe Mitola en 1991[4][5] para referirse a la clase de radios reprogramables o reconfigurables. En una misma pieza de hardware se pueden realizar diferentes funciones en diferente tiempo, obteniendo un dispositivo hardware de propósito general en un ámbito de comunicaciones. Este concepto abre un amplio panorama de oportunidades para la industria de las comunicaciones y la investigación.

Un SDR, es un equipo que se caracteriza por su flexibilidad, la cual permite que pueda diversificar su funcionalidad con sólo cambiar o modificar su programa [6]. Es decir, el mismo equipo de radio puede ser configurado para operar en diferentes modos o sistemas con solo modificar su programación. Mediante circuitos electrónicos especializados, una señal de radio es digitalizada y estos datos son procesados por un circuito electrónico denominado Procesador Digital de Señales, cuya abreviatura en inglés es DSP. El DSP realiza miles de millones de cálculos por segundo con los datos numéricos, de acuerdo a un programa previamente cargado en el dispositivo, y de esta manera extrae la información deseada de la señal de radio [7].

Un punto importante a destacar, es que se puede extraer información de varias señales que se encuentren contenidas en un ancho de banda que está siendo recibido, permitiendo la recepción simultánea en varias modalidades, después esta información puede ser convertida en una señal digital para extraer la información y lograr desarrollar una aplicación.

La reconfiguración del hardware a través de software exige el intercambio de información. Entonces, es importante contar con esquemas que permitan la reconfigurabilidad de los protocolos empleados en las comunicaciones actuales y futuras para asegurar su intercomunicación [8].

Desde la creación y evolución constante de los estándares en comunicaciones, existe una amplia incompatibilidad entre tecnologías de redes inalámbricas utilizadas por diferentes países [9].

Desde una perspectiva comercial y global, este problema inhibe el uso de servicios de roaming y otras facilidades de las comunicaciones globales. Esta es una de las grandes oportunidades de SDR, ya que el aspecto económico es de suma importancia, para la integración transparente de diversas arquitecturas de comunicación. El desarrollo tecnológico en materia de telecomunicaciones se puede percibir a cada instante, cuando usamos el teléfono celular, navegamos por Internet, el conocer nuestra posición en cualquier parte del mundo a través de un receptor GPS, cuando vemos un programa vía cable o tenemos una tarifa reducida de larga distancia gracias a la telefonía IP.

Entonces un receptor de señales GPS basado en software ofrecerá la posibilidad de aplicar rediseños de forma directa y con costos bajos [10]. Este receptor, al contrario de un receptor ordinario, realiza todas sus operaciones en un microprocesador programable, haciendo que el sistema sea altamente flexible, tal que permitirá verificar la efectividad de diferentes algoritmos y ajustar la operación del receptor en el caso de que sucedan cambios en la señal GPS que se recibe.

Los FPGA (**Field Programmable Gate Array)** resultan ser una alternativa muy atractiva para el tipo de procesamiento requerido por los radio receptores definidos por software, ya que permiten obtener velocidades en hardware adecuadas para determinadas aplicaciones, sin perder de vista la flexibilidad en software. La posibilidad de reutilización de su lógica reconfigurable disminuye su costo, permitiendo utilizar el mismo hardware para varias aplicaciones cambiando únicamente su programación, esto resultará de mucho valor ya que estarán disponibles nuevas señales GPS (L2 y L5), así como Galileo, el sistema de navegación satelital de Europa, que será compatible con GPS, además sumadas las señales de posicionamiento ya existentes en todo el mundo[11].

Estas nuevas señales y constelaciones principales de satélites buscan una convergencia final al Sistema Global de Navegación por Satélite (GNSS, su acrónimo en inglés). La utilización de señales más robustas y las diversas frecuencias planeadas para el GPS, el GLONASS y Galileo serán usadas efectivamente para el desarrollo de este sistema global [12].

En este trabajo se aborda el desarrollo de un modulo para la adquisición de señales GPS. Este modulo forma parte de una arquitectura SDR para un receptor GPS. El elemento básico de electrónica reconfigurable para la arquitectura propuesta es un FPGA de la familia Virtex.II de Xilinx.

**Contenido de la Tesis**

Capítulo 1. Se muestran las características del sistema GPS, identificando sus componentes y realizando una breve descripción de ellos, se analiza la estructura básica de un receptor para éste sistema de posicionamiento y finalmente se describe y analiza la señal procedente de los satélites que intervienen en el funcionamiento del GPS.

Capítulo 2. En este capítulo se describe el modo de adquisición de las señales GPS, inicialmente como señales de espectro expandido, para finalmente analizar, identificar e iniciar el análisis de los algoritmos empleados para realizar esta tarea.

Capítulo3. Aquí se hace una descripción de los sistemas embebidos y su arquitectura, debido a que el sistema desarrollado contiene características de este tipo. Principalmente se estudia a los dispositivos lógicos programables FPGA, mostrando sus características, funciones, herramientas, modo de operación y ventajas que son utilizadas para la implementación del receptor.

Capítulo 4. En esta sección se presenta el diseño de la plataforma de trabajo basada en el uso de un FPGA de Xilinx y el desarrollo de los programas de simulación en MatLab y Lenguaje C, para los algoritmos de adquisición de señales GPS

Capítulo 5. Se presentan los resultados de los capítulos previos. Los algoritmos son sometidos a evaluaciones en diferentes ambientes de trabajo y simulación, logrando observar las características y cualidades de cada uno de los métodos empleados.

Finalmente se presentan las conclusiones y observaciones del trabajo desarrollado.

# Capítulo 1

# ESTRUCTURA BASICA DEL RECEPTOR

En el primer capítulo se presentan las generalidades de un receptor de señales GPS, comenzando por describir al sistema de posicionamiento global y las etapas que lo conforman, además de analizar los elementos que forman parte del dispositivo capaz de recibir y extraer la información derivada de su uso, distinguiendo las etapas de procesamiento que sigue la señal, para terminar con el análisis de ésta.

## 1.1 Sistema GPS

El Global Position System (GPS) permite determinar en todo el mundo la posición de un objeto, una persona, un vehículo o una nave, con precisión de metros, siendo en el mejor de los casos hasta centímetros, con mejoras al sistema.

Aunque su invención se le atribuye a los gobiernos francés y belga, el sistema fue desarrollado, instalado, y actualmente operado por el Departamento de Defensa de los Estados Unidos [13].

Cuando se desea determinar la posición, el aparato que se utiliza para ello, localiza automáticamente como mínimo cuatro satélites de la red, de los que recibe una señal indicando la posición de cada uno de ellos. Por triangulación se calcula la posición en que éste se encuentra, esta tarea se basa en determinar la distancia de cada satélite respecto al punto de recepción.

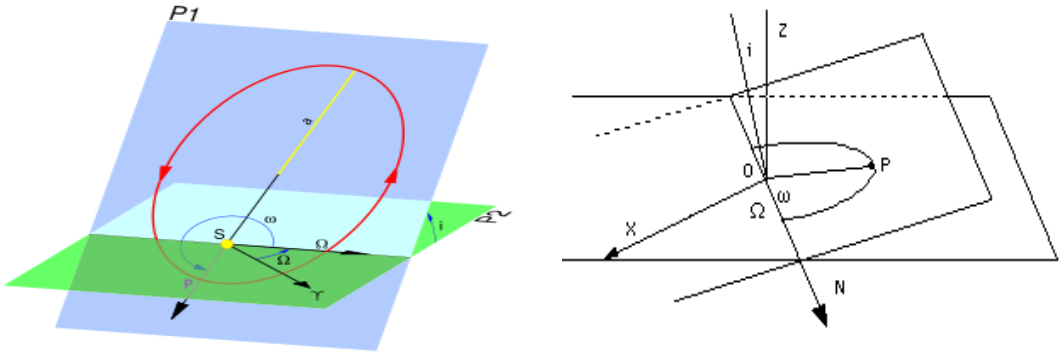



*Fig. 1.1 Planos Orbitales con respecto a la tierra.*

Las localizaciones de los planos orbitales de los satélites, con respecto a la Tierra, son definidas por la longitud de su nodo ascendente, que es el punto de intersección de cada plano orbital con el plano ecuatorial, como es mostrado en la figura 1.1. El meridiano de Greenwich es el punto de referencia en el cual la longitud del nodo ascendente *(Ω)* tiene valor cero.

La estructura del sistema de posicionamiento se divide por segmentos, cada uno de ellos tiene un ámbito de acción, y realizan funciones distintas.

### 1.1.1 Segmento de Control

El segmento de control consiste en un sistema de seguimiento por estaciones base que están distribuidas por todo el mundo, realizando funciones de seguimiento y monitoreo de la operación de la constelación. Las estaciones también se encargan de auto regular las órbitas de los satélites y sincronizar las variables temporales de cada satélite [14].

El segmento de control comprende 5 bases militares Falcon de la US Air Force localizadas en:

- Colorado Springs
- Hawaii (Pacifico Oriental)
- Isla Ascensión (Atlántico)
- Diego García (Indico)
- Kwajalein (Pacifico Occidental)

Tres de estas cinco estaciones constituyen los ojos y oídos del GPS, siguiendo a los satélites cuando pasan sobre ellas y midiendo su distancia cada 1.5 segundos. Estos datos son procesados junto con información meteorológica e ionosférica y enviados a la Estación Central de Control de la US Air en Colorado Springs. Es ahí donde se determinan las órbitas y se estiman los resultados del reloj comprobando el funcionamiento correcto de los satélites y si es necesario corregir su posición. Toda esta información se envía a las estaciones de enlace ascendente, localizadas en las estaciones de seguimiento de isla Ascensión, Diego García y Kwajalein; desde las cuales se transmite la información a los satélites. Esta distribución de las estaciones por todo el mundo, consigue que los satélites GPS sean seguidos el 92% del tiempo.

### 1.1.2 Segmento Espacial

El segmento espacial comprende una red de satélites en órbitas circulares con un periodo de 12 horas siderales. La constelación original era de 24 satélites (21 satélites de navegación y 3 satélites de control) distribuidas en 3 órbitas planas inclinadas sobre el plano ecuatorial. Actualmente se ha cambiado su disposición ocupando 6 planos orbitales con 4 satélites por plano. Los satélites NAVSTAR GPS están a una altitud de 20200 Km[15] con una inclinación de 55 grados, visibles sobre el horizonte por 5 horas aproximadamente. Esta constelación de satélites proporciona una cobertura global de 24 horas diarias, permitiendo que el usuario pueda disponer de un mínino de 5 y un máximo de 8 satélites GPS visibles, desde cualquier punto de la tierra.

Existen diferentes categorías de satélites GPS. El primer prototipo Block I fue lanzado en Febrero de 1978 desde la base de Vandenberg. La segunda categoría de satélites GPS Block II [16], fue lanzada en 1985, transmitiendo señales L1+L2 (1575.42 + 1227.60 MHz) y recibiendo en la frecuencia 1783.74 MHz. Los satélites Block IIA tienen una ligera modificación respecto al diseño original, fueron diseñados para operar 180 días sin contactar con el segmento de control, lanzados entre Noviembre de 1990 y Marzo de 1996. Los Block IIR (la R por replenishment-reaprovisionamiento) tienen un tiempo de vida de 10 años y son capaces de realizar comunicaciones de satélite a satélite, fueron lanzados en 1996 para mantener la constelación completa. Fig. 1.2.

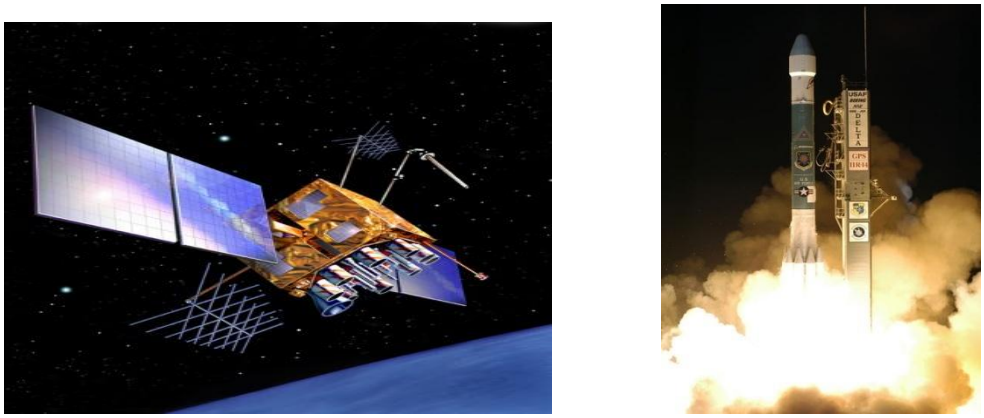


*Figura 1.2 Lanzamiento de satélites para la constelación NAVSTAR-GPS mediante un cohete Delta*

La figura 1.3 muestra una tabla completa de la evolución del sistema GPS através del tiempo [16][17].

| Bloque | Periodo de lanzamiento | Satélites lanzados | Actualmente en servicio |
|---|---|---|---|
| **I** | 1978–1985 | 10+1[1] | 0 |
| **II** | 1985–1990 | 9 | 0 |
| **IIA** | 1990–1997 | 19 | 13 |
| **IIR** | 1997–2004 | 12+1[1] | 12 |
| **IIR-M** | 2005–2009 | 7+1[2] | 6 |
| **IIF** | 2009–2011 | 0+10[2] | 0 |
| **IIIA** | 2014–? | 0+12[3] | 0 |
| **IIIB** | | 0+8[3] | 0 |
| **IIIC** | | 0+16[3] | 0 |
| **Total** | | 59+2[1]+12[2]+36[3] | 31 |

*Fig. 1.3 Tabla que muestra el desarrollo del sistema GPS. [1]Fallado [2]En preparación [3]Planeado.*

### 1.1.3 Segmento de Usuario

El segmento de usuario se refiere a los receptores diseñados para decodificar las señales transmitidas desde los satélites, convirtiendo la señal proveniente en valores de velocidad, posición y tiempo[18]. Estos receptores deben seleccionar uno o más satélites de entre los avistados, capturar las señales GPS, realizar mediciones y seguimientos; para finalmente extraer los datos de navegación.

Existen dos tipos de servicio disponibles para la recepción de señales GPS que pueden ser utilizados.

Los usuarios civiles de todo el mundo pueden usar el servicio de posicionamiento estándar SPS[19] (Standard Positioning Service) sin cargos ni restricciones, contando con cierto grado de precisión debido a que es basado en el uso de una única frecuencia L1. La inmensa mayoría de receptores GPS pueden recibir y utilizar el SPS tomando en cuenta su precisión limitada.

El servicio de posicionamiento preciso PPS (Precise Positioning Service), se logra utilizando receptores GPS en posiciones referenciales para permitir correcciones através de datos enviados para reposicionar su emplazamiento. Su grado de precisión se basa en las medidas obtenidas con el código P, recibido en forma paralela al código C/A. Este código es el denominado Preciso o Protegido que es una larga secuencia binaria pseudoaleatoria.

## 1.2 Estructura Básica de un Receptor GPS

La señal procedente de un satélite llega al receptor a través de una antena, esta señal pasa por un filtro pasa banda que elimina las señales que no interesan. A continuación es introducida en un amplificador de gran calidad y con baja ganancia de ruido, a fin de no arrastrarlo por el resto de las etapas. La señal atraviesa un filtro pasa bajas que elimina armónicos indeseables y a continuación en un mezclador, se mezcla la señal recibida con la de un oscilador local, para obtener un par de frecuencias mas. Mediante otro filtro pasa bajas se queda solo con la frecuencia más baja, repitiendo el proceso se puede reducir la frecuencia hasta el nivel adecuado, para amplificarla y pasarla a la etapa de procesamiento. Un esquema general puede observarse en la figura 1.4[20].

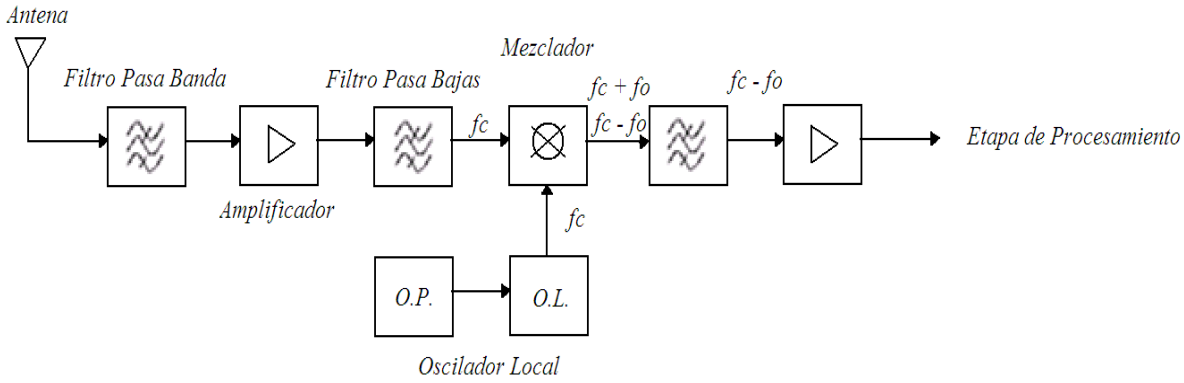



*Fig. 1.4 Etapas para el acondicionamiento de una señal GPS*

### 1.2.1 Antena

Una antena para este tipo de receptores es diseñada particularmente para su uso en GPS. Debe cumplir con los requerimientos determinados para su uso en el sistema. Estos requerimientos están definidos por parámetros como: ganancia y elevación, pérdida por interferencia, pérdida por multitrayectoria y propiedades físicas adecuadas como el tamaño, material, etc.

Esta antena recibe las señales procedentes de los satélites, suele presentar una polarización circular y una cobertura semiesférica. Un valor típico de cobertura son 160º pero los diseños de la antena varían desde bobinas helicoidales hasta líneas de microcinta.

### 1.2.2 Terminal de Entrada de RF

Este es el bloque que prepara a la señal analógica para ser muestreada por el convertidor Analógico Digital A/D; éste consta básicamente de dos funciones, el acondicionamiento de la señal y la conversión a bajas frecuencias. Figura 1.5.

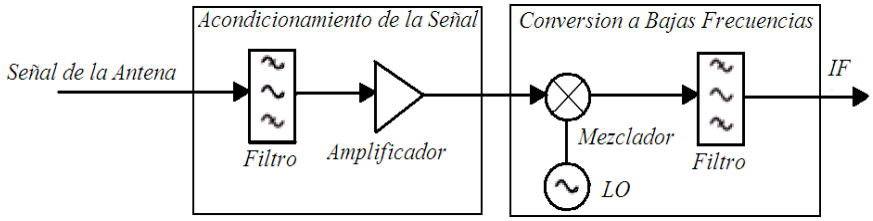



*Fig.1.5 Estructura de la terminal de entrada de RF en un Receptor GPS. Contiene dos bloques: Acondicionamiento de la señal y una conversión de bajada de la frecuencia*

El propósito principal del bloque de acondicionamiento de la señal es remover la interferencia de los componentes de la señal. Para remover la interferencia se hace uso de un filtro pasa banda, este filtro remueve todo tipo de señales y sus componentes fuera del ancho de banda de interés y en el caso de un receptor GPS de una sola frecuencia, tan solo deja pasar a la frecuencia L1 GPS. Fig. 1.6.

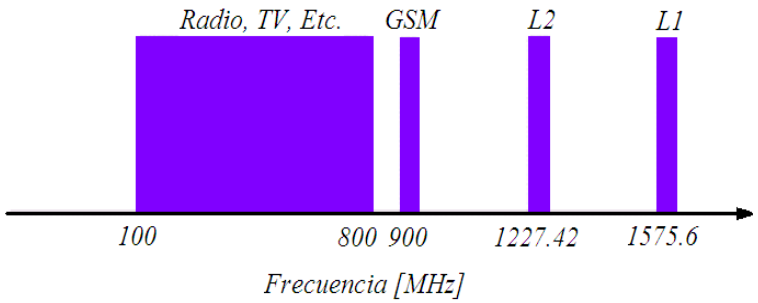



*Fig. 1.6 Las señales GPS entre otras señales de radio en la banda de frecuencias vecinas*

La siguiente función de este bloque de acondicionamiento es amplificar la señal recibida, ya que en el momento que es captada por la antena tiene una amplitud muy baja por la combinación de una baja potencia de trasmisión del satélite y la gran distancia de viaje. La amplificación de la señal mejora el resultado en el resto de etapas del receptor.

### 1.2.3 Convertidor de Bajada

El propósito de esta etapa es hacer una conversión de la señal que viene del bloque de acondicionamiento a una en menor frecuencia. La señal recibida es convertida de RF (1575.42 MHz) a la frecuencia intermedia IF (38.400 KHz). Esta conversión está dada por el trabajo de un mezclador, que mezcla la señal de entrada con la señal de un oscilador local LO. Representada por la ecuación (1.1), donde se muestra la convolución de las señales.

$$s_{out}(t) = s_{in}(t) * s_{LO}(t) \qquad (1.1)$$

$$\downarrow F$$

$$S_{out}(f) = S_{in}(f) \cdot S_{LO}(f) \qquad (1.2)$$

*S(f)* es la transformada de Fourier de *s(t)*, siendo que en el dominio de la frecuencia la función de convolución puede realizarse como una multiplicación.

La frecuencia de LO es seleccionada como:

$$f_{LO} = f_{RF} - f_{IF} \quad (1.3)$$

Para obtener cuatro componentes de la señal identificados como:

$$(-f_{RF} - f_{LO}) \quad (-f_{RF} + f_{LO}) \quad (f_{RF} - f_{LO}) \quad (f_{RF} + f_{LO}) \quad (1.4)$$

Las componentes de bajas frecuencias pasan el filtro y dos son removidas de la señal. Con esto se consigue tener solo componentes en bajas frecuencias que serán utilizadas en la siguiente etapa.

### 1.2.4 Convertidor A/D

El convertidor A/D es responsable de muestrear la señal analógica en baja frecuencia, estableciendo una tasa de muestreo dada por la ecuación 1.5.

$$f_s = 2 \cdot \Delta f \quad (1.5)$$

Donde $\Delta f$ es el ancho de banda y la tasa mínima de muestreo está dada por:

$$f_s = 2 \cdot f_{max} \quad (1.6)$$

$f_{max}$ es la frecuencia máxima de los componentes de la señal.

### 1.2.5 Canales del Receptor

El procesamiento de señales GPS toma lugar en diferentes canales. La señal de cada satélite visible a la antena es alojada sobre un canal, limitado por un máximo número de canales en el receptor. La figura 1.7 (a) muestra la estructura del receptor de GPS y en (b) se observa a detalle a cada uno estos canales.

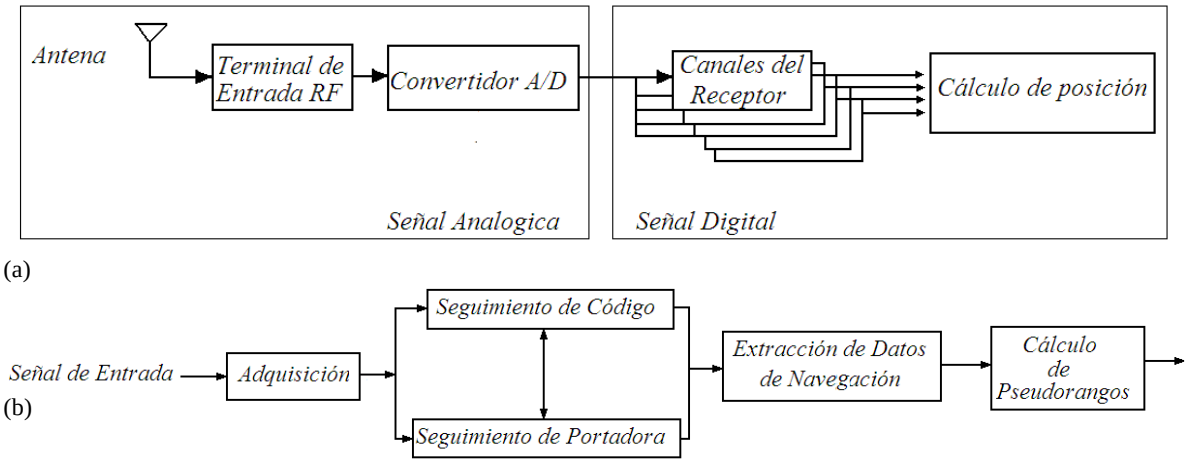



*Fig. 1.7 (a) Estructura Basic a de un Receptor GPS, mostrando sus elementos y etapas principales. (b) La Figura muestra el contenido de un canal del receptor. La Adquisición estima los parámetros de la señal. Estos parámetros son referidos a los dos bloques de seguimiento. Después del seguimiento, los datos de navegación son extraídos y los pseudorangos, son calculados.*

Existen dos métodos comunes para visualizar los satélites desde el receptor. Uno es llamado arranque templado (*warm start)* y el otro arranque frio (*cold start).*

Arranque Templado: En el arranque templado, el receptor combina información de los datos del almanaque y la última posición calculada. Estos datos son usados para calcular el curso y la posición de todos los satélites presentes y por medio de un algoritmo es calculado cuál de los satélites esta actualmente visible. En el caso en que el almanaque no está actualizado no se puede tener un buen cálculo de posiciones, para este tipo de casos el receptor tiene que realizar el trabajo de arranque frio.

Arranque Frio: Para este caso el receptor no tiene la información inicial para la búsqueda de los satélites, el método de búsqueda es referido como adquisición. En ésta, la búsqueda considera todos los posibles satélites en todo momento.

## 1.3 Tareas del Receptor

En base a los conceptos presentados a partir de ahora se referirá al receptor como un procesador de señales. Todos los satélites transmiten su información en dos frecuencias L1 y L2. Las dos frecuencias portadoras son moduladas por códigos PRN y por el mensaje de navegación. Se utilizan dos tipos de códigos PRN: Código P, utilizado por el servicio PPS y Código C/A, empleado en el servicio SPS.

La frecuencia portadora L1 es modulada por los códigos C/A y P, y por los datos de navegación, mientras que la portadora L2 es modulada por el código P y los datos de navegación. Debido a esto existen dos tipos de receptores: Aquellos que demodulan los códigos P y C/A y los que sólo demodulan los códigos C/A. De esta forma los usuarios no autorizados no acceden a la frecuencia L2, ya que no pueden descifrar el código P sólo demodulan el código C/A en la frecuencia L1; en cambio los receptores militares tienen acceso a ambas frecuencias, demodulan el código C/A en la frecuencia L1 y luego el P en ambas frecuencias.

Por su arquitectura podemos distinguir tres tipos principales de receptor GPS:

1. Receptor secuencial.- Este receptor tiene un solo canal que va siguiendo secuencialmente los satélites visibles, extrayendo el retardo del código. La figura 1.8 muestra la estructura de este tipo de receptores.

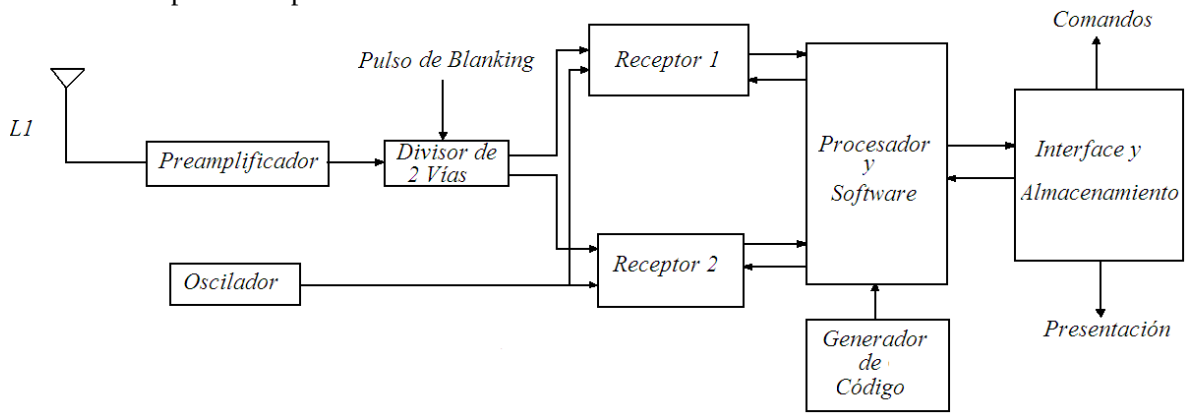



*Fig. 1.8 Receptor GPS de tipo secuencial.*

Este receptor incluye un segundo canal para ayudar a la adquisición de otro satélite, típicamente el receptor debe permanecer con cada satélite al menos un segundo para adquirir la señal y extraer los datos.

2. Receptores con canales multiplexados.- Como en el caso anterior, sólo tienen un canal físico en hardware, sin embargo, disponen de cuatro o más ciclos de seguimiento en software. El receptor debe muestrear todos los satélites asignados a dicho canal en un tiempo igual o inferior a 20 milisegundos, pues es la duración de un bit del mensaje de navegación y de este modo los mensajes son leídos simultáneamente.

3. Receptores continuos.- Este tipo de receptores incorpora cinco o seis canales físicos de seguimiento capaces de monitorear simultáneamente a cuatro satélites, los canales adicionales son asignados para seguir una segunda frecuencia y obtener datos de otro satélite a la vista. Este tipo de receptores permiten la rápida obtención de la posición. En la figura 1.9 podemos ver el esquema de uno de los receptores más complejos de este tipo, capaz de seguir las dos frecuencias de GPS y varios satélites simultáneamente.

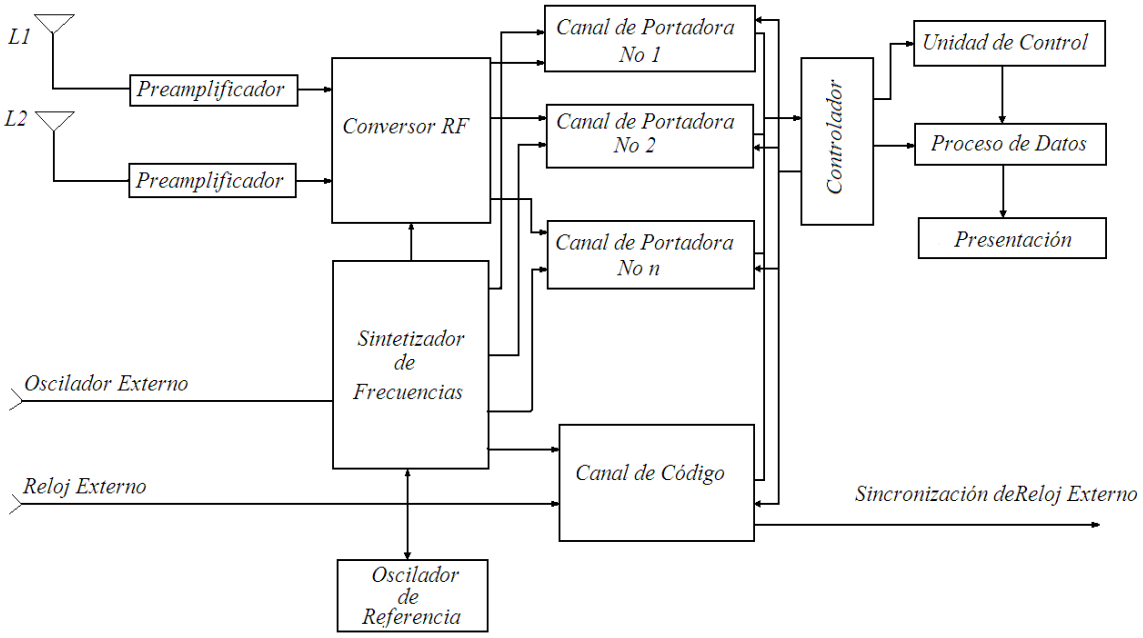



*Fig. 1.9 Receptor Continuo o de varios canales.*

## 1.4 Estructura de la Señal GPS

La descripción oficial de la señal GPS se muestra en GPS INTERFACE CONTROL DOCUMENT (IDC-GPS-200) 1992[21]. Por lo que solo se destacan algunas de sus características.

En estas señales se emplean códigos para las lecturas de los datos en el satélite, ambas caracterizadas por una secuencia de ruido pseudoaleatorio (PRN).

El código-C/A (*Coarse /Acquisition o Clear/Access*) se repite cada 1 ms.

El código-P *(Precision o Protected*) tiene la misma frecuencia que la portadora fundamental y se repite aproximadamente cada 266.4 días. El código-W se usa para encriptar el código-P y obtener así el código-Y, para el caso en que se requiere implementar el método A-S (*Anti-Spoofing*) y generar una restricción de servicio a usuarios no autorizados [22].

La codificación del mensaje de navegación requiere 1500 bits que, a una frecuencia de 50Hz, se transmiten en 30s. Las señales en L1 y L2 están moduladas por el código-P y el código-C/A se envía en cuadratura defasada de 90º en el mismo código-P. Si se denotan las portadoras sin modular como:

$$L_i(t) = a_i \cos(f_i t) \quad (1.7)$$

Y las secuencias del código-P, código C/A, código-W y el mensaje como:P(t), C/A(t),W(t) y D(t) respectivamente, las ecuaciones de las portadoras moduladas estarán representadas:

$$L_i(t) = a_1 P(t) W(t) D(t)\cos(f_1 t) + a_1 C/A(t)D(t)\mathrm{sen}(f_1 t) \quad (1.8)$$

$$L_i(t) = a_2 P(t) W(t) D(t)\cos(f_2 t) \quad (1.9)$$

El análisis de estas funciones determina que en el dominio de la frecuencia, la señal de difusión se caracteriza por un espectro expandido.

Los sistemas de espectro expandido se emplean a menudo en sistemas de comunicación por su robustez frente a interferencias intencionadas. Teniendo como principales características:

1. La señal modulada ocupa mucho más ancho de banda del necesario para poder transmitir la secuencia de información.
2. Para conseguir la expansión del espectro se emplea un código en transmisión que es independiente de la secuencia de información. El mismo código es empleado en el receptor para recuperar la señal deseada con su espectro original y el receptor debe operar de forma síncrona con el transmisor.

En el caso del GPS, el mensaje tiene un ancho de banda de 100Hz, mientras que el código-P tiene un ancho de 20MHz. Debido a que las señales GPS pertenecen a este tipo de espectro, se distinguen tres tipos de modulación para este tipo de sistemas:

1. Secuencia Directa (*Direct Sequence, DS*), consiste en multiplicar la secuencia de datos de entrada por un código de banda ancha para modular posteriormente a una portadora que generalmente cuenta con una modulación PSK.
2. Salto en Frecuencia (*Frequency Hopping, FH*), el espectro de una portadora modulada por la secuencia de información es expandido, cambiando la frecuencia de la portadora de una forma pseudoaleatoria.
3. Salto en tiempo (*Time Hopping, TH*), se divide el tiempo en intervalos llamados tramas, y éstas en ranuras (slots). Para cada trama se selecciona de forma pseudoaleatoria una ranura, y sólo se transmite información durante esa ranura.

La señal de GPS es una señal de espectro expandido de secuencia directa, con modulación de fase BPSK (Binary Phase Shift Keying) (Fig. 1.10) y las señales se transmiten por radio, concretamente en el espectro perteneciente a las microondas, constituida de dos componentes: Una de 1575,42 MHz, que es portadora del mensaje de navegación así como los códigos SPS y otra de 1227,6 MHz, que se utiliza para medir el retardo ionosférico en los receptores PPS.

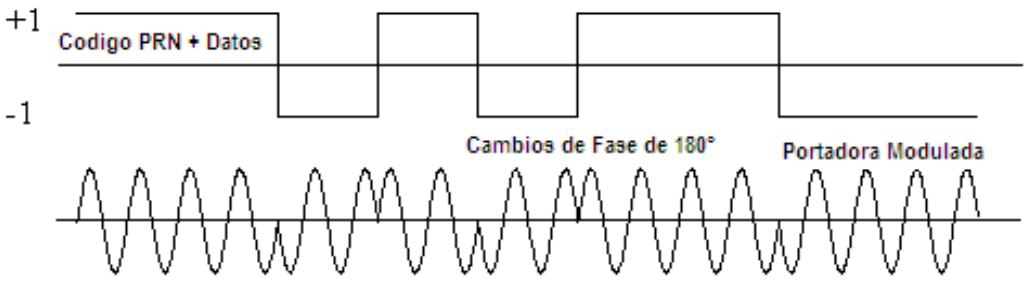



*Fig. 1.10 Modulación BPSK, utilizada en las señales GPS*

Si la señal L1 es modulada por dos señales pseudoaleatorias, la señal L2 puede estar modulada por el código P o por el código C/A, y puede o no contener modulación de datos. En este análisis solo se considera la señal L1 con modulación de código C/A.

La señal recibida por un receptor, sin tener en cuenta el ruido, tiene la siguiente representación:

$$s(t) = A\,C(t)\,D(t)\cos(2\pi(f_p + f_D)\,t + \phi_p) \qquad (1.10)$$

donde $A$ es la amplitud de la señal, $C(t)$ es la señal de modulación de código pseudoaleatorio $C/A$, $D(t)$ es la señal de modulación de datos, $f_p$ es la frecuencia portadora, $f_D$ es el desplazamiento de frecuencia debido principalmente al efecto Doppler, y $\phi_p$ es la fase inicial de la señal portadora.

Las señales moduladoras de datos, y de código son de la forma:

$$C(t) = \sum_{n=-\infty}^{\infty} c_n\,p(t - nT_c) \qquad (1.11)$$

$$D(t) = \sum_{n=-\infty}^{\infty} d_n\,p(t - nT_b) \qquad (1.12)$$

donde $p(t)$ es un pulso rectangular de amplitud unitaria y duración $T_c$ o $T_b$, $c_n$ y $d_n$ son secuencias binarias que pueden tomar valores de 1. Particularmente, $c_n$ es la secuencia pseudoaleatoria que le da las propiedades de espectro expandido a la señal $s(t)$. Cada pulso rectangular del código es llamado chip y el intervalo de tiempo $T_c$ es denominado tiempo de chip.

Una trama de datos consiste en 1500 bits divididos en cinco subtramas de 300 bits. Una trama se transmite cada 30 segundos. De aquí tres subtramas de seis segundos contienen datos de la órbita y del reloj. Las correcciones del reloj se envían en la primera subtrama y los valores de las orbitas son enviados en las subtramas dos y tres. Las cuatro y cinco se utilizan para transmitir distintas páginas de datos del sistema. Y después, un conjunto de 25 tramas (125 subtramas) completan el mensaje de navegación que es enviado sobre un periodo de 12.5 minutos.

### 1.4.1 Descripción de la Señal de Código

Las señales C/A de GPS pertenecen a la familia de los códigos de ruido pseudoaleatorio (Pseudorandom Noise - PRN) conocidos como códigos Gold[23]. El código correspondiente a cada satélite tiene una duración de 1 ms y una frecuencia de chip de 1,023 MHz. Cada código C/A es un código Gold de 1023 chips, el cual es el resultado de la suma módulo-2 de otras dos secuencias G1 y G2i. Donde G2i es una versión de G2 retrasada un número entero i de chips que va de 5 a 950.

Las secuencias G1 y G2 son generadas con registros de corrimiento de 10 etapas como se observa en la Figura 1.11, Donde además se muestra una manera de obtener la versión retrasada de G2 sumando en módulo-2 la salida de dos de las etapas del registro correspondiente.

Esta implementación sólo es capaz de generar 45 códigos, de los cuales 9 son desbalanceados y los 36 restantes están reservados para GPS.

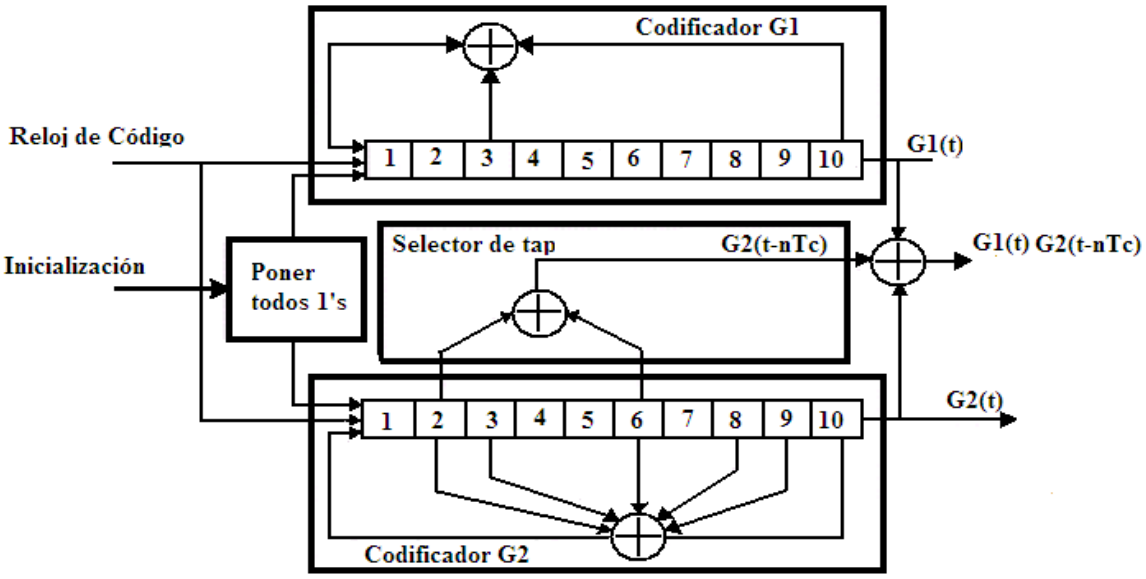



*Fig. 1.11 Generador de Códigos Pseudoaleatorios (PRN) con registros de corrimiento*

En la figura 1.12 se muestra la función de auto correlación correspondiente a un código C/A.

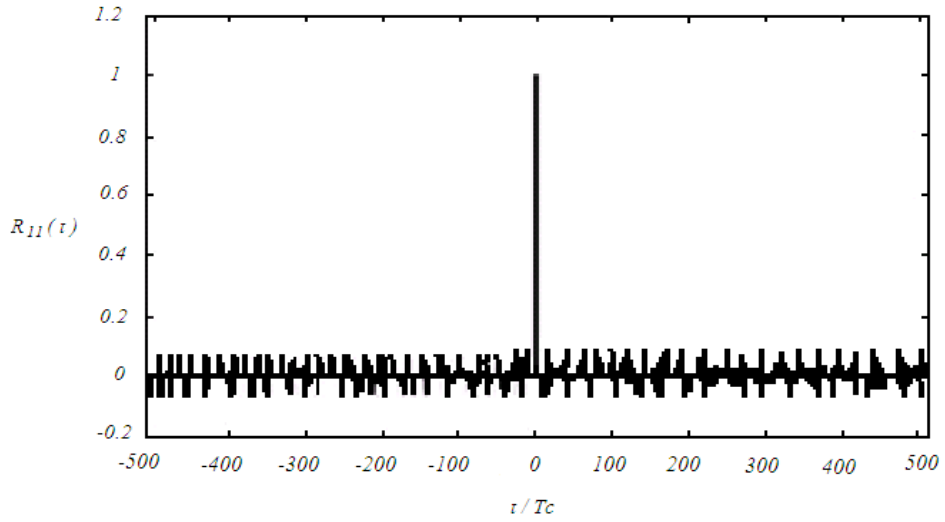



*Fig. 1.12 Auto correlación del Código C/A correspondiente al satélite 1*

El gráfico 1.12 responde a la ecuación:

$$R_{ij}(\tau) = \frac{1}{1023\,T_c}\int_{-511T_c}^{511T_c} c_i(t)c_j\,(t+\tau)\,dt \quad (1.13)$$

donde $C_i$ y $C_j$ son los códigos C/A correspondientes a los satélites *i y j*, respectivamente.

### 1.4.2 Codificación PRN

La idea básica es que la correlación entre un código PRN y cualquier otro código, incluyendo versiones diferidas de él mismo, sea mínima y que la correlación solo sea máxima cuando se compare consigo mismo. Esto permite enviar múltiples señales en una misma frecuencia, mediante el procedimiento de modular cada señal con un código PRN diferente. Las señales se bloquean entre sí, pero al recibir se usa un código PRN concreto y debido a esto la señal correcta se ve realzada y el resto de señales desaparecen. Esta es la operación básica desarrollada en el GPS.

Un buen código PRN debe cumplir con ciertas propiedades matemáticas y algorítmicas como: correlación cruzada mínima o facilidad de generación con una representación compacta. Algo a destacar es que cuanto más largo sea el código es mejor, aunque esto implica tiempos de alineamiento más elevados, ya que el código PRN generado por el receptor debe sincronizarse exactamente con el código de la señal que llega.

El nombre de ruido pseudoaleatorio proviene de la asimilación de que la señal parece ruido y lo de pseudo es porque no es realmente aleatorio, sino que se obtiene de una fórmula.

La generación de secuencias PRN se basa en el empleo de registros de corrimiento realimentados linealmente. Estos registros se componen de n biestables en serie, de forma que la secuencia que generan es periódica con un periodo de valor $2^{n-1}$. Los bits que componen los códigos PRN se denominan chips para remarcar que esos códigos no aportan información.

Con cada pulso de reloj se produce un desplazamiento a la derecha del registro, siendo el bit que está más a la derecha el que aparece a la salida. El nuevo valor del bit de la izquierda se determina con la suma binaria de dos celdas definidas. La elección de las celdas es arbitraria y determina la propiedad del código resultante, este tipo de mecanismo es mostrado en la figura 1.11.

## 1.5 Receptor GPS basado en una Plataforma de Radio Definido por Software.

Las técnicas de radio definido por software (SDR) están fundamentalmente ligadas a la posibilidad de ser desarrollados por circuitos de altísima integración, aprovechando las herramientas necesarias como las arquitecturas, lenguajes, librerías, protocolos, etc; para su programación rápida y eficiente[24].

Esta tecnología permitirá superar las limitaciones resultantes de la incompleta y parcial estandarización internacional actual, que está resultando en una multiplicación de estándares incompatibles. Aunque se ha comenzado a explorar una solución con el nombre genérico de Radio Cognitivo[25] y es visto desde un enfoque novedoso para mejorar la utilización de un valioso recurso natural: el espectro electromagnético. Un radio definido por software es definido como un sistema inteligente de comunicación inalámbrica que es consciente de su entorno y utiliza la metodología de aprender de él para adaptarse a las variaciones de la entrada[26]. Esta característica le permite que se auto configure para adaptarse a las condiciones requeridas por la red a la que decida conectarse.

Un SDR, puede diversificar su funcionalidad con sólo cambiar o modificar su programa; es decir, el mismo equipo de radio puede ser configurado para operar en cualquier modo actualmente en uso por la radio afición, con solo modificar el aspecto software. Entonces, es importante contar con esquemas que permitan la reconfigurabilidad de los protocolos empleados en las comunicaciones actuales y futuras para poder lograr su intercomunicación.

El receptor GPS diseñado bajo una plataforma basada en software y sobre un dispositivo FPGA, ofrece la capacidad de lograr rediseños de una forma directa. Este receptor, al contrario de un receptor ordinario, realiza todas sus operaciones en un microprocesador programable, haciendo que el sistema sea altamente flexible y permita verificar la efectividad de diferentes algoritmos ajustando la operación del receptor para el caso en que sucedan cambios en la señal GPS.

Esta característica resultará de mucho valor ya que estarán disponibles nuevas señales GPS (L2 y L5), así como las señales Galileo en el sistema de navegación satelital de Europa y que será compatible con GPS. Estas nuevas señales y constelaciones principales de satélites buscan una convergencia final al Sistema Global de Navegación por Satélite GNSS. La reutilización de señales más fuertes y las frecuencias diversas planeadas para el GPS, el GLONASS y Galileo serán usadas efectivamente para el desarrollo de este sistema global de posicionamiento.

Así que el trabajo de tesis presentado se orienta al desarrollo de un receptor GPS basado en electrónica programable, estableciendo las características de radio definido por software.

# Capítulo 2

# ALGORITMOS DE ADQUISICIÓN DE SEÑALES GPS

En la sección anterior se vieron los servicios que ofrece el sistema GPS, la variedad de receptores, el funcionamiento de sus componentes, así como los códigos que se emplean en este servicio. En este capítulo se abordara la etapa de procesamiento, específicamente al trabajo de adquisición de satélites GPS, con diferentes métodos y formas para extraer los datos de la señal.

## 2.1 Procesamiento de la Señal GPS

Si la señal GPS pertenece al tipo de señales de espectro expandido, el primer paso en la recepción, es la sincronización de la portadora local del receptor con la portadora de la señal recibida y la sincronización del código PRN de la señal de entrada con una copia generada localmente; esta sincronización es llevada a cabo en dos etapas[27].

La primera etapa es la adquisición y tiene por objetivo lograr un alineamiento muy aproximado entre el código de la señal de entrada y la copia generada localmente, habitualmente con una incertidumbre menor a un tiempo de chip. Una vez que la secuencia pseudoaleatoria ha sido adquirida, comienza la segunda etapa, llamada seguimiento, en la que se trata de mantener un alineamiento fino entre los dos códigos, mediante el uso de un lazo retroalimentado.

La sincronización del código C/A no es tan complicada debido a su corta duración, el receptor debe generar su propio código desplazándolo hasta que la correlación entre éste y el que recibe sea máxima. Para la sincronización, el receptor se engancha primero al código C/A, y a continuación obtiene la palabra HOW (*Hand Over Word*) del mensaje de información, que le indica la distancia al inicio del código-P[28].

Si se tiene conocimiento previo de la distancia que le separa del satélite al receptor, con una precisión de 10 a 20Km, y también conoce la hora exacta del GPS, entonces puede sincronizarse con el código-P sin tener que hacerlo previamente con el código-C/A. El mensaje de navegación se transmite a una velocidad de 50 bps, y contiene toda la información necesaria para la determinación de la posición.

Los datos que se incluyen son: la hora del sistema, correcciones a los relojes de los satélites, el almanaque con las posiciones de todos los satélites de la constelación e información acerca del estado del sistema. El mensaje consiste en una trama de 1500 bits subdividida en 5 que requiere de 6 segundos para ser transmitida[29].

Cada una de las subtramas contiene 10 palabras de 30 bits. Las dos primeras palabras de cada subtrama son la palabra TLM (*Telemetry Message*) y la palabra HOW. Estas dos palabras son generadas por el propio satélite, mientras que las restantes son generadas por el centro de control constituyendo los llamados data block (bloque de datos). Fig. 2.1.

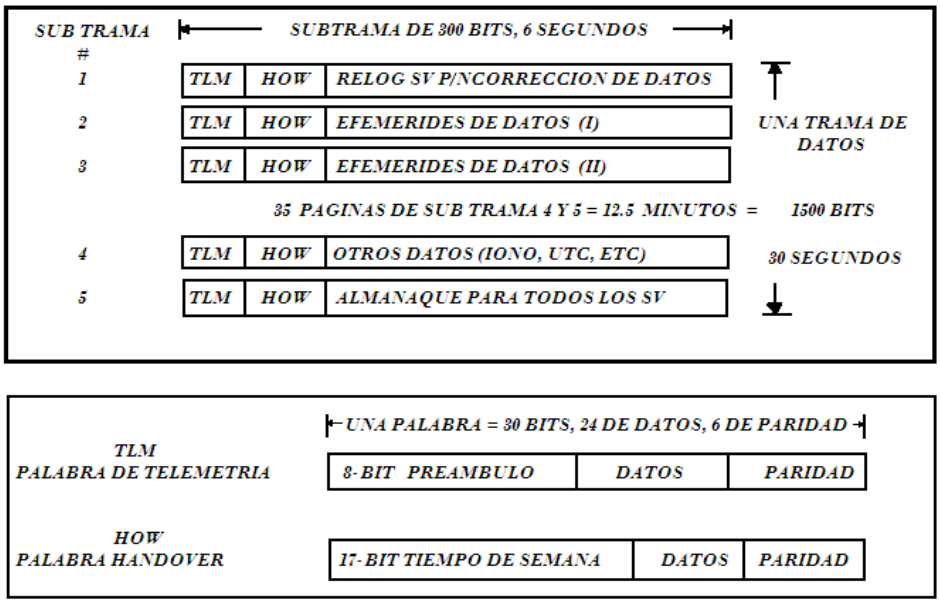



*Fig. 2.1 Formato de los Datos de Navegación del Sistema GPS*

La palabra TLM se utiliza por las estaciones de seguimiento para comprobar si los satélites han recibido de forma correcta las actualizaciones transmitidas desde la Tierra. Y la palabra HOW se emplea para facilitar la adquisición del código-P.

Dado que sólo se transmiten 6 segundos del almanaque en cada trama, se requieren un total de 750 segundos para transmitir el almanaque entero. Los equipos receptores deben almacenarlo en su memoria ya que es necesario para obtener información sobre la visibilidad de los satélites, de forma que se pueda seleccionar al grupo de satélites que proporcione la mejor geometría para determinar la posición.

Como ya se mencionó, el sistema GPS emplea dos códigos PRN: El código-P, diseñado para tener una duración de 280 días y que se transmite a una velocidad de 10.23 Mbps.; para esto todos los satélites están equipados con generadores de código-P idénticos, pero cada uno tiene asignado un segmento del código de 7 días de duración, que consiste en una sucesión aparentemente aleatoria de $6.2x10^{12}$ unos y ceros. El segundo es referido como código C/A con una longitud de 1023 chips que se transmite a una velocidad de 1.023 Mbps, por lo que se repite 1000 veces por segundo y cada satélite tiene asignado un código C/A específico.

## 2.2 Adquisición de Datos GPS

La señal GPS es obtenida por la antena y en conjunto con una terminal de entrada de RF se acondiciona y debido a que cuenta con interferencia y atenuación se pasa por varios filtros y amplificadores para convertir su nivel y ser movida del centro de frecuencias de RF a IF. Un convertidor A/D muestrea la señal analógica de entrada obedeciendo al criterio de Nyquist[30], resultando en una representación digital.

La adquisición de señales de espectro expandido de secuencia directa y con modulación BPSK, es usualmente realizada empleando detección no coherente[31]. Debido a esto, la adquisición de la señal GPS no sólo es un proceso de búsqueda en tiempo (fase de código PRN), sino que también es necesario realizar la búsqueda en el eje de frecuencias.

La búsqueda en la dirección del eje de código se realiza habitualmente en incrementos de medio tiempo de chip a lo largo de todas las posibles fases de código, resultando en un error de estimación menor a medio tiempo de chip.

La búsqueda en la dirección del eje de frecuencias es empleado el siguiente patrón: se recorre el eje de fases de código PRN, para una frecuencia portadora determinada; si no se consigue la alineación de código, se repite el procedimiento para otra frecuencia portadora. En la Figura 2.2 se puede apreciar un diagrama en bloques para la adquisición serie no coherente.

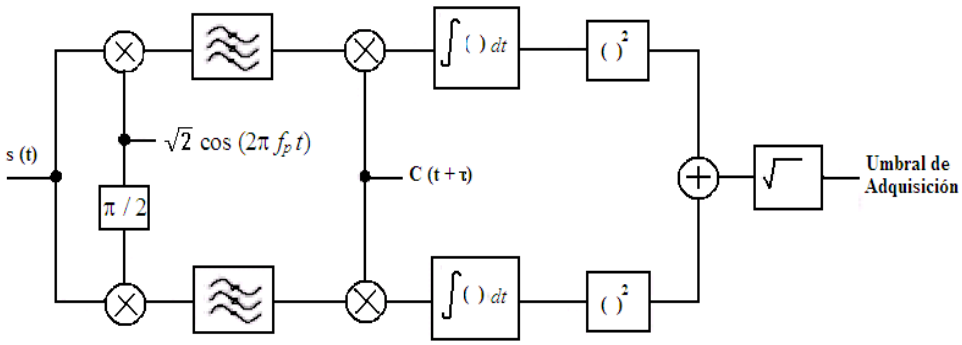



*Fig.2.2 Diagrama a Bloques de la Adquisición no coherente de una señal de espectro expandido*

## 2.3 Propósito de la Adquisición

La principal tarea de la adquisición es primero identificar si cierto satélite es al usuario o no visible. Si el satélite es visible, la adquisición debe determinar las dos características de la señal proveniente de ese satélite:

- *Frecuencia:* Es la portadora de un satélite específico que puede ser diferenciado de su valor nominal debido a que las señales son afectadas por el movimiento relativo del satélite causando un efecto Doppler. El corrimiento de la frecuencia con velocidad máxima del satélite y movimiento en el acercamiento al receptor, resulta en valores diferidos de hasta $\pm$10kHz y para un receptor inmóvil, el cambio de la frecuencia de Doppler nunca excederá 5 KHz.

- *Fase de Código*: Esta fase denota el punto actual en el bloque de datos donde el código C/A comienza.

Entonces la señal recibida *s* es una combinación de señales de los *n* satélites visibles:

$$s(t) = s^2(t) + s^2(t) + \ldots + s^n(t) \quad (2.1)$$

Al adquirir un satélite k, la señal *s* se multiplica con el código local C/A correspondiente a este satélite, la correlación cruzada entre códigos para diversos satélites implica que las señales de otros son casi quitadas por este procedimiento y para evitar eliminar el componente deseado se debe alinear correctamente y a tiempo con la fase del código de la señal de entrada.

Después de que es multiplicado con el código generado, la señal se debe mezclar con una onda de portadora local, para quitar la portadora de la señal recibida. Al quitar esta onda la frecuencia generada debe estar cerca de la frecuencia portadora de la señal de entrada.

Debido a que la frecuencia puede cambiar hasta ±10kHz de la frecuencia nominal, deben recorrerse diversas frecuencias dentro de este espectro, para identificar sí un satélite es visible o no. Esta búsqueda sobre las frecuencias es suficiente realizarla en pasos de 500Hz dando por resultado 41 diversas frecuencias, en caso de un receptor móvil y 21 para un receptor estático.

Después de mezclarse con la onda de portador generada localmente, todos los componentes de la señal se ajustan y se suman proporcionando un valor numérico. El desarrollo de la adquisición trabaja como procedimiento de búsqueda, para cada una de las diversas frecuencias se intentan 1023 fases del código.

Cuando todas las posibilidades de la fase y de la frecuencia del código se intentan, una búsqueda para el valor máximo se realiza. Si el valor máximo excede un umbral, el satélite se adquiere con la frecuencia y el desplazamiento de fase correspondientes.

La Figura 2.3 muestra un diagrama típico de la adquisición, realizado para un satélite. El diagrama demuestra un pico significativo, que indica un valor alto en la función de correlación y por el cual el satélite se está adquiriendo. Para el caso en que el pico está ausente el satélite no es visible al receptor

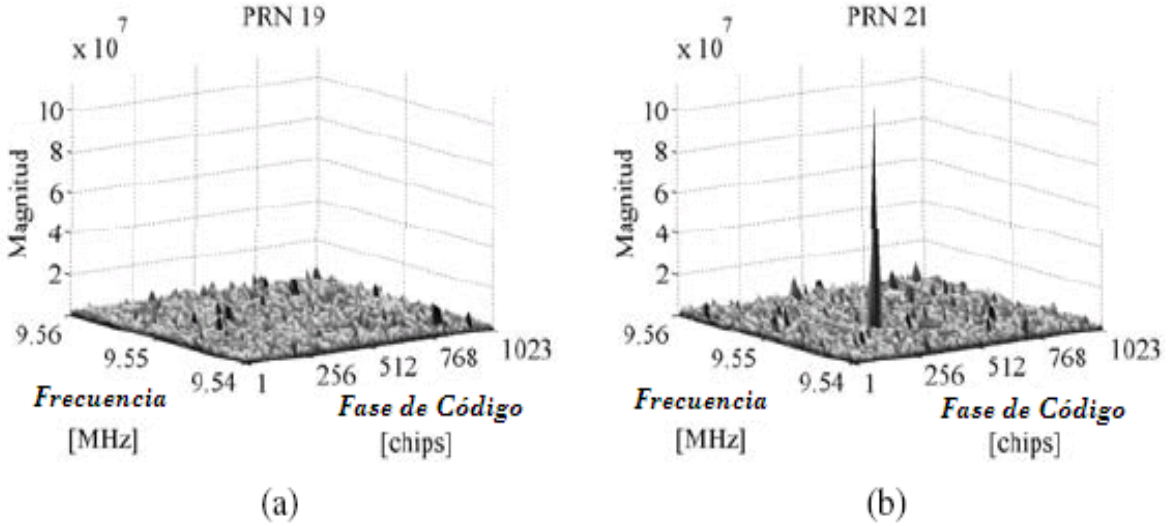



*Fig. 2.3 (a) Muestra un grafico típico de adquisición, realizado para un satélite el cuál no es actualmente visible en el receptor GPS. (b). Las señales que se originan del satélite 21 están presentes en la señal recibida. Esto se refleja en el pico significativo que se observa en el grafico de la adquisición. El pico representa la fase del código C/A y la frecuencia de la señal de entrada.*

## 2.4 Algoritmos para la Adquisición de Datos GPS

La adquisición debe determinar a los satélites visibles, los valores de frecuencia de portadora y cifrar la fase de las señales en los satélites a través del código PRN.

Los satélites son distinguidos por tres diferentes parámetros. El primero son las 32 diversas secuencias de PRN. El segundo parámetro es la fase del código, que es la alineación en tiempo con el código de PRN en el bloque actual de datos, necesaria para generar un código local de PRN, que se alinee perfectamente con el código entrante y poder quitar la señal L1. El tercer parámetro es la frecuencia portadora, que cuando se somete a la conversión de bajada corresponde al valor de Frecuencia Intermedia (IF).

Esta IF es obtenida de la frecuencia portadora L1 de 1575.42MHz y de los mezcladores en el convertidor. Sin embargo, la frecuencia puede desviarse del valor previsto, dando por resultado una frecuencia más alta o más baja.

A partir de estas características que distinguen a cada satélite, se generan tres métodos y algoritmos para identificar si uno de estos satélites está a la vista del receptor y al cual se pueda extraer la información contenida en los códigos que lo distinguen.

Las secciones siguientes describirán la teoría detrás de estos diversos métodos de adquisición, para demostrar la posibilidad de implementación, orientado a la aplicación de un receptor GPS basado en software.

### 2.4.1 Adquisición por Búsqueda Serial

La adquisición por búsqueda serial es uno de los primeros métodos usados en los sistemas CDMA dentro de los característicos sistemas de espectro expandido. La Figura 2.4 es un diagrama a bloques de este algoritmo.

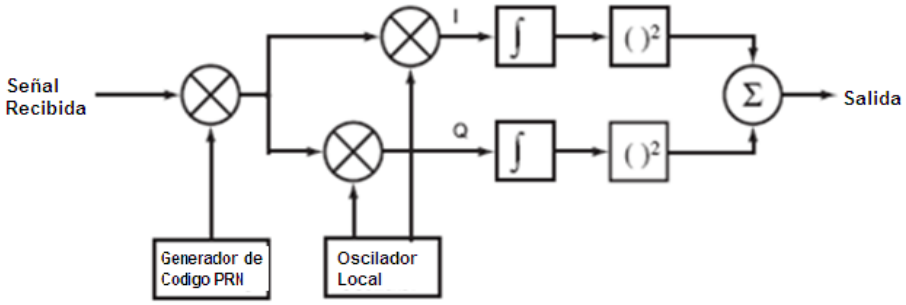



*Fig.2.4 Diagrama a Bloques de de la Adquisición por Búsqueda Serial*

Según lo considerado en el diagrama, el algoritmo se basa en la multiplicación de las secuencias del código de PRN y de la señal portadora, generadas localmente. Este generador arroja una secuencia correspondiente a la de un satélite específico, con cierta fase de código a partir de 1 hasta 1023 chips. La señal de entrada es multiplicada inicialmente por esta secuencia local de PRN, después con una señal de portadora local. La multiplicación genera la señal I, y con una versión defasada 90° genera la señal Q.

Las señales I y Q se integran sobre cierto período y finalmente se ajustan y se suman. Idealmente, la energía de la señal se debe situar en la parte de la señal I, pues el código C/A se modula solamente sobre éste. Sin embargo la señal I generada en el satélite puede no corresponder necesariamente a la I demodulada, debido a que la fase de la señal es desconocida. Por lo tanto si una señal es detectada, es necesario investigar I y Q.

A final de cuentas, la salida del sistema es el valor de correlación que hay entre la señal de entrada y la señal generada localmente. Si se excede un umbral predefinido, los parámetros de frecuencia y fase de código, se asumen como correctos y los parámetros se pueden pasar directamente a la etapa de seguimiento.

El algoritmo de búsqueda serial hace dos barridos: uno sobre todas las frecuencias portadoras posibles de IF ±10kHz en pasos de 500Hz y otro sobre la fase en los 1023 diversos códigos.

### 2.4.2 Adquisición por Búsqueda Paralela en el Espacio de la Frecuencia.

El método de adquisición serial es un procedimiento que consume mucho tiempo al buscar secuencialmente todos los valores posibles para ambos parámetros, pero si alguno de los dos se pudiera eliminar del procedimiento de búsqueda o ejecutarse en paralelo, el desempeño de la adquisición aumentaría perceptiblemente.

La adquisición por búsqueda paralela en el espacio de la frecuencia, paraleliza la búsqueda para un único parámetro. Este método utiliza la transformada de Fourier, para realizar una transformación del dominio de tiempo al dominio de la frecuencia. La figura 2.5 muestra el diagrama a bloques que describe su funcionamiento.

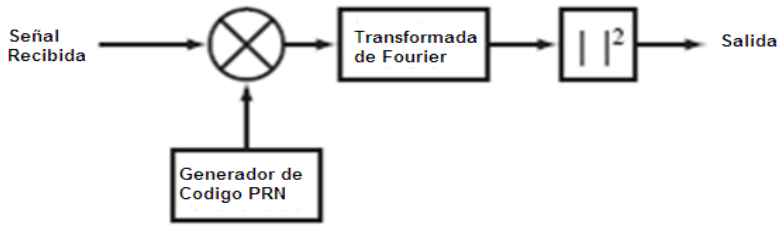



*Fig. 2.5 Diagrama a bloques de la Adquisición por búsqueda paralela en el espacio de frecuencia*

La señal de entrada es multiplicada por una secuencia de PRN generada localmente, con un código que corresponde a un satélite específico y a una fase de código entre 0 y 1023 chips. La señal que resulta es pasada al dominio de la frecuencia con una transformada de Fourier, puesta en ejecución como una FFT de orden $2^{[33]}$, si esta señal contiene componentes de otros satélites, estos serán reducidos al mínimo como resultado de las características de la correlación cruzada de las secuencias PRN.

El resultado es una señal en onda continua, solamente cuando el código generado localmente se alinea perfectamente con el código de la señal de entrada.

La Figura 2.6 ilustra el resultado de multiplicar la señal de entrada con una secuencia generada localmente, y alineada perfectamente a la secuencia PRN.

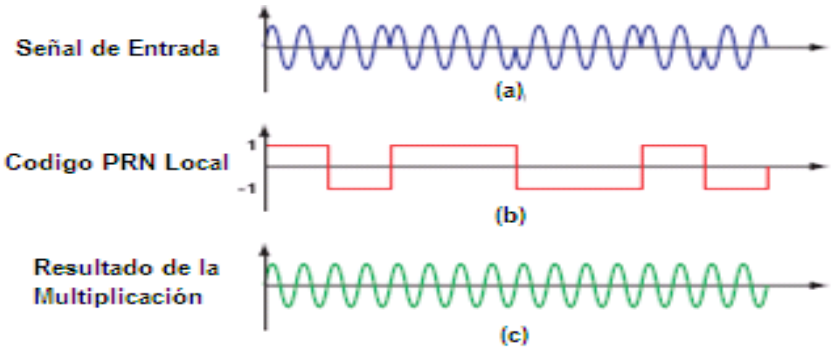



*Fig. 2.6 Demodulación del código PRN. (a) El código PRN modulado sobre la onda portadora. (b) La secuencia perfectamente alineada de PRN. (c) Después de la multiplicación de la señal entrante con la secuencia perfectamente alineada de PRN, la señal que resulta es una onda continua.*

La señal resultante*(c)*, es sometida a la transformada de Fourier y si el código es perfectamente alineado a la secuencia PRN, la salida de la transformada mostrará un pico distinto en magnitud, que será situado en el índice de la frecuencia que corresponde a la señal de la onda continua y por consecuencia a la frecuencia de la señal de portadora de la entrada.

La exactitud para determinar la frecuencia depende de la longitud de la transformada de Fourier y corresponde al número de muestras en los datos analizados, si 1ms de datos se analiza, el número de muestras se puede encontrar como 1/1000 por la frecuencia de muestreo que es *fs* = 10MHz y el número de muestras es N = 10000. Entonces, la frecuencia resultante se obtiene como:

$$\Delta f = \frac{f_S/2}{N/2} = \frac{f_S}{N} \quad (2.2)$$

Y para el caso de *fs*=10MHz, la frecuencia resultante es:

$$\Delta f = \frac{10\ MHz}{10000} = 1\ kHz \quad (2.3)$$

En este caso la frecuencia portadora estimada es de 1kHz y comparado con la de 500Hz en la adquisición serial, presenta mayor exactitud.

La Figura 2.7 muestra dos diagramas de la salida de la transformada de Fourier,*(a)* es la salida de la transformada de Fourier cuando el código generado tiene la fase perfectamente alineada con el código de la señal de entrada y se puede notar en el pico que sobresale en el diagrama.*(b)* Demuestra que la salida de la transformada con una fase no alineada del código origina la ausencia del pico en el diagrama.

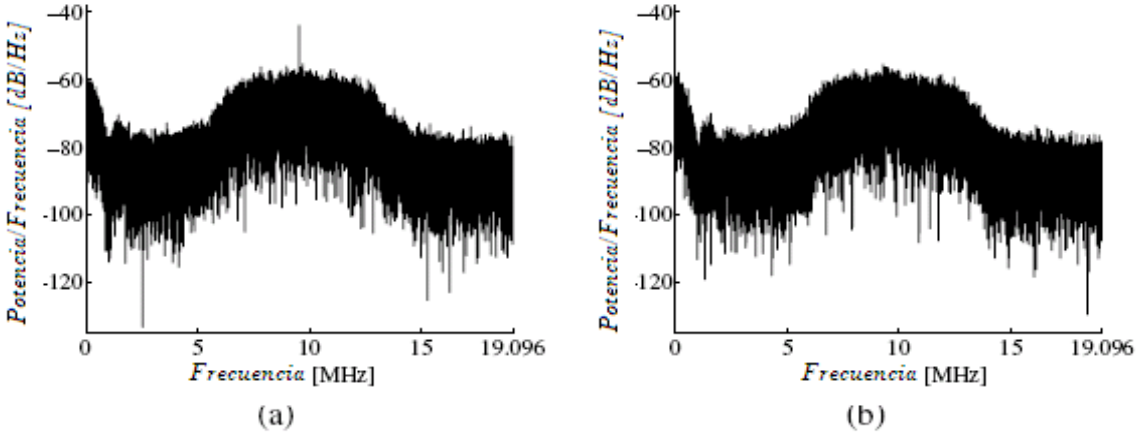



*Fig. 2.7 Transformada de Fourier de la señal de entrada multiplicada por una secuencia generada localmente del código de PRN. (a) Cuando se multiplica con un código perfectamente alineado de PRN, la salida mostrará un pico en la frecuencia portadora. (b) Al multiplicarse con un código no alineado, la salida no mostrará ningún pico.*

A diferencia de los pasos seguidos para la adquisición serial, que son a través de todas las posibles frecuencias y códigos de fase, la adquisición paralela prueba solamente con las 1023 diversas fases del código, esta ventaja implica el costo de una transformación al dominio de la frecuencia con cada código de fase y dependiendo de la puesta en práctica de esa transformación, debe ser posible hacer más práctico a este método, comparado al método serial.

### 2.4.3 Adquisición por Búsqueda Paralela en el Espacio de la Fase de Código.

Si la cantidad de pasos de búsqueda en la dimensión de fase de código es perceptiblemente más pequeño que en la dimensión de la frecuencia y si el método anterior eliminaba la necesidad de buscar las 41 frecuencias posibles. Ahora en la dimensión de fase de código se hace de igual forma paralelismo y sólo 41 pasos se deben realizar en comparación a los 1023 del el algoritmo anterior.
Uno de los métodos más recientes para la adquisición de la señal de GPS utiliza la ventaja mencionada.

Este método se refiere como adquisición por búsqueda paralela en el espacio de la fase de código, donde la meta es realizar la correlación con la señal de entrada y un código PRN, pero en vez de multiplicar la señal de entrada con un PRN de 1023 diversas fases del código es más conveniente hacer una correlación circular entre la entrada y el código PRN pero con un corrimiento en el código de fase.

Cuando la representación del dominio de la frecuencia de la correlación se encuentra, la representación en el dominio del tiempo se puede obtener a través de la transformada inversa de Fourier.

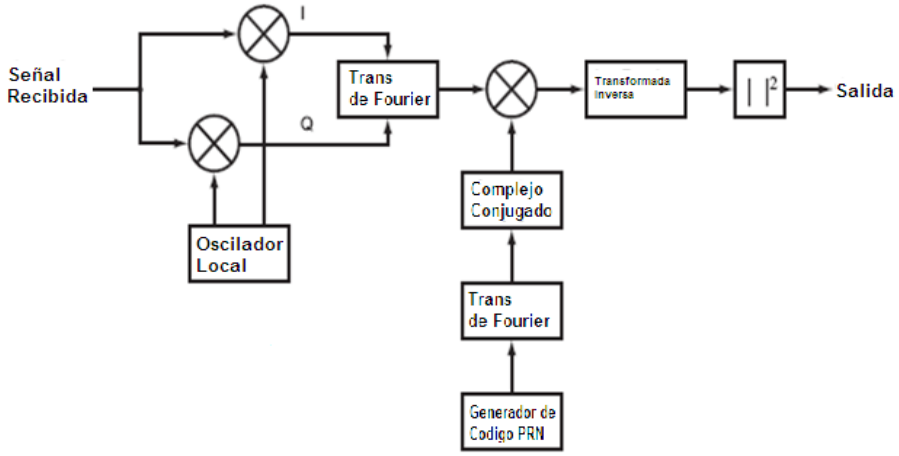



*Fig.2.8 Diagrama a Bloques de Adquisición por Búsqueda Paralela en el Espacio del Código de Fase*

La Figura 2.8 es un diagrama a bloques de este algoritmo y se observa como la señal de entrada es multiplicada por una señal portadora generada localmente, para obtener a *I*, y con la versión desplazada 90 a *Q*. Combinadas I y Q forman una señal compleja $x(n) = I(n) + jQ(n)$, que es sometida al cambio de dominio por la transformada de Fourier.

Si la transformada de Fourier discreta de secuencia finita *x(n)* de longitud N es calculada como:

$$X(k) = \sum_{n=0}^{N-1} x(n)\, e^{-j2\pi kn/N} \quad (2.4)$$

La correlación de cruce entre dos secuencias finitas *x(n)* y *y(n)* cada una con una longitud N, es calculada como:

$$z(n) = \sum_{m=0}^{N-1} x(m)y(n+m) \quad (2.5)$$

La convolución entre dos secuencias finitas *x(n)* y *y(n)* es calculada como:

$$z(n) = x(n)^* y(n) = \sum_{m=0}^{N-1} x(m)y(n-m) \quad (2.6)$$

Las ecuaciones (2.5) y (2.6) muestran que la única diferencia entre la correlación de cruce y la convolución entre dos secuencias finitas es el signo interno *y(n + m)*, entonces la combinación de la ecuación (2.4) y (2.6) da la transformada de Fourier de la convolución entre *x* y *y*:

$$Z(k) = \sum_{n=0}^{N-1} \sum_{m=0}^{N-1} x(m)y(n-m)\, e^{-j2\pi kn/N}$$

$$= \sum_{m=0}^{N-1} x(m)\, e^{-j2\pi km/N} \sum_{n=0}^{N-1} y(n-m)\, e^{-j2\pi k(n-m)/N}$$

$$= X(k)\, Y(k) \qquad (2.7)$$

Esto verifica la característica de convolución-multiplicación de la transformada de Fourier, convolución en tiempo corresponde a la multiplicación en frecuencia:

$$z(n) = x(n) * y(n)$$

$$\downarrow F$$

$$Z(k) = X(k)\, Y(k)$$

La combinación de la ecuación (2.4) y (2.5) da la transformada de la relación de cruce entre *x* y *y*:

$$Z(k) = \sum_{m=0}^{N-1} \sum_{m=0}^{N-1} x(m)y(n+m)\, e^{-j2\pi kn/N}$$

$$= \sum_{m=0}^{N-1} x(m)\, e^{-j2\pi km/N} \sum_{n=0}^{N-1} y(n+m)\, e^{-j2\pi k(n+m)/N}$$

$$= X(k)^* Y(k) \qquad (2.8)$$

Es decir, la conexión entre una correlación del dominio tiempo y la representación del dominio de la frecuencia es casi similar a la conexión entre la convolución y su representación del dominio de la frecuencia. La única diferencia es que la transformada de Fourier de una de las dos secuencias de entrada debe ser el complejo conjugado de la multiplicación anterior. Cuando la representación del dominio de la frecuencia de la correlación se encuentra, la representación en el dominio del tiempo se puede encontrar a través de la transformada inversa de Fourier.

Comparado a los métodos anteriores de adquisición, el método paralelo de adquisición de búsqueda en la fase de código, ha reducido el espacio de la búsqueda a las 41 diversas frecuencias portadoras, para las cuales debe realizarse una transformada de Fourier y una transformada inversa, así que la eficiencia del método depende de la puesta en práctica de estas funciones.

La exactitud de los parámetros estimados por este método de adquisición está considerando una frecuencia similar a la del método serial. La fase del código PRN sin embargo, es más exacta comparada a los otros métodos pues da un valor correlación para cada fase muestreada del código. Es decir, si la frecuencia de muestreo es 10MHz, un código PNR muestreado tiene 10000 muestras, la exactitud de la fase del código puede tener 10000 diversos valores en vez de 1023.El código generado PRN es transformado al dominio de la frecuencia, tomando el complejo conjugado y se multiplica con la transformada de Fourier de la entrada, el resultado de la multiplicación es transformado al dominio del tiempo por una transformada inversa de Fourier.

# Capítulo 3

# DISPOSITIVOS LÓGICOS PROGRAMABLES

En los capítulos anteriores se ha analizado el sistema GPS, el tratamiento de sus señales y el funcionamiento de sus receptores; sin embargo para conformar de manera completa la tarea y cumplir con los objetivos de este trabajo, ahora se estudia la parte correspondiente al hardware. Debido a que se debe trabajar sobre una plataforma capaz de cumplir con los requerimientos del sistema, en este capítulo se hace el análisis de los dispositivos FPGA, los cuales cuentan con las características requeridas y aportan ventajas al sistema final.

## 3.1 Sistemas Embebidos

Un sistema embebido[1] es un sistema informático de uso específico, construido dentro de un dispositivo mayor, donde la mayoría de los componentes se encuentran incluidos en la placa base. En general, un sistema embebido consiste en un sistema con microprocesador cuyo hardware y software están específicamente diseñados y optimizados, para resolver un problema concreto eficientemente. Muchas veces un sistema embebido es un componente de un sistema mucho más grande.

El término embebido hace referencia al hecho de que el microprocesador está encerrado o instalado dentro de un sistema mayor y su existencia como microcontrolador puede no ser aparente. Un sistema embebido complejo puede utilizar un sistema operativo como apoyo para la ejecución de sus programas, sobre todo cuando se requiere la ejecución simultánea de los mismos.

Cuando se utiliza un sistema operativo lo más probable es que se tenga que tratar de un sistema operativo de tiempo real (RTOS)[2], que es un sistema operativo diseñado y optimizado para manejar fuertes restricciones de tiempo asociadas con eventos en aplicaciones de tiempo real. En una aplicación de tiempo real compleja la utilización de un sistema operativo de tiempo real multitarea puede simplificar el desarrollo del software; entendiéndose por sistemas en tiempo real a aquellos sistemas en los que el control del tiempo es vital para el correcto funcionamiento. Los sistemas en tiempo real necesitan realizar ciertas operaciones o cálculos en un límite de tiempo. Donde ese límite de tiempo resulta crucial o esencial.

Los microcontroladores son hoy una de las opciones más comunes a la hora de implementar sistemas embebidos[3]. Si bien no poseen la capacidad de procesamiento de una computadora de propósito general, si cuentan con la suficiente capacidad para el desarrollo de tareas simples y repetitivas. Cuando la tarea es simple, el costo asociado a trabajar con un microcontrolador es mucho menor.

Ya que el software y el hardware pueden ser reemplazados, en muchos casos por un circuito integrado que realice la misma tarea, se brinda de gran flexibilidad al sistema embebido a la hora de realizar alguna modificación en las líneas de código, esta acción acelera el tiempo de diseño pero no optimiza ni el tamaño del sistema ni el número de componentes utilizados.

Éste tipo de hardware se diseña normalmente a nivel de chips, o de interconexión de PCB (Process Control Block), buscando la mínima circuitería y el menor tamaño para una aplicación particular. El diseño a nivel de PCB[4] consiste en el ensamblado de placas con microprocesadores comerciales.

### 3.1.1 Arquitectura de un Sistema Embebido.

Un sistema embebido simple, contará con un microprocesador, memoria, unos pocos periféricos de entrada/salida y un programa dedicado a una aplicación concreta, almacenado permanentemente en la memoria. Estos sistemas se caracterizan normalmente por la necesidad de dispositivos especiales, para cumplir con los requisitos de la aplicación concreta.

Un sistema embebido en principio estaría formando por un microprocesador y un software que se ejecute sobre este. Sin embargo este software necesitara sin duda un lugar donde poder guardarse para luego ser ejecutado por el procesador, esto podría tomar la forma de memoria RAM o ROM. Todo sistema embebido necesitará en alguna medida una cierta cantidad de memoria, la cual puede incluso encontrarse dentro del mismo chip del procesador. Además de esto normalmente contará con una serie de salidas y entradas necesarias para comunicarse con el mundo exterior. Debido a que las tareas realizadas por sistemas embebidos son de relativa sencillez, los procesadores comúnmente usados cuentan con registros de 8 o 16 bits[5].

En su memoria solo reside el programa destinado a gobernar una aplicación determinada, sus líneas de Entrada/Salida soportan la conexión de los actuadores del dispositivo a controlar y todos los recursos complementarios disponibles, teniendo como única finalidad atender a sus requerimientos. Estas son las únicas características que tienen en común los sistemas embebidos, todo lo demás será totalmente diferente para cada sistema embebido en particular.

### 3.1.2 Componentes de un Sistema Embebido.

En la parte central de un sistema como este, se encuentra el microprocesador, microcontrolador, DSP, etc. Es decir, la unidad que aporta inteligencia al sistema. La comunicación adquiere gran importancia en los sistemas embebidos. Lo normal es que el sistema pueda comunicarse mediante interfaces estándar de cable o inalámbricas. Así un sistema embebido normalmente incorporará puertos de comunicaciones del tipo RS232, RS485, SPI, I²C, CAN, USB, IP, WiFi, GSM, GPRS, DSRC, etc.[6].

El módulo de E/S analógicas y digitales suele emplearse para digitalizar señales analógicas procedentes de sensores.

El módulo de reloj es el encargado de generar las diferentes señales de reloj a partir de un único oscilador principal. El tipo de oscilador es importante por varios aspectos: la frecuencia necesaria, la estabilidad necesaria y el consumo de corriente requerido. El oscilador con mejores características son los basados en resonador de cristal de cuarzo, mientras que los que requieren menor consumo son los RC.

El módulo de energía, que se encarga de generar las diferentes tensiones y corrientes necesarias para alimentar los diferentes circuitos del sistema embebido. Usualmente se trabaja con un rango de posibles tensiones de entrada, que mediante conversores AC/DC o DC/DC se obtienen las diferentes tensiones para los diversos componentes activos del circuito.

### 3.1.3 Sistemas Embebidos Basados en FPGA.

Se ha resaltado que los sistemas embebidos son sistemas altamente integrados que suelen ocupar muy poco espacio y tener un consumo de potencia muy reducido.

Con la entrada al mercado de los FPGA (*Field Programmable Gate Array*) los sistemas embebidos aun se pueden reducir más, y ya podemos hablar de sistemas SoC[7] (system on chip, o sistema en

chip), es decir todo un sistema compuesto de procesador, memoria, y circuitos de apoyo personalizados, pueden entrar en un solo microchip.

Esto brinda una solución completa de desarrollo para trabajar más eficientemente, generando prototipos e implementando aplicaciones embebidas en una sola plataforma. Esta plataforma es capaz de ejecutarse en una variedad de aplicaciones incluyendo sistemas comerciales en tiempo real.

El trabajo con los FPGA resulta sencillo con el diseño en sistemas gráficos, así se puede diseñar, generar prototipos y desplegar rápidamente sistemas embebidos[8]; usando el software de programación con hardware comercial perfectamente integrado, ya sea para diseñar vehículos autónomos, dispositivos médicos, máquinas industriales, máquinas semiconductoras, electrónica de consumo u otros dispositivos personalizados; aprovechando la plataforma embebida del FPGA para mejorar el desempeño del diseño.

Las arquitecturas reconfigurables con núcleos de procesadores embebidos como las Virtex-II Pro y Virtex-4 de Xilinx[9], son algunas de las plataformas adecuadas para el diseño de estos sistemas embebidos.

El objetivo de trabajar con este tipo de dispositivos es desarrollar las interfaces de hardware necesarias para construir un sistema embebido basado en una arquitectura reconfigurable FPGA y un sistema operativo de código abierto con capacidades de entrada y salida de datos.

## 3.2 FPGA

Los FPGA[10] a groso modo son chips que se pueden transformar en diversos circuitos para alguna aplicación especifica siempre y cuando este disponga de los recursos necesarios. Estos circuitos se suelen crear describiéndolos con lenguajes especiales para hardware.

Las aplicaciones donde más comúnmente se utilizan los FPGA incluyen al DSP (procesamiento digital de señales), radio definido por software, sistemas aeroespaciales y de defensa, prototipos de ASIC, sistemas de imágenes para medicina, sistemas de visión para computadoras, reconocimiento de voz, bioinformática, emulación de hardware de computadora, entre otras.

Dentro del modo de trabajo de los FPGA existe código fuente disponible de sistemas como microprocesadores, microcontroladores, filtros, módulos de comunicaciones y memorias, entre otros. Estos códigos son llamados núcleos (*cores)* de un sistema.

El FPGA es un dispositivo semiconductor que contiene bloques de lógica cuya interconexión y funcionalidad se puede programar. La lógica programable puede reproducir desde funciones tan sencillas como las llevadas a cabo por una puerta lógica o un sistema combinacional, hasta complejos sistemas en un chip.

Históricamente los FPGA surgen como una evolución de los conceptos desarrollados en los PLA y los CPLD[11]. Son inventados en el año 1984 por Ross Freeman[12], co-fundador de Xilinx, y surgen como una evolución de los CPLD. Tanto los CPLD como las FPGA contienen un gran número de elementos lógicos programables, si medimos la densidad de los elementos lógicos programables en puertas lógicas equivalentes, podríamos decir que en un CPLD hallaríamos del orden de decenas de miles de puertas lógicas y en una FPGA del orden de cientos de miles hasta millones de ellas.
Aparte de las diferencias en densidad entre ambos tipos de dispositivos, la diferencia fundamental entre las FPGA y los CPLD es su arquitectura. La arquitectura de los CPLD es más rígida y consiste

en una o más sumas de productos programables cuyos resultados van a parar a un número reducido de biestables síncronos, también denominados flip-flop y en cambio la arquitectura de los FPGA se basa en un gran número de pequeños bloques utilizados para reproducir sencillas operaciones lógicas, que cuentan a su vez con biestables síncronos. La enorme libertad disponible en la interconexión de dichos bloques agrega a los FPGA gran flexibilidad.

Otra diferencia importante es que en la mayoría de los FPGA se pueden encontrar funciones de alto nivel como sumadores y multiplicadores, todas de forma embebidas en la propia matriz de interconexiones, así como bloques de memoria.

### 3.2.1 FPGA vs ASIC

Los FPGA se utilizan en aplicaciones similares a los circuitos integrados de aplicación específica ASIC, sin embargo son más lentos, tienen un mayor consumo de potencia y no pueden abarcar sistemas tan complejos como ellos. A pesar de esto, los FPGA tienen las ventajas de ser reprogramables después de haber salido al mercado, a fin de corregir posibles errores, además de que sus costos de desarrollo, investigación, diseño y pruebas son menores; así como una reducción del tiempo de introducción al mercado, su adquisición en menores y pequeñas cantidades y el tiempo de desarrollo menor.

Hoy en día existen en el mercado fabricantes que integran un microprocesador y los elementos controladores de los dispositivos fundamentales de entrada y salida en un mismo chip, pensando en las necesidades de los sistemas embebidos.

Su capacidad de proceso suele ser inferior a los procesadores de propósito general pero cumplen con su cometido ya que los sistemas donde se ubican no requieren tanta potencia.

### 3.2.2 Arquitectura del FPGA

Una tendencia reciente ha sido combinar los bloques lógicos e interconexiones de los FPGA con microprocesadores y periféricos relacionados para formar un sistema programable en un chip. Ejemplo de tales tecnologías híbridas pueden ser encontradas en los dispositivos Virtex-II PRO y Virtex-4 de Xilinx, los cuales incluyen uno o más procesadores PowerPC embebidos, junto con la lógica del FPGA.

El FPSLIC[13] de Atmel es otro dispositivo similar, el cual usa un procesador AVR en combinación con la arquitectura lógica programable de Atmel. Otra alternativa es hacer uso de núcleos de procesadores implementados haciendo uso de la lógica del FPGA. Esos núcleos incluyen los procesadores MicroBlaze y PicoBlaze, Nios y Nios II de Altera, y los procesadores de código abierto: LatticeMicro32 y LatticeMicro8.

Muchos FPGA modernos soportan la reconfiguración parcial del sistema, permitiendo que una parte del diseño sea reprogramada, mientras las demás partes siguen funcionando. Este es el principio de idea en la computación reconfigurable, o los sistemas reconfigurables.

Ahora conoceremos la estructura interna que hace posible que los FPGA tengan gran flexibilidad en su trabajo. Éstos tienen tres tipos de componentes en su interior: CLB, IOB y Red de interconexión.

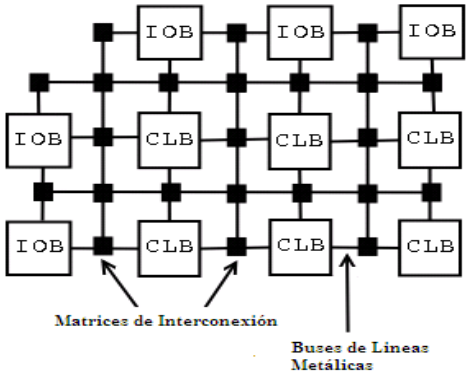



*Fig. 3.1 Estructura Interna de un FPGA*

Los CLB (*Bloques lógicos configurables*) son la parte básica del dispositivo, se encuentran distribuidos uniformemente por toda el área del FPGA y todos son idénticos y dependiendo del tipo de FPGA, estos bloques lógicos tienen unos componentes u otros; pero básicamente tienen multiplexores y pequeñas memorias para implementar funciones combinacionales. Estos bloques son configurables, estableciéndose qué elementos se conectan con cuales y qué funciones combinacionales realiza, si es que realiza alguna.

Los IOB (*Bloques de Entrada/Salida*) son una especie de CLB especializado, que se encuentran junto a los pines del chip y tienen la función de interconectar la lógica interna con el exterior.

La red de interconexión es un conjunto de caminos formados por líneas metálicas y matrices de interconexión, que permiten la conexión entre CLB y IOB. En la figura 3.1 se muestra gráficamente la estructura del FPGA. Los rectángulos negros representan las matrices de interconexión, a las que llegan buses de líneas metálicas.

Los bloques exteriores son los IOB y lo interiores los CLB. Todas las funciones lógicas, multiplexores, decodificadores, biestables, etc.; se implementan utilizando los componentes internos y en cuanto mayor sea el circuito a diseñar, mayor cantidad de CLB se necesitan.

La etapa de software establece las conexiones entre todos los CLB con el exterior a través de los IOB, esta tarea la realiza convirtiendo nuestro diseño en un fichero de configuración para el FPGA del tipo *Bitstream*[14]. Entonces si la arquitectura básica consiste en una matriz de bloques lógicos programables CLB y canales de comunicación, un bloque lógico típico de este modelo consiste en una tabla de funciones lógicas de cuatro entradas y un flip-flop como se muestra en la figura 3.2.

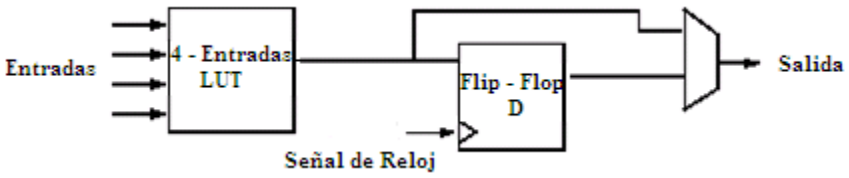



*Fig. 3.2 Componentes de un Bloque Lógico Programable*

Hay solamente una salida, la cual puede ser la salida registrada por el flip-flop, o la salida de la tabla de funciones lógicas. Este bloque tiene 4 entradas y una señal de reloj para el flip-flop; la cual es manejada por separado en los FPGA comerciales. Para esta arquitectura, la localización de los pines de los bloques lógicos de la FPGA son mostrados en la figura 3.3.

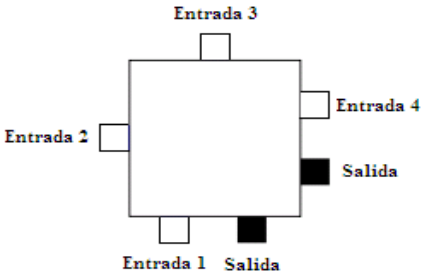



*Fig. 3.3 Localización de los pines en el Bloque Lógico del FPGA*

Cada entrada es accesible desde un lado del bloque lógico, mientras que el pin de salida puede conectarse a cables de comunicación en ambos canales. Cada pin de salida del bloque lógico puede conectarse a cualquier segmento de línea en el canal de comunicación adyacente a él. Similarmente, un conector de Entrada/Salida puede conectarse a cualquier segmento de línea en el canal de comunicación.

En cada punto donde se intercepta un canal de conexión, hay una caja de interruptores la cual permite conectar una línea a otras tres posibles dentro del segmento del canal. Una línea solo puede conectarse con otra de las tres pero no puede conectarse directamente con líneas de otras intersecciones. Figura 3.4.

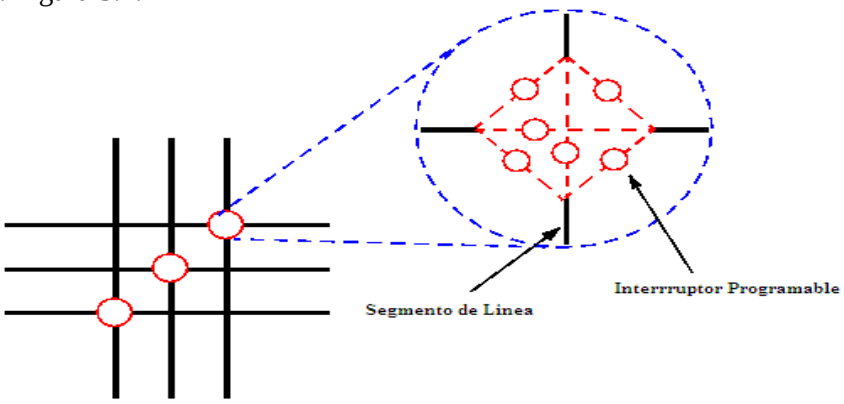



*Fig.3.4 Caja de Interruptores*

Los bloques usados como interfaz entre el FPGA y otros dispositivos, son los IOB y definen si un pin del FPGA será usado como entrada o como salida en el sistema implementado.

## 3.3 Metodología de Diseño con FPGA

Si los componentes básicos están integrados en un chip mayor en el que solo se debe especificar su conexión y se pueden unir según las necesidades, es decir, se puede configurar para que tome la forma del circuito a diseñar. Esta configuración se almacenara en una memoria interna, y se cargará desde el exterior, de igual forma que en los microcontroladores se carga el software, en los FPGA se carga la configuración que determina en qué circuito se va a convertir.

La tarea del programador es definir la función lógica que realizará cada uno de los CLB, seleccionar el modo de trabajo de cada IOB e interconectarlos.

El diseñador cuenta con la ayuda de entornos de desarrollo especializados en el diseño de sistemas implementados en un FPGA, puede ser capturado ya sea como un archivo esquemático o haciendo uso de un lenguaje de programación especial. Estos lenguajes de programación son conocidos como HDL o *Hardware Description Language*, que se pueden traducir como lenguajes de descripción de hardware. Los HDL más utilizados son: VHDL, Verilog y ABEL.

La metodología de diseño es similar a la cualquier sistema digital, salvo que al final obtenemos un fichero o archivo ejecutable que se envía al FPGA para que se reconfigure, implementando así el diseño.

Es posible realizar una descripción de hardware utilizando algún lenguaje, con esta descripción se pueden realizar simulaciones del circuito, para comprobar que lo diseñado es correcto y que hace lo requerido o si es necesario volver a modificar la descripción, esquemas o programa; hasta que la simulación sea satisfactoria.

A partir de la especificación del hardware y utilizando un compilador especial, se obtiene el archivo que contiene toda la información necesaria para configurar el FPGA. Este archivo, que es el equivalente a un programa ejecutable en software, es el que se descarga en el FPGA.

Así, se tiene el hardware situado en el interior de un chip y los cambios en el diseño se pueden hacer igual de rápidos que en el software, solo cambiando la especificación del diseño y reconfigurar el FPGA con el nuevo *bitstream* generado.

Con la aparición de los microcontroladores, la electrónica, o mejor dicho, el diseño electrónico ha ido evolucionando hacia el mundo del software y los FPGA permiten iniciar esos diseños en hardware puro, utilizando una metodología de diseño muy parecida a la del software.

### 3.3.1 Lenguajes de Descripción Hardware

Para describir el diseño de un circuito pueden ser utilizados los *lenguajes de descripción hardware*. Existen varios lenguajes que desarrollan la tarea como: VHDL, Verilog, Handle C, JBits. La ventaja de estos lenguajes es que además de permitir describir el circuito, permiten definir bancos de pruebas, que son muy útiles para la simulación y la depuración de los sistemas que describen.

En la simulación lo que vemos son los estímulos de entrada y sus correspondientes salidas. Para diseños no muy complejos, se utilizan herramientas gráficas que dibujan las señales para comprobar que efectivamente las salidas son las esperadas. Existiendo la posibilidad de que el propio banco de pruebas compruebe si las salidas son las correctas.

Una vez simulado y depurado nuestro diseño, se utiliza un sintetizador, llamado genéricamente compilador, que a partir del archivo fuente crea el archivo ejecutable *Bitstream*. Este archivo se puede cargar tantas veces como se requiera como el archivo original o modificado.

Se observa que usando FPGA, el diseño en hardware se acerca mucho al del software y a continuación se muestra el paralelismo que existe entre los dos tipos de desarrollo.

- Para poder desarrollar software se requiere tener una computadora, donde se genera y compila el código fuente, y posiblemente donde se realicen pruebas al ejecutable.
- Para poder desarrollar hardware se necesita la computadora, donde es editado, simulado y sintetizado el hardware, además de una placa de desarrollo con un FPGA donde descargar el diseño.

- Existe un archivo fuente que contiene toda la información.
- Existe un archivo final ejecutable que es entendido por el CPU/FPGA empleado (ejecutable/bitstream).
- Existe una herramienta que convierte los ficheros fuente en ejecutables: compilador/sintetizador.
- Se puede contar con librerías lo que permite hacer programas/diseños más complejos, sin entrar en detalles sobre cómo está implementación de estas librerías.

### 3.3.2 Ciclo de Desarrollo

Los componentes que hacen posibles el de desarrollo con dispositivos FPGA y que son utilizados actualmente, pueden resumirse en los siguientes puntos:

- El código HDL
  - Selección del HDL para realizar los diseños.
  - Convenciones y reglas del código para desarrollo en grupo y mayor reusabilidad.
  - Interconectividad de *cores*.
  - Extensiones y asistentes a lenguajes HDL.
  - Librerías HDL.
- Edición del código
  - Software de edición.
  - Asistentes del software de edición.
- Simulación
  - Herramienta de simulación.
  - Banco de pruebas.
  - Asistentes del banco de pruebas.
  - Visualización de formas de onda.
  - Asistentes para el proceso de simulación.
- Síntesis
  - Herramienta de síntesis.
- Transferencia al FPGA
  - Selección del dispositivo FPGA.
  - Hardware para grabación en la FPGA o memoria.
  - Software para grabación en la FPGA o memoria.

**Edición del Código**

No son muchos los editores libres que poseen facilidades avanzadas para la edición de código VHDL. Estas son algunas de las características que hacen a un editor una buena elección para el trabajo con VHDL:

- Alta flexibilidad en sintaxis para VHDL.
- Macros específicas con construcciones típicas de VHDL.
- Utilización de Exuberant C Tags ECTAGS con soporte específico para VHDL.

Para potenciar las capacidades del software de edición, existen aplicaciones externas que pueden ayudar en el trabajo de edición. Una de estas aplicaciones es CTAGS o Exuberant CTAGS que ayuda en la tarea de búsqueda e inspección dentro del código fuente soportando a Verilog pero no VHDL[18].

Otra herramienta útil es tpl2file que es una utilidad para extraer las plantillas de ejemplos VHDL que vienen con el ISE WebPack y colocarlas en archivos independientes dentro de una estructura de directorios.

Usualmente abreviada PPC, PowerPC es el nombre original de la arquitectura de computadoras de tipo RISC, es un procesador de 32 bits de un solo núcleo desarrollados principalmente por IBM, Motorola y Apple. El PowerPC ha sido una de las arquitecturas más extendidas gracias a su alto rendimiento y de gran implementación tecnológica.

Embedded Development Kit (EDK), es un paquete de solución de software para el diseño de sistemas de procesamiento embebido. Incluye la suite de herramientas Studio Plataform, así como toda la documentación y la propiedad intelectual que se necesita para el diseño en una plataforma con FPGA y un procesador PowerPC o MicroBlaze.

**Simulación**

Debido a su amplio uso y a la capacidad de compilar sin problemas proyectos, el sistema Modelsim dentro el WebPack de ISE; se considera como la herramienta principal de simulación para sistemas desarrollados en FPGA y que utiliza la tecnología del compilador GCC, el compilador de software libre más utilizado.

**Síntesis**

Luego de pasar por el proceso de simulación, es necesario realizar la síntesis del diseño para su posterior grabación en el FPGA o en la memoria de configuración. Los dos fabricantes más importantes de dispositivos FPGA brindan herramientas, cada uno para sus propios productos, de síntesis de uso gratuito.

La herramienta de síntesis ISE WebPack de Xilinx es de uso gratuito (no libre), que contiene una versión para sistemas GNU/Linux y permite su utilización con línea de comandos, lo que facilita la automatización del proceso con herramientas como GNU Make.

**Transferencia al FPGA**

Para desarrollar una aplicación, además del chip FPGA es necesaria una electrónica de soporte: Circuito impreso, circuitos de alimentación, memoria, conectores, etc. Aunque la forma más fácil es adquirir y trabajar con algún kit de desarrollo para FPGA, como el kit XtremeDSP.

Los dispositivos FPGA pueden poseer varios mecanismos para la programación de su configuración. Uno de los mecanismos que garantizan compatibilidad de hardware entre diferentes fabricantes y modelos, es el estándar IEEE 1149.1[19] del JTAG (Joint Test Action Group).

La transferencia de la configuración al hardware involucra dos tipos de conocimientos. Por un lado el protocolo de transferencia, que en este caso es el JTAG y por otro lado el conocimiento de las características específicas del dispositivo a usar. Herramientas como xilinx_jtag incluye ambos aspectos, pero se encuentra limitadas a dispositivos específicos.

# Capítulo 4

# DISEÑO E IMPLEMENTACIÓN

En esta sección se retoman e integran aspectos abordados en los capítulos anteriores. Se ha estudiado y analizado a todos los componentes que intervienen en la implementación y desarrollo del proyecto. El siguiente paso es la planeación y ejecución detallada, de la estructura con la que se integra el sistema. En la primera parte del capítulo se presenta una descripción del proyecto y la arquitectura utilizada, resaltando los elementos principales; después se presenta el desarrollo de los algoritmos de adquisición en MatLab y finalmente una revisión de los principales pasos para la implementación de los algoritmos en la plataforma de trabajo FPGA.

## 4.1 Arquitectura del Sistema Desarrollado

Como objetivo general se ha planteado el desarrollo de una plataforma de hardware basado en FPGA con características de radio definido por software, para realizar la tarea de adquisición de señales del sistema GPS. En el capítulo 2 se ha presentado la teoría detrás de los métodos de adquisición; sin embargo es necesario implementarlos y evaluarlos sobre diversos ambientes; para hacer evidentes las características particulares de cada uno de estos algoritmos.

A partir de seguir este camino se logrará extraer datos y resultados preliminares para trasladarlos a la etapa de seguimiento, que junto al cálculo de posición y a la adquisición forman la parte principal de procesamiento del receptor de GPS. Es necesario tomar una perspectiva general del proyecto, por lo que se debe describir a groso modo la estructura del receptor, así como de sus etapas.

La arquitectura de un radio definido por software es la mostrada en la Figura 4.1, en esta estructura podemos observar las etapas fundamentales del procesamiento de las señales bajo esta arquitectura.

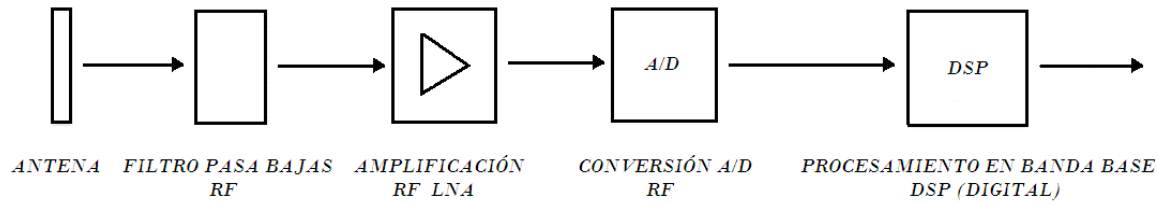



*Fig. 4.1 Estructura básica de un radio definido por software*

Se puede observar a una antena para recibir las señales, posteriormente se encuentra la etapa encargada de adaptar la señal, esta etapa se encuentra conformada por un filtro pasa bajas, un amplificador y un convertidor A/D, que proporciona una señal en banda base, adecuada para tratarse en la etapa de procesamiento.

Estos elementos se han descrito en la primera sección del trabajo, sin embargo es necesario resaltar que las plataformas de radio definido por software se desarrollan principalmente en la última etapa del diagrama mostrado, eso quiere decir que se implementa en el bloque de DSP.

La arquitectura del receptor mas apegada a la empleada para este trabajo es mostrada en la Figura 4.2, donde se observa a detalle con que debe contar el sistema y algunos de los parámetros de los dispositivos.

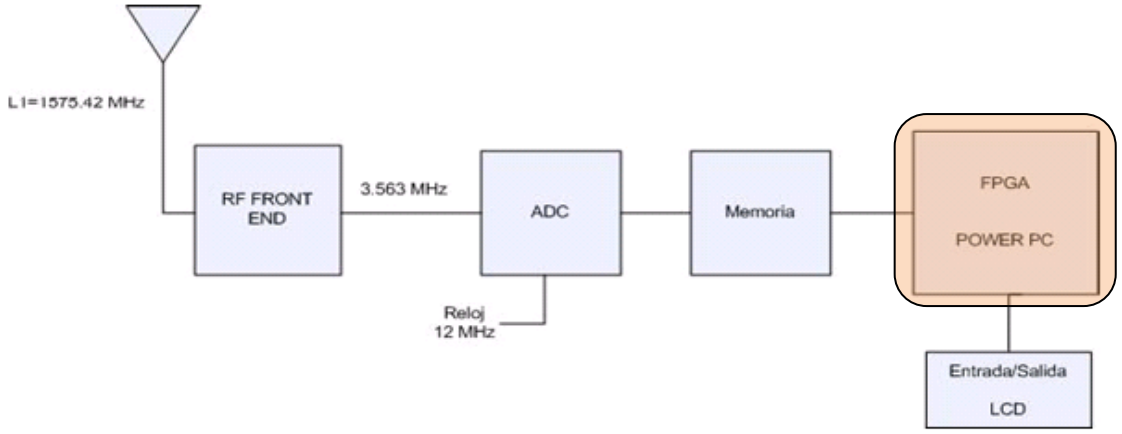



*Fig. 4.2 Estructura de un Receptor GPS basado en software*

La antena requiere de parámetros tales que permitan recibir señales dentro de la banda L, en la cual opera el sistema GPS. Los parámetros del convertidor y la terminal de entada de RF, deben ser precisos para obtener la señal deseada.

A la estructura se suma un bloque de memoria, que permite depositar una muestra de la señal de GPS, para el desarrollo preliminar de las primeras etapas; además de contener datos y resultados para la etapa de visualización.

Sobre el recuadro se identifica la sección dedicada al procesamiento de señales, que se lleva a cabo con el FPGA através del PowerPC, que es el indicado para desarrollar dicho procesamiento y en el cual se implementarán y evaluarán los algoritmos; realizado un análisis de desempeño dentro del dispositivo.

En el diagrama a bloques mostrado en la figura 4.3, podemos observar a detalle las etapas de procesamiento que sigue una señal GPS dentro del receptor. La primera etapa de procesamiento en banda base es la adquisición, que entra en función justo después de que la etapa de adaptación termina; esta es la encargada de obtener la lista y características de los satélites que están a la vista del receptor.

Posteriormente esta la etapa de seguimiento, la cual tiene como propósito mantener el contacto con los satélites obtenidos por la adquisición. Después de mantener esta comunicación, el siguiente paso es extraer los datos de navegación que se encuentran en las señales adquiridas, calculando los pseudorangos de los satélites a la vista, para finalmente realizar la triangulación y obtener la posición del receptor.

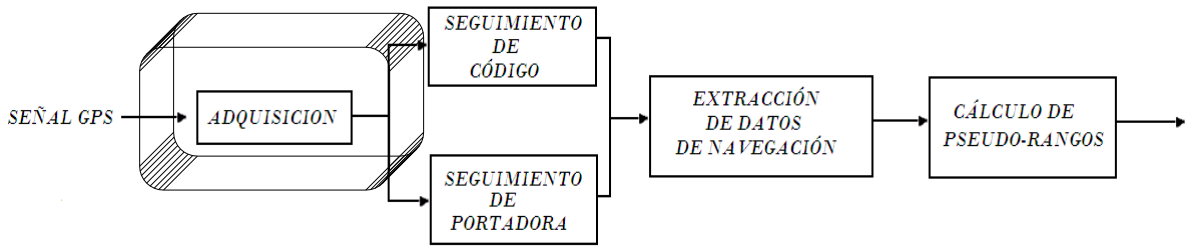



*Fig. 4.3 Módulos principales del procesamiento de señales en receptores GPS*

La propuesta de arquitectura general del receptor de señales GPS basado en FPGA es la mostrada en la figura 4.4. En esta se han considerado los elementos de hardware y software disponibles para la realización del proyecto. En primer lugar se dispone de una terminal de entrada de RF que además incorpora un convertidor ADC para entregar una señal de banda base muestreada.

Posteriormente se encuentra la etapa de desarrollo realizada en una PC y que está asociada a la captura de un archivo binario de la señal GPS procedente de la terminal de entrada de RF. Este archivo es procesado en MatLab para desarrollar los métodos de adquisición, con propósito de análisis y simulación, y finalmente trasladarlos a lenguaje C para su descarga sobre el FPGA.

El núcleo de procesamiento es desarrollado dentro del FPGA, que en conjunto con el PowerPC se encargaran de desarrollar las etapas de adquisición, seguimiento y cálculo de posición. Aunque cabe aclarar, que este trabajo aborda únicamente los algoritmos de adquisición de señales.

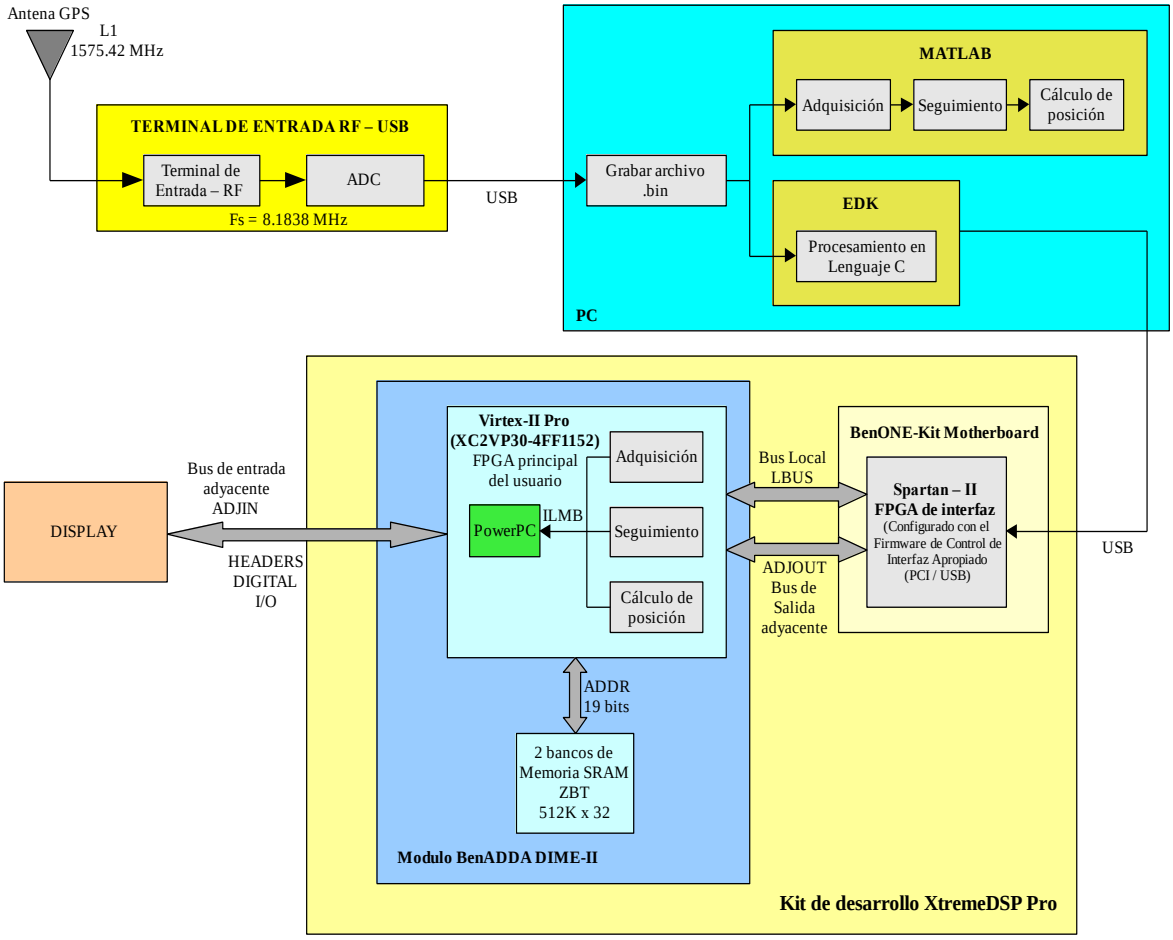



*Figura 4.4 Estructura de un Receptor GPS basado en Software*

Para las etapas que son desarrolladas se requiere de la información real del sistema GPS, esta es obtenida a través de la antena y la terminal de entada de RF.

### 4.1.1 Terminal de Entrada de RF

Originalmente para la etapa de programación y con el propósito de verificar el funcionamiento de los algoritmos desarrollados se utilizó un archivo de datos binario que contiene la información de la señal GPS L1 en banda base. Este archivo está disponible en línea en el sitio web de la universidad de Colorado en Estados Unidos y es ofrecido a desarrolladores de aplicaciones GPS. El archivo contiene aproximadamente 1 segundo de la señal L1 muestreada y digitalizada.

Posteriormente durante el desarrollo del proyecto se adquirió una terminal de entrada de RF (Front–End RF) para la señal GPS L1. Este dispositivo es conectado directamente a una antena GPS y entrega a la salida muestras de la señal L1 en banda base. El formato de la información entregada por el dispositivo es el mismo que el archivo de datos usado previamente lo que facilitó su uso dentro del sistema ya desarrollado. En la figura 4.5 se muestra una imagen del dispositivo conectado a la antena GPS y a una computadora de escritorio, además un diagrama a bloques con los parámetros de funcionamiento de esta terminal.


*Fig. 4.5 Implementación del sistema para una toma de muestra de la señal del sistema GPS*

Las características de los dos dispositivos utilizados para tomar la muestra de la señal GPS son:

- Antena GPS.- La antena es diseñada para inducir un voltaje de la propagación de ondas de radio en la frecuencia L1 o 1575.42 MHz. Además tiene la capacidad de trabajar en un ancho de banda apropiado para la señal deseada. Esto se hace usualmente con dos parámetros adicionales en la antena: la Razón de Voltaje de Onda Estacionaria (VSWR – *Voltage Standing Wave Ratio*), que es una medida de desacoplamiento de impedancia o cuanta potencia incidente será absorbida y que tanto será reflejada, usualmente este parámetro esta en el orden de 2:1, lo que equivale al 90% de absorción de energía en todo el ancho de banda de la frecuencia deseada. Como segundo parámetro tenemos a la impedancia que típicamente es de 50Ω.
  Las características completas de la antena son*:* Ganancia de 26 dB, VSWR < 2.0, Voltaje: 3.3 V +/- 0.5 V, Corriente: 12 mA, Peso: 50 gr, 5 metros de cable

- Terminal de entrada RF – USB (GN3S).- El GN3S es diseñado para capturar directamente los datos de la señal de bajo nivel (muestras de frecuencia intermedia) que serán entregados por la red de satélites GPS y que serán procesados por la terminal de entrada de RF SiGe. estos datos capturados serán procesados en MatLab y lenguaje C. El programa de captura desarrollado para la plataforma de Windows XP tiene un límite de 600 MB (o 38.4 segundos). Esto significa que se puede capturar un registro completo de GPS. La arquitectura propuesta utiliza la segunda versión del GN3S (GN3S v2.0) la cual está construida sobre el SiGe 4120 GPS ASIC. Este dispositivo proporciona un flujo de datos con una frecuencia de muestreo baja (8.1838 MHz) y un par de muestras I/Q.

  Los parámetros específicos de los datos capturados por este modulo son los siguientes: Frecuencia de muestreo: 8.1838 MHz, Frecuencia intermedia: 38.400 kHz, 2 bits de muestras I/Q (1 bit para I y 1 bit para Q) en formato binario schar.

A partir de ahora y con la muestra de señal GPS obtenida, comienza el desarrollo de la adquisición en el procesamiento de la señal; con la evaluación de los métodos estudiados para obtener estas señales.

## 4.2 Implementación en MatLab de los Algoritmos de Adquisición

Como paso inicial en el análisis de los algoritmos, se comienza en el ambiente de simulación de MatLab, con este lenguaje y su alto rendimiento, se proporcionan las bases de desarrollo de los algoritmos, para posteriormente ser adaptados al ambiente de tiempo real.
Las propiedades teóricas y el desempeño de los diferentes métodos han sido descritos en el capítulo 2, por lo que esta sección se enfoca en presentar solo los aspectos más relevantes desde el punto de vista de programación en MatLab, para el desarrollo de los algoritmos de adquisición.

### 4.2.1 Algoritmo de Adquisición por Búsqueda Serial

El primer algoritmo establecido para la adquisición es el de búsqueda serial. Sus características han sido estudiadas por lo que ahora se hará la descripción de su funcionamiento en MatLab.

La implementación del método por búsqueda serial se desarrolla siguiendo paso a paso el diagrama a bloques ya analizado y mostrado en la Figura 4.6.

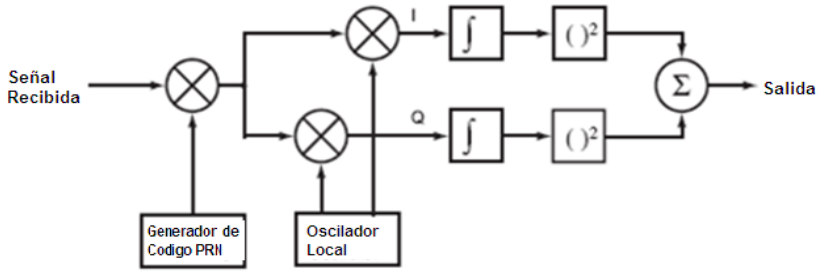



*Fig. 4.6 Diagrama a Bloques de la Búsqueda Serial*

Cuando las 32 secuencias PRN son obtenidas con el generador local, todas las posibles secuencias originadas por los satélites, pueden ser generadas.
Estas secuencias pueden ser obtenidas a través del registro de corrimiento mostrado en la figura 2.12, que desarrollado en el lenguaje de C/MatLab.se puede observar como:

```
for i=1:1023
        g1(i)= reg(10);
        acarr= reg(3)*reg(10); %%%  Registro de corrimiento para generar código PRN
        reg(2:10)= reg(1:9);
        reg(1)= acarr;
end
```

Con estas líneas de código obtenemos las diferentes secuencias PRN para realizar la comparación con respecto a la señal entrante y obtener el primer valor de similitud, con una función de correlación

```
CodCA = xcorr(senal1,TablaCodCA(PRN, :));
```

El segundo paso después de multiplicar la señal recibida con el código PRN es el de multiplicar a la señal de entrada, con una señal portadora localmente generada. El diagrama muestra un generador de portadora con una fase diferida de 90°, correspondiendo al seno y coseno de la señal.

La secuencia de muestras necesarias, correspondiente a 1ms se obtiene como:

```
carrier = exp(j*2*pi*fc*ts*nn);
```

De la cual los datos del seno y el coseno son obtenidos a partir de:

```
cosine = real(carrier);
sine = imag(carrier);
```

En la parte final de la búsqueda serial encontramos una integración y una cuadratura de los resultados de la multiplicación con el coseno y el seno de las señales respectivas. Esta integración es simplemente una suma de todos los puntos correspondientes al largo de los datos procesados, esta operación se realiza a ambas componentes de la señal y como paso final se suman los valores de I y Q.

```
sinCarr = sin(PasoFrec(IndicePasos) * puntosFase);
cosCarr = cos(PasoFrec(IndicePasos) * puntosFase);
I1= xcorr(sinCarr.*CodCA);
Q1= xcorr(cosCarr.*CodCA);
Res=(I1.^2)+(Q2.^2)
```

Si el código local generado es alineado con el código de la señal entrante y la portadora generada coincide con la frecuencia de la señal recibida, la salida reflejará un valor alto que es sometido al criterio umbral que se haya establecido como decisión para la adquisición de la señal.

```
[Tamano_Pico Indice_Inter_Frec] = max(max(resultados, [], 2));
[Tamano_Pico Cod_Fase] = max(max(resultados));

if (Tamano_Pico/Segundo_Pico) > Umbral_Adq
xCarrier = Senal0DC(CodigoFase:(CodigoFase + 10*Muestras_por_Cod-1)).* longCodCa ;
```

### 4.2.2 Algoritmo de Adquisición por Búsqueda Paralela en el Espacio de la Frecuencia

De la misma forma que en el método de búsqueda serial, la implementación del algoritmo de búsqueda paralela en el espacio de la frecuencia, se consigue después de seguir los bloques del diagrama mostrado en la figura 4.7.

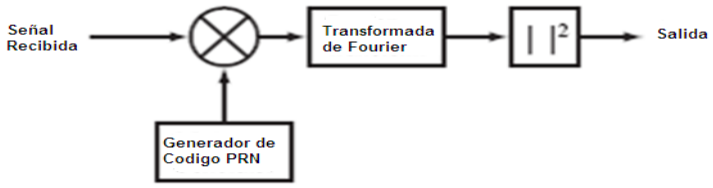



*Fig. 4.7 Diagrama a Bloques del Método en el Espacio de la Frecuencia*

La primer parte de este método corresponde a la misma descrita en el método anterior. Esto es, un generador local de códigos PRN, genera una réplica de alguno de los códigos PRN posibles, el cual es multiplicado por el contenido en la señal recibida.

Lo novedoso y que da nombre al método, es el paso en el que después de la multiplicación con la señal de entrada, la señal resultante es transformada al dominio de la frecuencia a través de una transformada de Fourier.
El resultado de esta transformada arroja valores precisos de comparación, para asumir si un satélite es adquirido o no. Sin perder de vista al valor de umbral establecido.

Desarrollar la función de transformada de Fourier en el ambiente de MatLab no es difícil, se requiere el uso de la función FFT aplicada directamente a la señal y que corresponde a una transformada de orden 2[34]. Asumiendo que cuando la señal es trasformada al dominio de la frecuencia se convierte en una señal compleja.

*CodCAdomFec = fft(TablaCodCA(PRN, :))*

### 4.2.3 Algoritmo de Adquisición por Búsqueda Paralela en el Espacio del Código de Fase

Al final de los tres métodos de adquisición se presenta el algoritmo de búsqueda paralela en el espacio de la de código que en el mismo sentido que los métodos anteriores, la forma conveniente de desarrollar el algoritmo, es seguir paso a paso el trabajo que señala el diagrama a bloques establecido para éste, mostrado en la fig. 4.8.

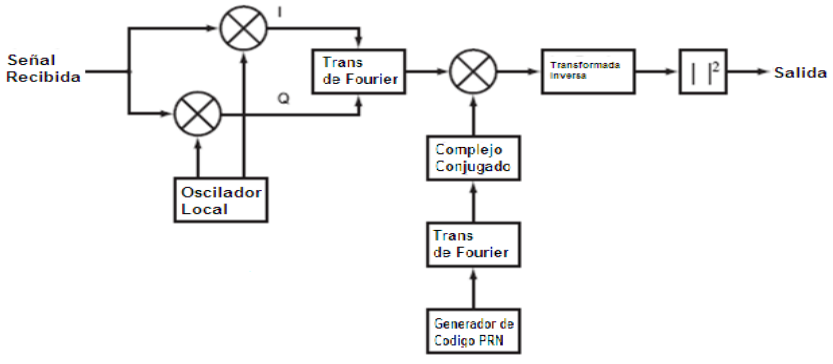



*Fig. 4.8 Diagrama a Bloques del Método en el Espacio del Código de Fase*
Para este método se necesita de un mayor procesamiento de señal, como puede reflejarse en la figura 4.8, encontramos nuevos bloques comparados a los desarrollados anteriormente.

Como resultado de esto, solo algunos de los elementos son reutilizados para su implementación. Una diferencia en este método es que solo se utiliza un código PRN por cada adquisición. Por lo que no es necesario generar ni desarrollar la etapa de reproducción de todos estos códigos.

Aquí el primer paso es la multiplicación de la señal de entrada con una localmente generada dividida es dos componentes un seno y un coseno, para obtener los componentes I y Q de la señal, estas dos componentes resultantes son combinadas para generar la entrada compleja a la primera transformada de Fourier.

```
IQDomfrec1= fft(I1 + j*Q1);
```

El resultado de la transformada es multiplicado por el resultado del la rama que aparece en la parte baja del diagrama y se desarrolla de la siguiente manera: el primer paso es generar un código PRN como en los algoritmos anteriores, apoyado en el registro de corrimiento.

Este código es pasado a través de la transformada de Fourier al dominio de la frecuencia y el resultado se toma como el complejo conjugado, combinando ambas señales.

```
convCodIQ1 = IQDomfrec1 .* CodCAdomFec;
CodCAdomFec = conj(fft(TablaCodCA(PRN, :)));
```

El resultado es sometido a la trasformada inversa de Fourier y pasado al dominio del tiempo, facilitándose su trabajo con la función IFFT de MatLab.

```
Res = abs(ifft(convCodIQ1)) .^ 2;
```

Lo arrojado por esta transformada inversa puede ser sometido al criterio de umbral establecido, y si los valores satisfacen a los deseados, se logra adquirir al satélite específico.

## 4.3 Desarrollo en la plataforma FPGA

### 4.3.1 Plataforma de Trabajo

Xilinx con el desarrollo del Virtex-II, dispone del kit del desarrollo XtremeDSP, que en conjunto con Nallatech, han integrado una plataforma de trabajo muy eficiente, esta plataforma es de gran alcance para el desarrollo de trabajos con procesamiento digital de señales, específicamente para los usos de alto rendimiento, tales como radio definida por software, el establecimiento de una red, la radio HDTV, 3G y procesos de video.

Este kit permite diseñar sistemas de alto rendimiento con algoritmos exigentes del procesamiento de señales y ejecución de funciones de control, usando el procesador integrado PPC405 de IBM.

El kit XtremeDSP de desarrollo con Virtex-IIPRO(fig. 4.9), combina sus ventajas en software con el generador v6.3 del sistema de Xilinx y ofrecer un marco completo de diseño para FPGA y DSP, permitiendo a los sistemas parciales o enteros funcionar en hardware rápida y eficientemente dentro de la plataforma.
Este enfoque redefine el tiempo de desarrollo, permitiendo que los diseñadores produzcan sistemas avanzados de DSP con toda la flexibilidad del diseño, y capacidades de reconfiguración de un FPGA.


*Fig. 4.9 XtremeDSP Development Kit*

Con las ventajas y características de esta plataforma se favorece la elaboración del trabajo, para los requerimientos en implementación y evaluación de los algoritmos de adquisición dentro de un ambiente de tiempo real.

Con el desarrollo de los algoritmos en la etapa de simulación, se da el siguiente paso para desarrollar y trabajar el sistema en el ambiente hardware. La metodología de diseño es similar a la cualquier sistema digital, salvo que al final obtenemos un archivo ejecutable que se lo podemos descargar al FPGA para que se reconfigure, implementando así el diseño. El enfoque se da sobre la programación e implementación de algoritmos de adquisición sobre el FPGA es mostrado en la estructura de la figura 4.10.

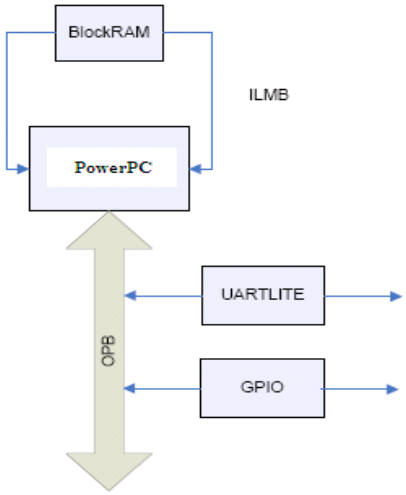



*Fig. 4.10 Estructura de desarrollo dentro del FPGA*

Inicialmente se observa al microprocesador PowerPC (PPC) encargado de contener al software de la aplicación, constituyendo la inteligencia del dispositivo. Además se debe contar con un bloque de memoria suficiente para almacenar al programa que se descarga al FPGA, así como los datos y resultados para las demás etapas desarrolladas en este bloque.
ILMB es el bus que logra interactúen el bloque de memoria y el PPC; otro bus requerido es el de OPB que nos conecta al microcontrolador con el FPGA o algún otro registro de entrada o salida.

Básicamente esta es la estructura en hardware que se desarrolla para la implementación de los algoritmos dentro de éste ambiente, donde son programados y especificados con el archivo binario, que se descargara al FPGA y que contiene la información necesaria en su ejecución.

### 4.3.2 Paso a lenguaje C

Para integrar los algoritmos de adquisición al PPC405, es necesario obtener la información del microprocesador y documentar acerca de este dispositivo.

La figura 4.11 muestra una parte del manual de este dispositivo con referencia a su desempeño.

## PowerPC 405 Performance

The PowerPC 405 processor executes instructions at sustained speeds approaching one cycle per instruction. Table 1-3 lists the typical execution speed (in processor cycles) of the instruction classes supported by the PowerPC 405 processor.

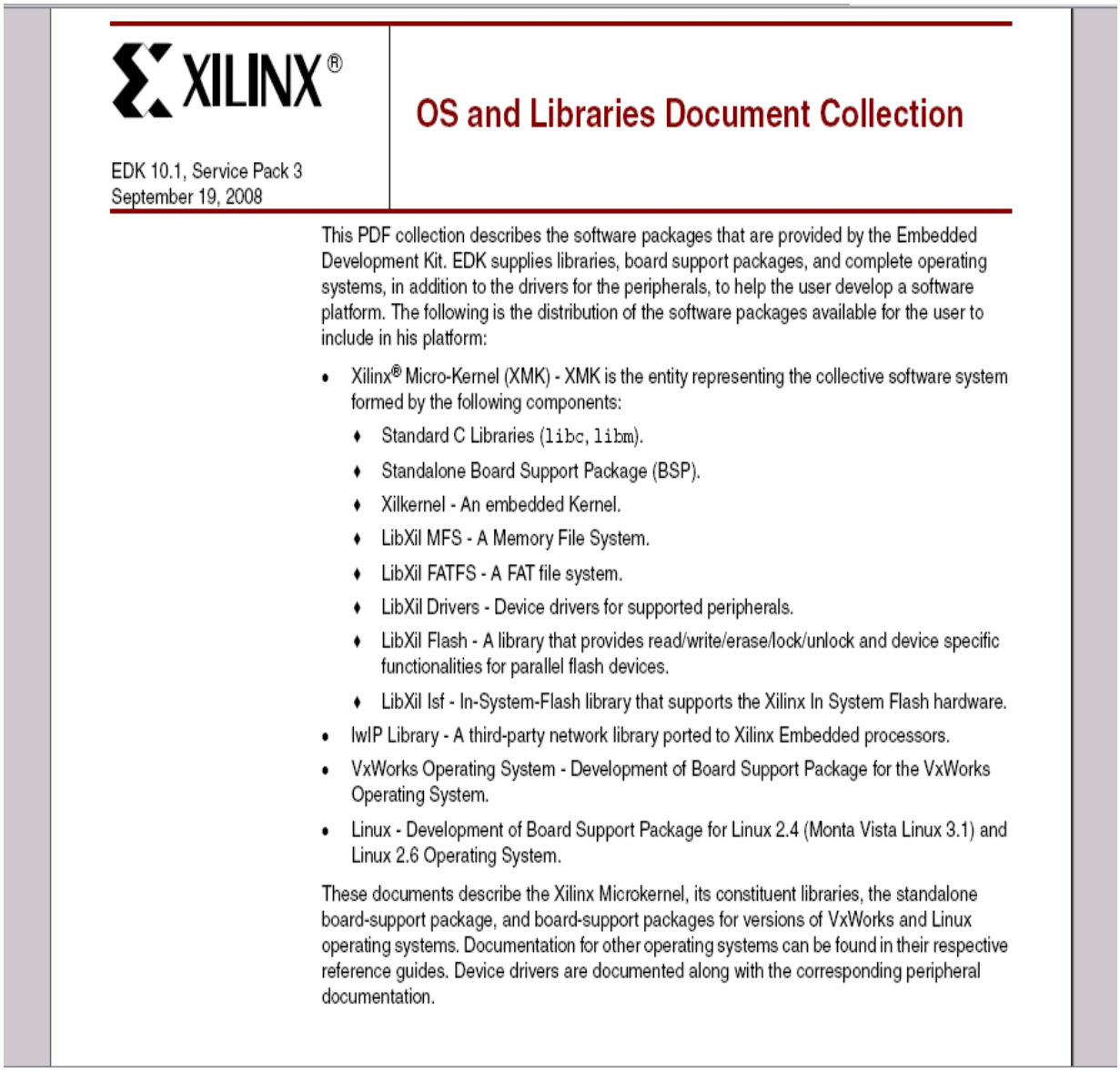
Instructions that access memory (loads and stores) consider only the "first order" effects of cache misses. The performance penalty associated with a cache miss involves a number of second-order effects. This includes PLB contention between the instruction and data caches and the time associated with performing cache-line fills and flushes. Unless stated otherwise, the number of cycles described applies to systems having zero-wait-state memory access.

*Table 1-3:* **PowerPC 405 Cycles per Instruction**

| Instruction Class | Execution Cycles |
|---|---|
| Arithmetic | 1 |
| Trap | 2 |
| Logical | 1 |
| Shift and Rotate | 1 |
| Multiply (32-bit, 48-bit, 64-bit results, respectively) | 1, 2, 4 |
| Multiply Accumulate | 1 |
| Divide | 35 |
| Load | 1 |
| Load Multiple and Load String (cache hit) | 1 per data transfer |
| Store | 1 |
| Store Multiple and Store String (cache hit or miss) | 1 per data transfer |
| Move to/from device-control register | 3 |
| Move to/from special-purpose register | 1 |
| Branch known taken | 1 or 2 |
| Branch known not taken | 1 |
| Predicted taken branch | 1 or 2 |
| Predicted not-taken branch | 1 |
| Mispredicted branch | 2 or 3 |

*Figura 4.11 Información obtenida de la pagina de Xilinx del PPC405*

Dentro de la información recabada se encuentra el documento de paqueterías y sistemas operativos compatibles y soportados por el EDK a través de su compilador GCC, el cual es la base para el desarrollo e implementación de aplicaciones en software dentro del FPGA. La Figura 4.12 muestra la pagina inicial de este documento.



XILINX®

OS and Libraries Document Collection

EDK 10.1, Service Pack 3
September 19, 2008

This PDF collection describes the software packages that are provided by the Embedded Development Kit. EDK supplies libraries, board support packages, and complete operating systems, in addition to the drivers for the peripherals, to help the user develop a software platform. The following is the distribution of the software packages available for the user to include in his platform:

- Xilinx® Micro-Kernel (XMK) - XMK is the entity representing the collective software system formed by the following components:
  - Standard C Libraries (`libc`, `libm`).
  - Standalone Board Support Package (BSP).
  - Xilkernel - An embedded Kernel.
  - LibXil MFS - A Memory File System.
  - LibXil FATFS - A FAT file system.
  - LibXil Drivers - Device drivers for supported peripherals.
  - LibXil Flash - A library that provides read/write/erase/lock/unlock and device specific functionalities for parallel flash devices.
  - LibXil Isf - In-System-Flash library that supports the Xilinx In System Flash hardware.
- lwIP Library - A third-party network library ported to Xilinx Embedded processors.
- VxWorks Operating System - Development of Board Support Package for the VxWorks Operating System.
- Linux - Development of Board Support Package for Linux 2.4 (Monta Vista Linux 3.1) and Linux 2.6 Operating System.

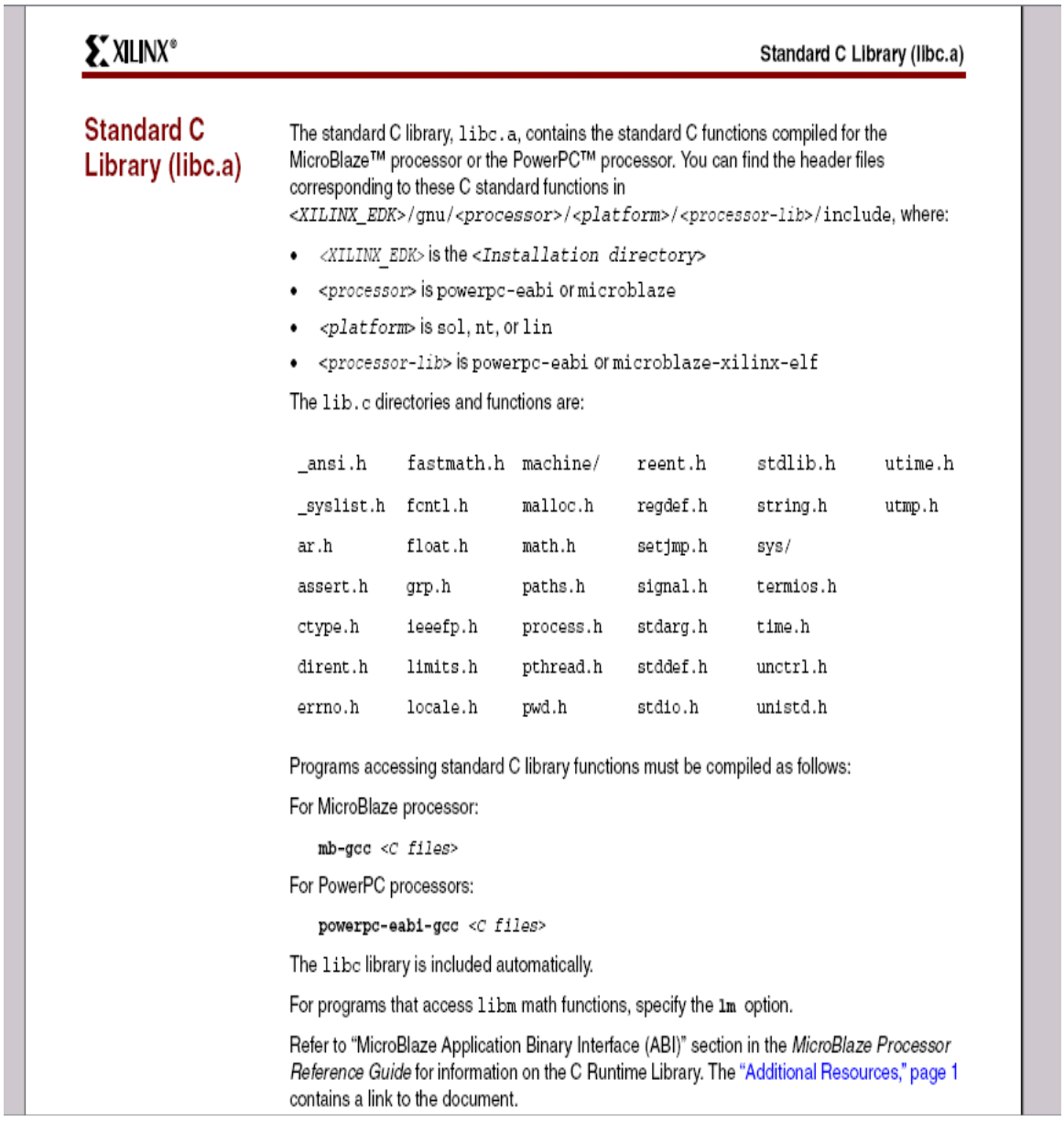
These documents describe the Xilinx Microkernel, its constituent libraries, the standalone board-support package, and board-support packages for versions of VxWorks and Linux operating systems. Documentation for other operating systems can be found in their respective reference guides. Device drivers are documented along with the corresponding peripheral documentation.

*Fig. 4.12 Colección de librerías y sistemas operativos compatibles con el sistema de desarrollo EDK.*

Como librerías de apoyo para el desarrollo de sistemas con software, encontramos las librerías estándar de lenguaje C, esto completa la perspectiva para la adición del sistema operativo en la aplicación.

Los algoritmos de adquisición de señales GPS, han sido desarrollados en MatLab; sin embargo ahora se requiere que el código sea adaptado para ser usado en el ambiente del FPGA, es decir llevar este código a un lenguaje de bajo nivel, en particular en lenguaje C, que es uno de los lenguajes de programación soportados por las herramientas de desarrollo para sistemas embebidos.
El desarrollo de los algoritmos en el lenguaje de MatLab utiliza funciones especializadas para el procesamiento de las señales, sin embargo el trabajo desarrollado en el PPC del FPGA, requiere que estas funciones sean desarrolladas a su nivel nativo dentro de las funciones de C.

Para reconocer el punto de aceptación de funciones de C en el EDK, se hace referencia al documento de la figura 4.12 para identificar la información referente a estas librerías. La figura 4.13 muestra la página del documento con la lista de funciones aceptadas dentro de este ambiente.



XILINX®

Standard C Library (libc.a)

## Standard C Library (libc.a)

The standard C library, `libc.a`, contains the standard C functions compiled for the MicroBlaze™ processor or the PowerPC™ processor. You can find the header files corresponding to these C standard functions in *<XILINX_EDK>*/gnu/*<processor>*/*<platform>*/*<processor-lib>*/include, where:

- *<XILINX_EDK>* is the *<Installation directory>*
- *<processor>* is powerpc-eabi or microblaze
- *<platform>* is sol, nt, or lin
- *<processor-lib>* is powerpc-eabi or microblaze-xilinx-elf

The `lib.c` directories and functions are:

| | | | | | |
|---|---|---|---|---|---|
| _ansi.h | fastmath.h | machine/ | reent.h | stdlib.h | utime.h |
| _syslist.h | fcntl.h | malloc.h | regdef.h | string.h | utmp.h |
| ar.h | float.h | math.h | setjmp.h | sys/ | |
| assert.h | grp.h | paths.h | signal.h | termios.h | |
| ctype.h | ieeefp.h | process.h | stdarg.h | time.h | |
| dirent.h | limits.h | pthread.h | stddef.h | unctrl.h | |
| errno.h | locale.h | pwd.h | stdio.h | unistd.h | |

Programs accessing standard C library functions must be compiled as follows:

For MicroBlaze processor:

```
mb-gcc <C files>
```

For PowerPC processors:

```
powerpc-eabi-gcc <C files>
```

The `libc` library is included automatically.

For programs that access `libm` math functions, specify the `-lm` option.

Refer to "MicroBlaze Application Binary Interface (ABI)" section in the *MicroBlaze Processor Reference Guide* for information on the C Runtime Library. The "Additional Resources," page 1 contains a link to the document.

*Figura 4.13 Lista de las librerías de lenguaje C, compatibles con el ambiente EDK.*

Ahora las funciones de MatLab deben ser desarrolladas e implementadas bajo el rigor y simplicidad del lenguaje C y avaladas por el compilador GCC. Para conocer más sobre las instrucciones integradas a cada librería se puede recurrir al mismo documento que se encuentra en la página oficial del dispositivo.
El paso de lenguaje C/MatLab a C, es un trabajo arduo pues consiste en el traspaso de funciones de un lenguaje de alto nivel a uno con restricciones que el compilador usado impone.

En la figura 4.14 se muestra una comparación de las ventanas de los dos diferentes ambientes en que se han desarrollado los algoritmos de adquisición.

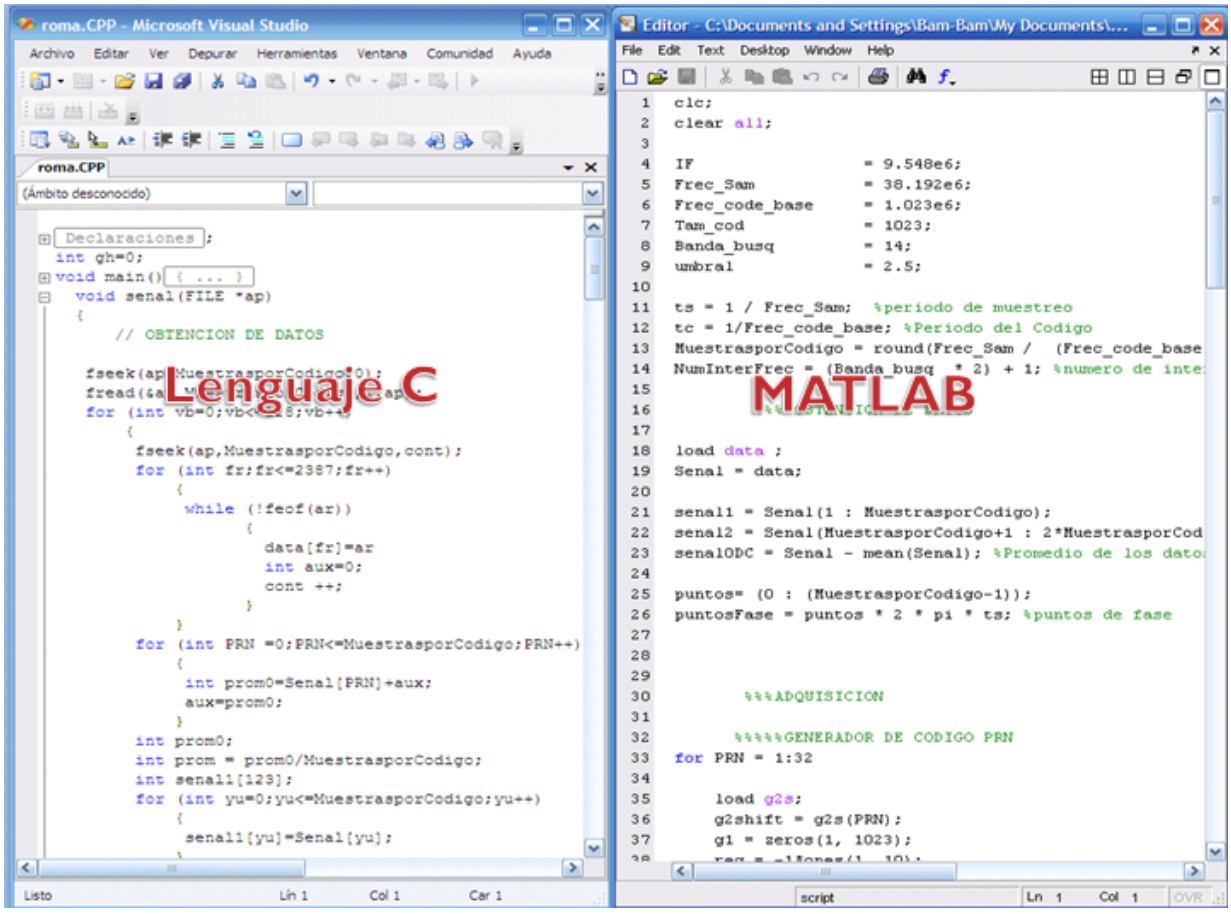



*Fig. 4.14 Ventana que muestra la comparación entre funciones de MatLab y su desarrollo en Lenguaje C.*

En esta figura solo se muestra una parte de código de ambas aplicaciones; sin embargo el traspaso de ambiente implica una complejidad muy alta, así como disposición de recursos en materia de desarrollo. Solo las instrucciones más simples como ciclos y operaciones básicas pueden ser utilizadas en la implementación en lenguaje C de los algoritmos.

Parte del trabajo de traspaso requiere de una adaptación de funciones a nivel de formatos en los datos, tal cual son trabajados en MatLab, estos necesitan ser modificados de acuerdo a las restricciones indicadas.

Por ejemplo, se puede hacer referencia al uso de datos complejos en los algoritmos y dado que este no es un formato soportado por el compilador, estas operaciones requieren del uso de operaciones discretas en las señales y que deben satisfacer las tareas para no perder el enfoque que debe contener la aplicación.

Solo una pequeña parte de código es reutilizado, pues algunas de las funciones en C/MatLab son similares a las usadas en C. La mayoría son reajustadas según los requerimientos del ambiente en general. Las funciones de asignación de memoria y de almacenamiento, solo se distinguen en el formato de redacción, sin perder de vista las restricciones que C impone a estas operaciones.

Algunas de las novedades y complejidades que se presentan durante el paso de lenguaje, pueden ser identificadas, por ejemplo como referencia se puede tomar al número de líneas de código invertidas para una aplicación en cada uno de los ambientes.

Mientras que en MatLab, para el desarrollo de uno de los algoritmos tenemos 135 líneas en C después de desarrollar las funciones, obtenemos 256 líneas de código. Esto no es de extrañar ya que si en MatLab se utiliza solo una línea para el uso de la trasformada de Fourier y su inversa, en C debemos recurrir a cerca de 10 líneas de código, o simplemente para los registros de corrimiento debemos hacer un aumento de 5 o más líneas.

Con esto se hace evidente que para obtener los resultados obtenidos en el primer ambiente se requiera de un aumento de más o menos 120 líneas de código. Sin mencionar que la comparación se hace sobre el algoritmo de menor complejidad implementado; y como se hizo notar en la parte teórica esta complejidad aumenta según se avanza a través de los algoritmos y es reflejado en esta diferencia de líneas de código, aumentando el tiempo de desarrollo.

### 4.3.3 Implementación en el PowerPC

Una vez obtenidos y desarrollados los algoritmos en C, podemos continuar con la implementación en tiempo real. Se dijo que el dispositivo encargado del procesamiento de los algoritmos es el PPC y que éste es compatible con el lenguaje nativo de C.

Con todos estos elementos disponibles, se debe pasar al diseño del archivo encargado de configurar al FPGA y enfocado al trabajo del PPC. Para obtener el resultado que desempeñe las tareas del proyecto, se debe conjuntar las dos partes más importantes en la programación y uso de los FPGA.

Primero, se debe configurar el interior del FPGA, el hardware necesario para desarrollar la aplicación apoyándose del ambiente de trabajo gráfico con las herramientas que Xilinx ofrece y en caso de que se requiera el uso de algún lenguaje de descripción de hardware, esa sección será implementada bajo los protocolos del VHDL. Para esto es indispensable reconocer los bloques adecuados que deben ser adaptados a esta etapa de hardware o los que deben ser desarrollados únicamente como software.

La adaptación de hardware es facilitada dada la gran versatilidad del FPGA por lo que la mayor parte de la aplicación va referida al desarrollo del software, sin embargo no se debe perder de vista lo relevante de esta etapa y de los componentes necesarios para desarrollar al software.

Para la segunda parte y más importante en esta aplicación, es la implementación del software, aquí se deben identificar las herramientas y pasos para el desarrollo de la etapa.

En la figura 4.15 se pueden observar las secciones en que se divide el desarrollo de una aplicación con software dentro del PPC y utilizando el EDK. En la parte izquierda, dentro del cuadro con mayor grosor, se encierra la etapa encargada de preparar al sistema a nivel de hardware. Esta sección es separada en dos secciones más.

En la primera se encuentra el trabajo del XPS (*Xilinx Platform Studio*) en hardware y esta encarga de dar de alta a todos los periféricos y dispositivos que serán utilizados en el sistema, esta etapa resulta en un archivo con extensión .ngc.

En la segunda parte tenemos al archivo con extensión .ucf, éste contiene la información en VHDL de restricción, esto es necesario debido a que de igual forma en que se dan de alta a los elementos que integran al sistema, se requiere especificar que terminales o dispositivos no deben o no pueden estar un funcionamiento.

En la última etapa de esta sección son unidos y sintetizados ambos archivos através del ambiente ISE o el mismo EDK, que al final obtiene un primer archivo bitstream de extensión .bit que contiene la información global en hardware, tanto de los dispositivos a utilizar así como de los que están restringidos para la aplicación.

En la parte derecha de la figura se encuentra la segunda etapa del trabajo en FPGA y es la referente al sistema operativo, aquí se implementa al software de la aplicación. Dentro de la sección identificada por el recuadro en línea punteada encontramos la etapa desarrollada a través de la herramienta SDK, en la que se evalúa e integra el software.

El primer paso es la compilación del programa desarrollado en C, con el GCC, de donde se obtiene el archivo objeto del programa, que será ligado con una segunda etapa. En esta segunda parte se encuentra el trabajo del XPS a nivel de software, éste proporciona la compatibilidad con las librerías estándar de C y consiguen un archivo con extensión .elf que contiene al sistema operativo listo para ser sumado al sistema.

Finalmente con el bloque identificado como Data2MEM, se realiza una nueva síntesis, integrados los archivos que se han generado, esta acción resultara en un archivo .bit final con las generalidades en software y hardware requeridas de la aplicación e implementación de los algoritmos de adquisición.

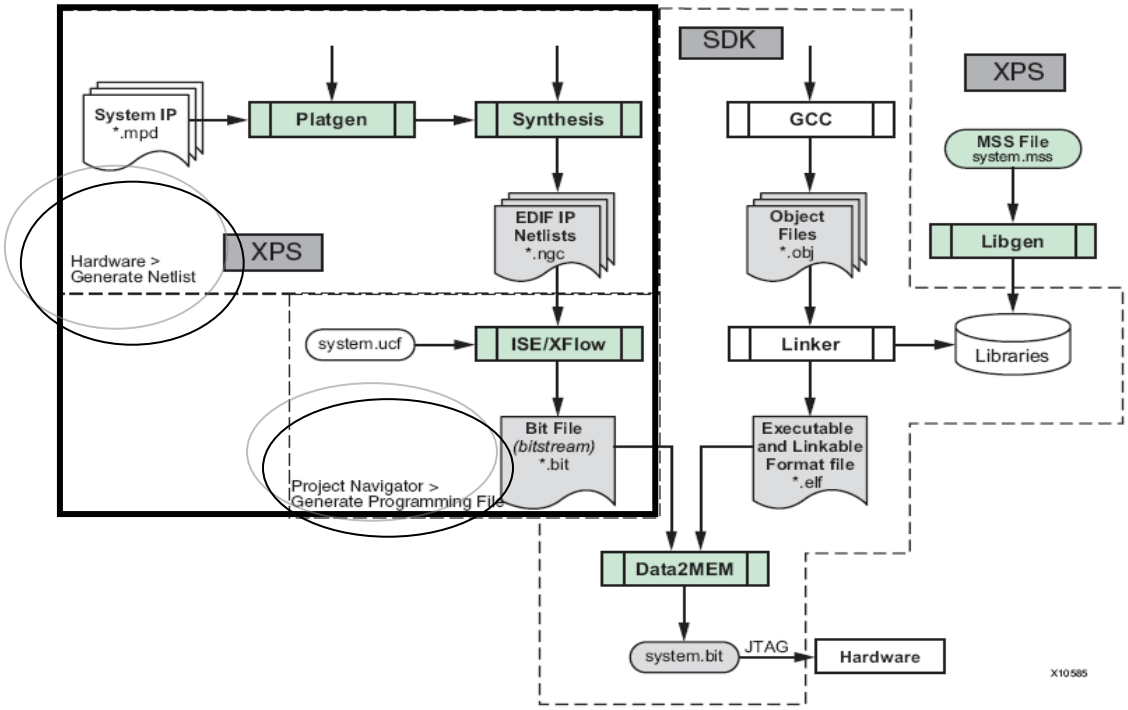



*Fig. 4.15 Diagrama a bloques del desarrollo de una aplicación utilizando un PowerPC405.*

Con todos los parámetros dispuestos se requiere utilizar la primera herramienta de Xilinx, el Embedded Development Kit (EDK), que permite ligar al código elaborado con la plataforma del PPC en el FPGA, obteniendo la aplicación sobre el dispositivo.

Dentro de la herramienta EDK, se encuentra la plataforma de SDK (Software Development Kit) que complementa al ambiente de trabajo para las aplicaciones en software, y que son las especializadas para elaborar estas aplicaciones en sistemas embebidos.

Como comentario es necesario remarcar que dentro de una estructura como esta se pueden desarrollar aplicaciones sobre dos diferentes Microprocesadores; Microblaze y PowerPC. Sin embargo en el kit de desarrollo XtremeDSP, solo puede ser implementado el trabajo sobre un PowerPC405, que brinda las ventajas suficientes para la elaboración del proyecto.

No se debe olvidar que la finalidad de desarrollar este sistema, es implementar y evaluar el desempeño de los algoritmos de adquisición, para monitorear el funcionamiento del proyecto se debe trabajar con el ambiente de ISE de Xilinx, que es la base de funcionamiento de las herramientas complementarias dentro del EDK.

Con la descripción anterior se desarrollan a los tres algoritmos que deben ser implementados en el PowerPC del FPGA.

### 4.3.4 Desarrollo del Sistema Embebido

Para continuar con la evaluación de los algoritmos debemos trabajar sobre la estructura descrita anteriormente.

La herramienta complementaria para diseñar el sistema que se está describiendo sobre PowerPC es el *Xilinx Platform Studio*.

Así que el trabajo debe desarrollarse sobre el ambiente de esta herramienta y, por lo que se debe describir el funcionamiento de esta aplicación.

Una vez que se arranca la herramienta (fig. 4.16) debemos tomar en cuenta algunas medidas, lo primero que se preguntara es si se quiere crear un proyecto nuevo, o si se quiere abrir un proyecto existente. Se iniciara el trabajo desde un proyecto nuevo.

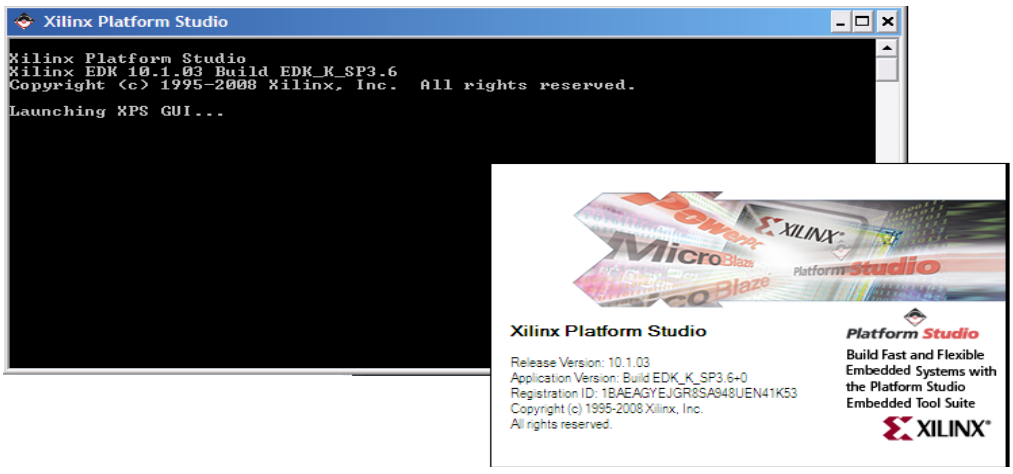



*Figura 4.16 Inicio de la herramienta Xilinx Plataform Studio*

Para iniciar un proyecto nuevo se ofrecen tres opciones:

- La primera es utilizar el *Base System Builder Wizard*, que es el asistente para crear sistemas en algunos modelos de placas de determinados fabricantes, o para un modelo genérico de placa.

- La opción *Blank XPS Project*, permite iniciar un proyecto desde cero, sin utilizar ningún asistente.

- Por último *Open a Recent Project* permite abrir un proyecto ya existente.

Para este caso es necesario iniciar el proyecto desde cero pues no existe en el asistente, la aplicación del kit XtremeDSP. Esta acción es mostrada en la figura 4.17

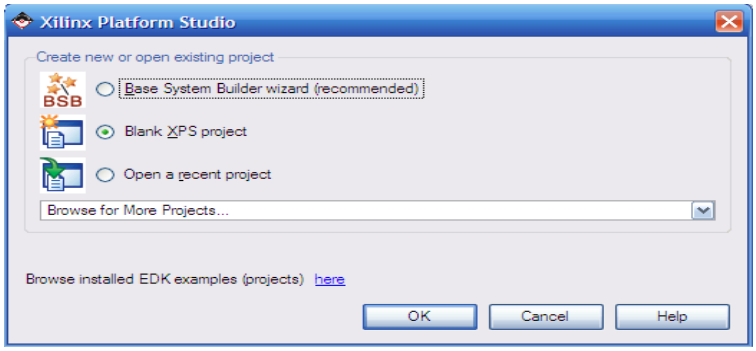



*Figura 4.17 Inicio de un proyecto sin ningún asistente para la aplicación.*

A continuación en la figura 4.18, se muestra la ventana para especificar el directorio donde se almacenara la información al crear el proyecto. Además de que es el momento en que debemos especificar las características del dispositivo.

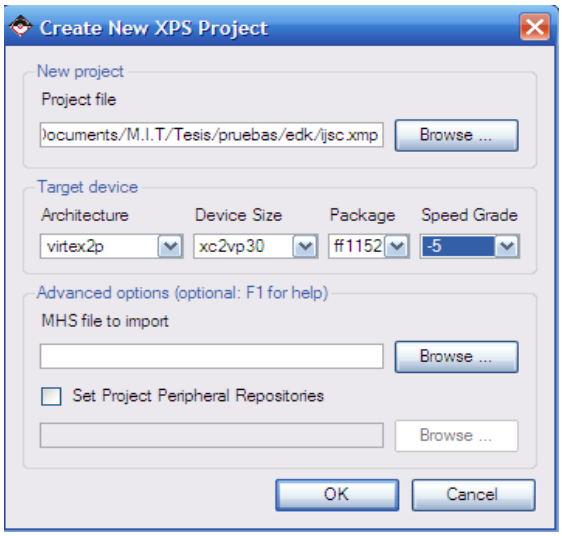



*Figura 4.18 Especificación del directorio creado para el proyecto*

Una vez especificadas las características del dispositivo usado, se inicia el trabajo sobre el ambiente grafico que ofrece la interfaz XPS.

**Interfaz XPS**

Ésta interfaz de la herramienta XPS, es el ambiente grafico que facilita el trabajo de desarrollo del proyecto, visualmente podemos notar que el ambiente está dividido en dos partes.

En la parte izquierda de la ventana de interacción, podemos observar y obtener el catalogo de periféricos, para añadir o quitar del sistema, además de encontrar los archivos del proyecto, desde donde se pueden abrir y editar fuentes o aplicaciones, que van a ejecutarse en el microprocesador. Esta ventana es mostrada en la figura 4.19.

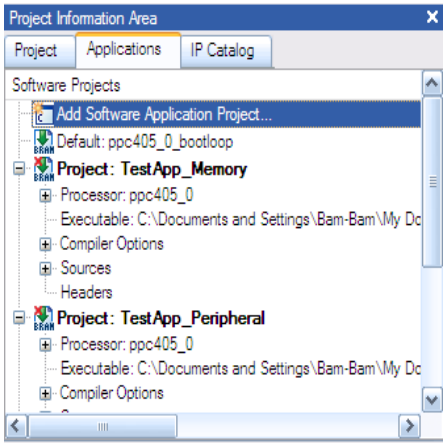



*Figura 4.19 Ventana donde se observan las fuentes y aplicaciones del sistema desarrollado.*

En la segunda parte de la herramienta, se muestra la estructura del sistema que se está diseñando. En la figura 4.20, a través del filtro de Bus Interface, se puede observar a los buses y elementos agregados en el sistema y su interconexión.

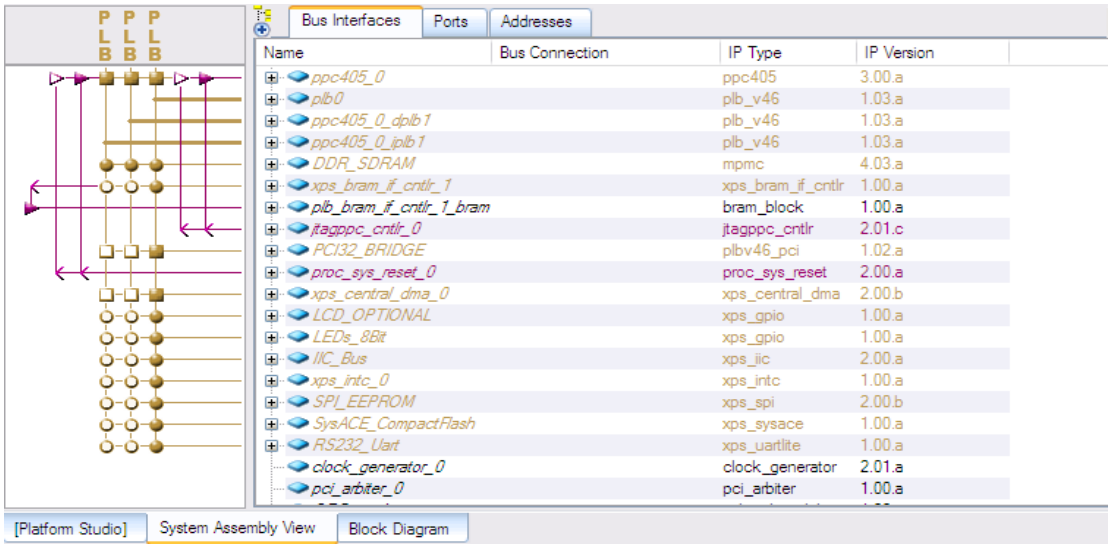



*Figura 4.20 Vista del sistema a nivel de elementos y buses implementados.*

Si se selecciona el filtro Ports, se puede ver y modificar las distintas conexiones a nivel de puertos que tienen los distintos elementos del sistema, así como a los externos. Y finalmente el filtro Addresses, muestra y permite modificar las direcciones de memoria en las que están mapeados los distintos elementos del sistema conectados a algún bus.

**Añadiendo Periféricos**

Según el diagrama mostrado en la figura 4.10, para obtener la conectividad en el sistema se debe incluir al periférico ILMB que da conexión entre la memoria y PPC. Esto es mostrado en la figura 4.21.

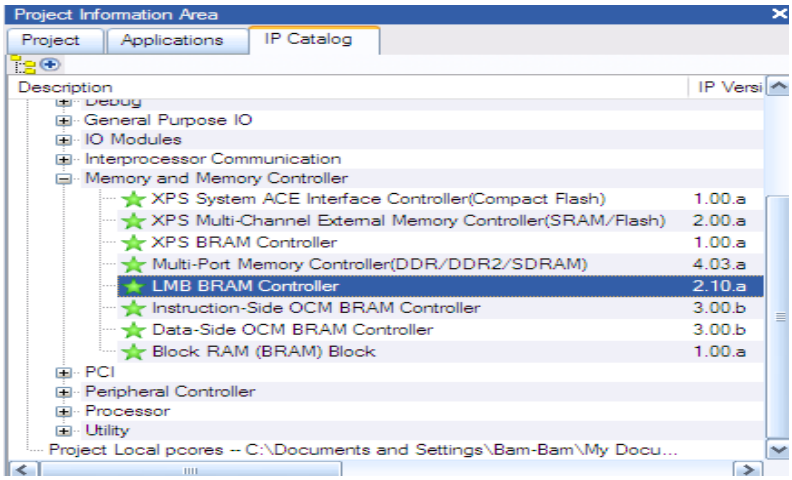



*Figura 4.21 Vista de la localización del bus ILMB para la conexión con la memoria.*

Se debe buscar al periférico en la pestaña de *IP Catalog* y basta con arrastrarlo a la parte derecha de la pantalla *Memory and memory controller > LMB BRAM Controller*, para ser agragado. El resultado de esta acción es mostrado en la ventana de la figura 4.22.

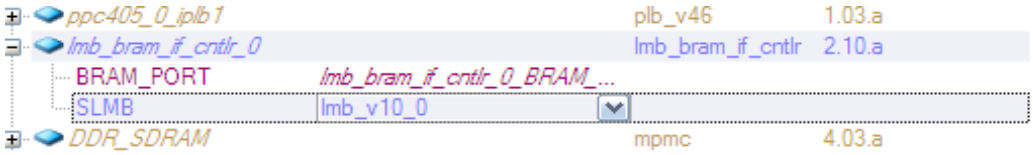



*Figura 4.22 Vista del IP Catalog, al momento de agregar al bus ILMB.*

Los periféricos tienen diferentes opciones de configuración. En la figura 4.23 se observan estas opciones para elegir la óptima.

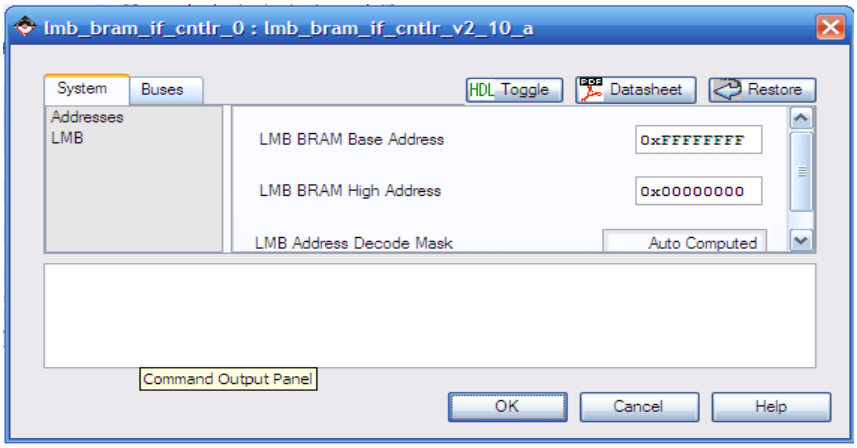



*Figura 4.23 Opciones que presentan los dispositivos para ser agregados al sistema*

Si existe alguna duda sobre estos dispositivos o su funcionamiento, se puede tomar la ayuda proporcionada por esta ventana, desde donde se puede acceder a las hojas de especificaciones de éstos y conocer su estructura interna y modo de conexión utilizada para este proyecto.

Hasta este punto, se tiene el sistema casi listo para ser sintetizado por primera vez. Falta indicar en el archivo de restricciones (.ucf) los pines y buses del FPGA con los que se desea hacer corresponder los puertos externos del diseño para tener su restricción al sistema.

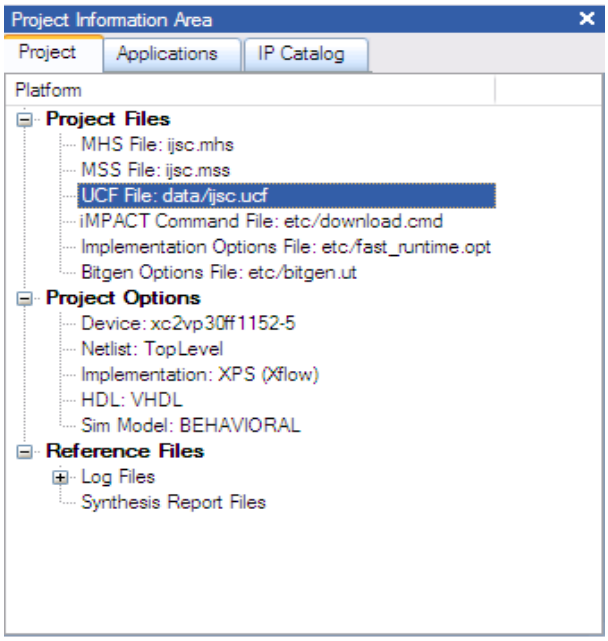



*Figura 4.24 Ventana con la información principal del proyecto desarrollado.*

Desde la pestaña de Project mostrada en la figura 4.24, se debe cambiar el archivo *Bitgen options file*. Este archivo indica las opciones de generación del archivo de programación de la placa.

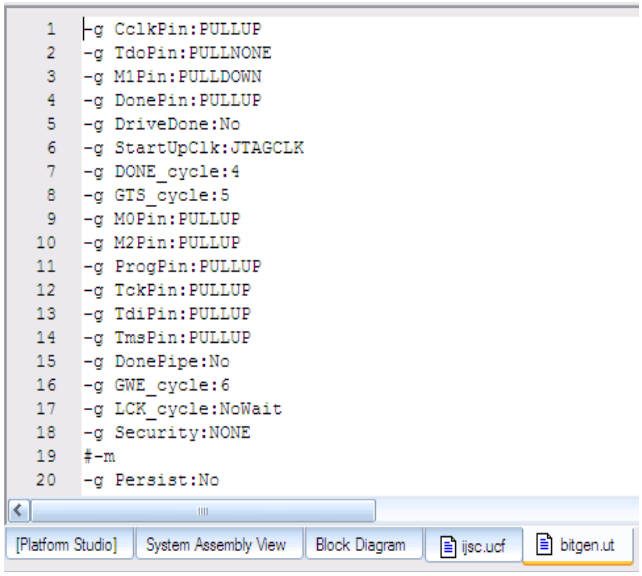



*Figura 4.25 Archivo .ucf que contiene la información en restricción del proyecto.*

Completada la etapa mostrada en la figura 4.25, se logra obtener la integración del diseño en la parte de hardware, correspondiente a la primera etapa descrita por la figura 4.15. Una vez hecho esto, ya se puede sintetizar el diseño para obtener el primer archivo .bit.

La ventana en la figura 4.26 muestra que el diseño ha sido sintetizado correctamente. Por lo que se debe proseguir a agregar al software de la aplicación.

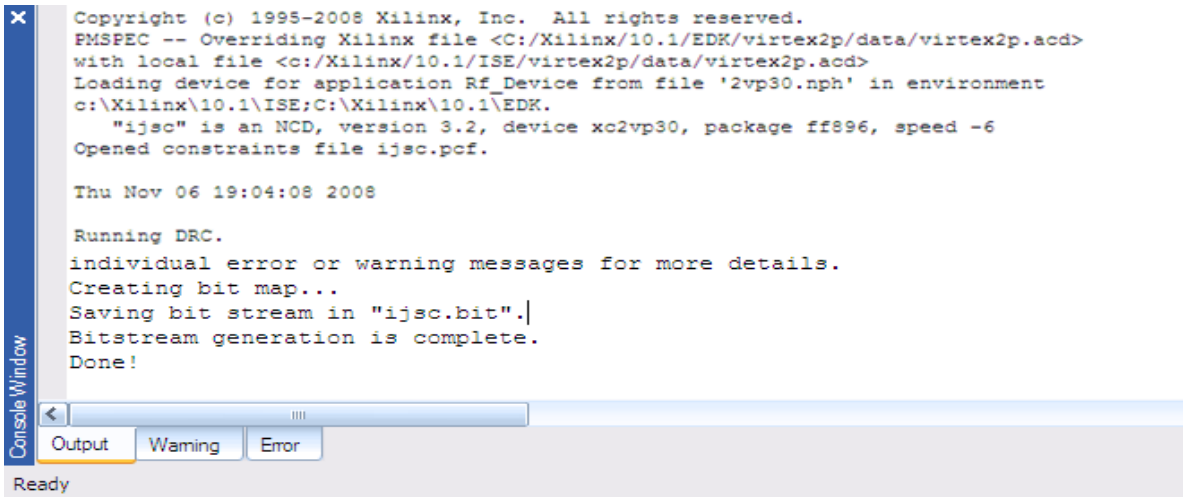



*Figura 4.26 Información que se muestra al completar la síntesis de la etapa de hardware.*

**Sistema Operativo**

Ahora todo está listo para agregar y configurar el software del sistema que finalmente ejecutará los programas desarrollados en MatLab y adaptados en lenguaje C, para ser evaluados sobre el FPGA.

Para configurar la parte software del sistema, hay que ir al menú Software->Software Platform Settings. Mostrado en la figura 4.27.

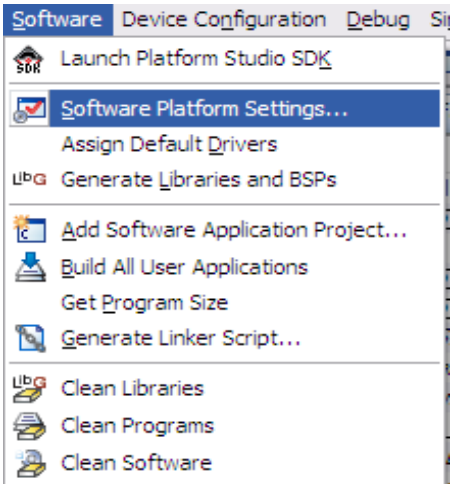



*Figura 4.27 Menú disponible para agregar la aplicación en software al proyecto*

Se abrirá la ventana mostrada en la figura 4.25, con diferentes opciones. La primera opción, Software Platform, permite modificar algunos parámetros del diseño, así como especificar las librerías que se deben incluir y si se utiliza sistema operativo o no.

Aquí se debe cambiar el parámetro CORE_CLOCK_FREQ_HZ al valor 10000000, que representa los 10MHz con que son muestreadas las señales GPS y que estará establecida como la frecuencia de reloj para el sistema que se está implementando.

También se debe seleccionar en el parámetro xmdstub_peripheral el valor opb_uartlite_0, pues indica que existe una conexión con un periférico de visualización, que de seguir indicada generara una advertencia al momento de sintetizar la aplicación.

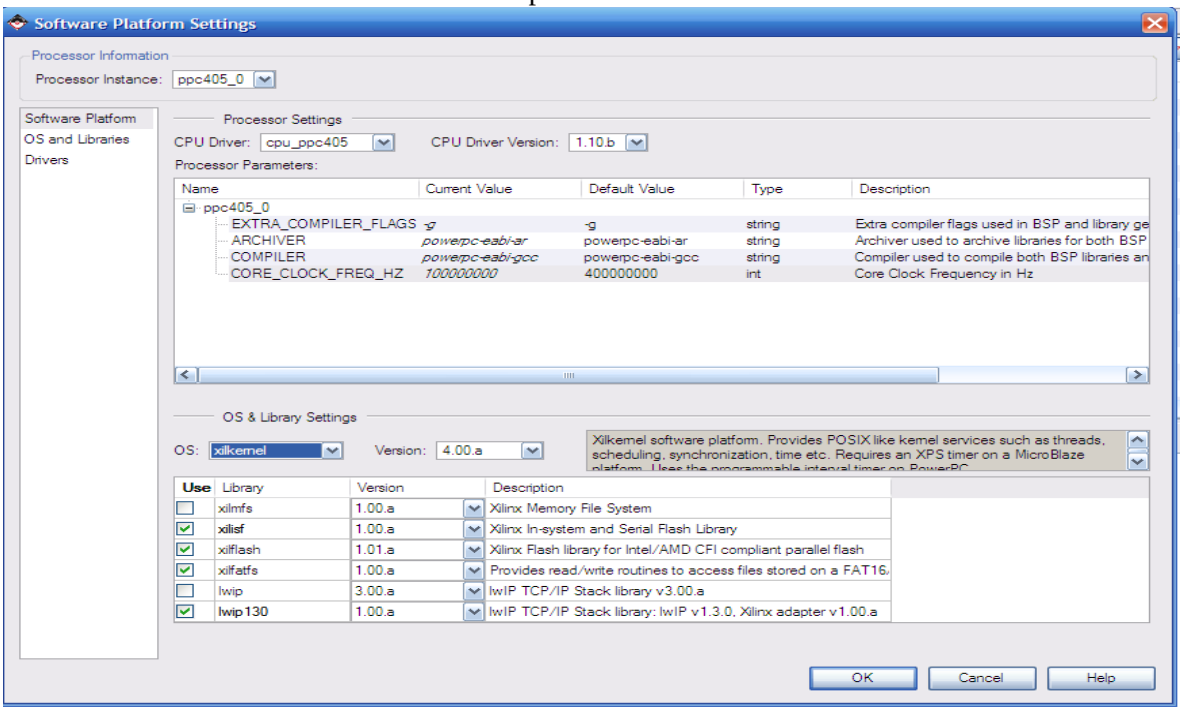



*Figura 4.28 Ventana que contiene las opciones principales para la generación de software.*

En la siguiente opción del menú del proyecto, OS and Libraries (fig. 4.28) se pueden modificar parámetros del sistema operativo que se vaya a utilizar. El resto de opciones, Drivers e Interrupt Handlers, no es necesario modificarlas, ya que se usarán los drivers por defecto, y no hay ningún dispositivo actualmente que genere interrupciones.

Después de establecer los parámetros del sistema se inicia la tarea de crear la aplicación software, dentro del menú de la figura 4.29 se abre la pestaña Applications de la parte izquierda de la ventana de Platform Studio. Una vez allí, se trabaja sobre Add Software Application Project.

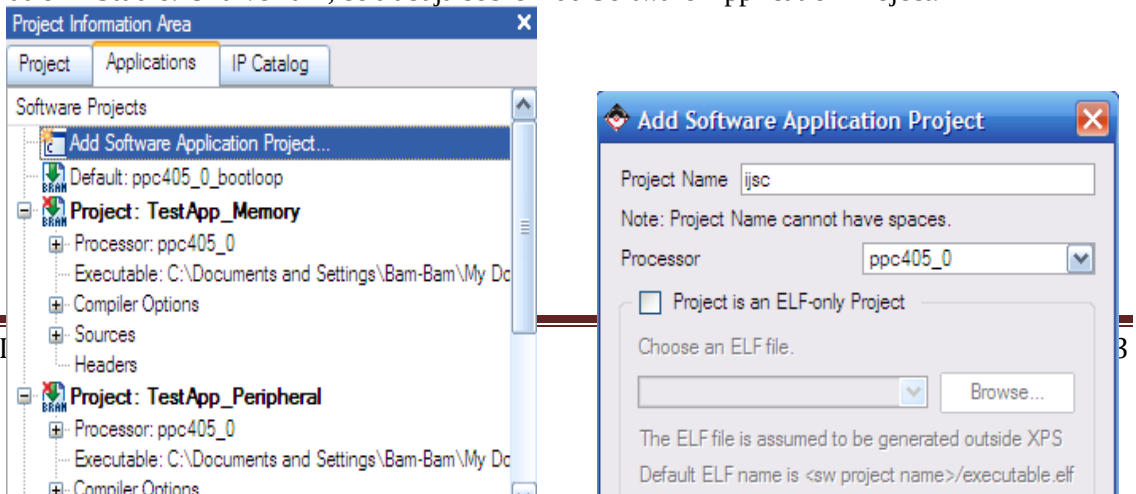


*Figura 4.29 Ventanas que muestran las opciones al iniciar un nuevo proyecto es software.*

En este punto, se tiene a la vista el árbol con los componentes del proyecto. Figura 4.30.

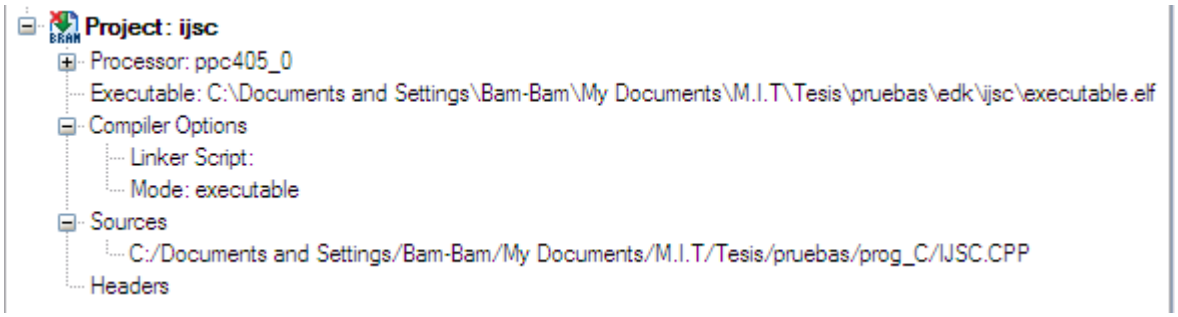



*Figura 4.30 Árbol con todos los componentes que integran al proyecto.*

En este árbol se debe enfocar el trabajo sobre la rama y la opción de Sources. En el cuadro de búsqueda de archivo que aparece, se debe localizar al archivo .c, que describe el código de programa desarrollado para la aplicación. Esta acción hará que se agregue al proyecto, el archivo con la información software de la aplicación.

Una vez añadido el archivo .c, se pueden modificar distintas opciones del compilador, mostradas en la figura 4.31. Se pueden cambiar parámetros como el *script* de ligado a utilizar, dirección de arranque del programa, optimizaciones, etc. Para este proyecto, se va a modificar la opción de nivel de optimización, desde Debug and Optimization, puesto que se descartara la opción de optimización.

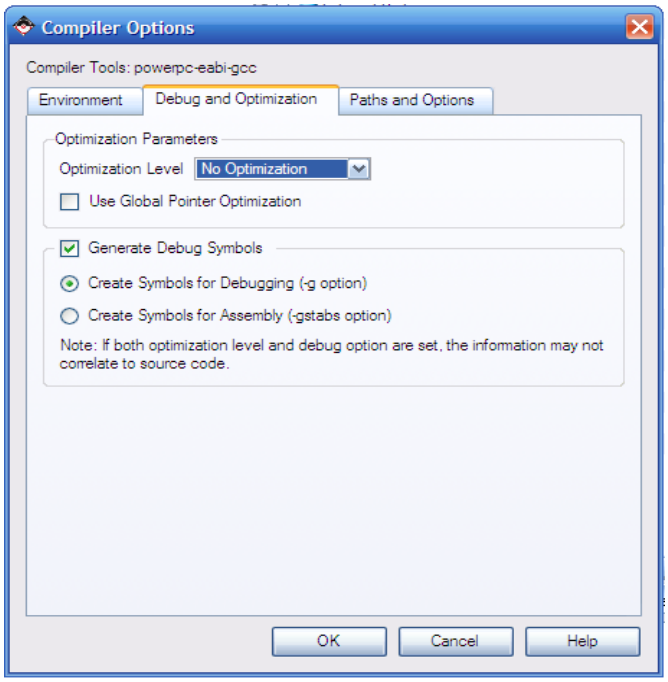



*Figura 4.31 Opciones del compilador.*

Hecho esto sólo queda compilar la aplicación, que habrá de incluirse en la memoria del sistema para que se ejecute cuando ésta se ha descargada sobre el kit.

Para verificar la aplicación, la rama principal del proyecto software debe ser localizada bajo el nombre que se da al iniciar su desarrollo. Mediante el Build Project, se realiza la compilación del archivo en lenguaje C, que contiene uno de los algoritmos de adquisición. Si la aplicación se compila correctamente, ya se puede incluir en la memoria del sistema este software.

Para ello, se ocupa la herramienta que inicializa la memoria de bloque y que debe alojar al código de la aplicación. Esto se consigue trabajando sobre la misma rama del proyecto software, seleccionando la opción de Mark to Initialize BRAMs. Estas dos acciones son vistas en la figura 4.32.

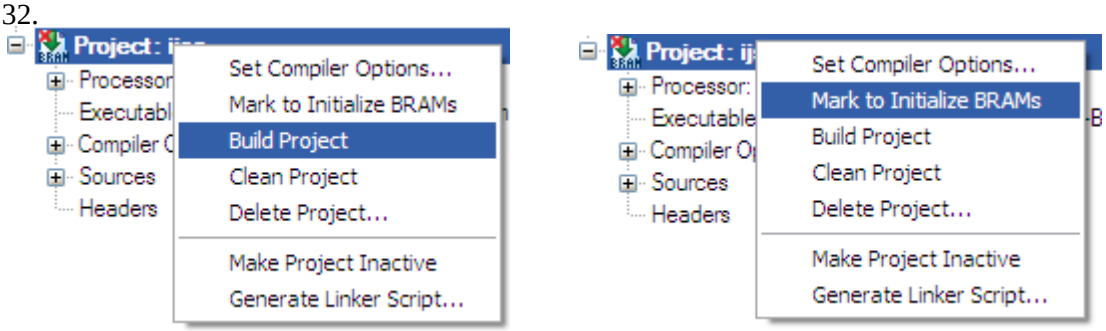



*Figura 4.32 Inicio de la compilación del proyecto e inicialización de la memoria de bloque.*

No se debe perder de vista el quitar de las memorias el bloque boot_loop que esta añadido por defecto, debido a que genera un inicio erróneo del sistema. Una vez hecho todo esto, el aspecto de la pestaña de Applications cambiara según la figura 4.33.

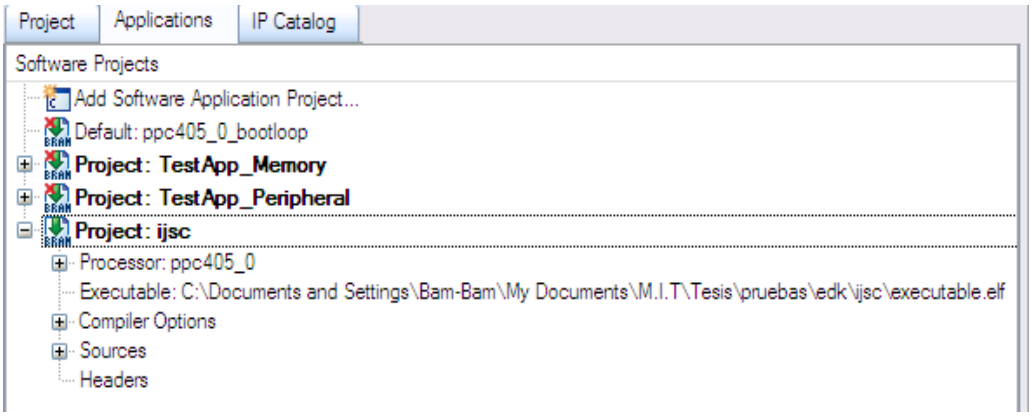



*Figura 4.33 Vista actualizada del contenido del proyecto*

Ahora ya se puede actualizar el archivo *bitstream* y descargarlo en el FPGA. Para ello sobre el menú Device Configuration->Update Bitstream, se realiza la opción de actualizar. Esto se puede observar en la figura 4.34.

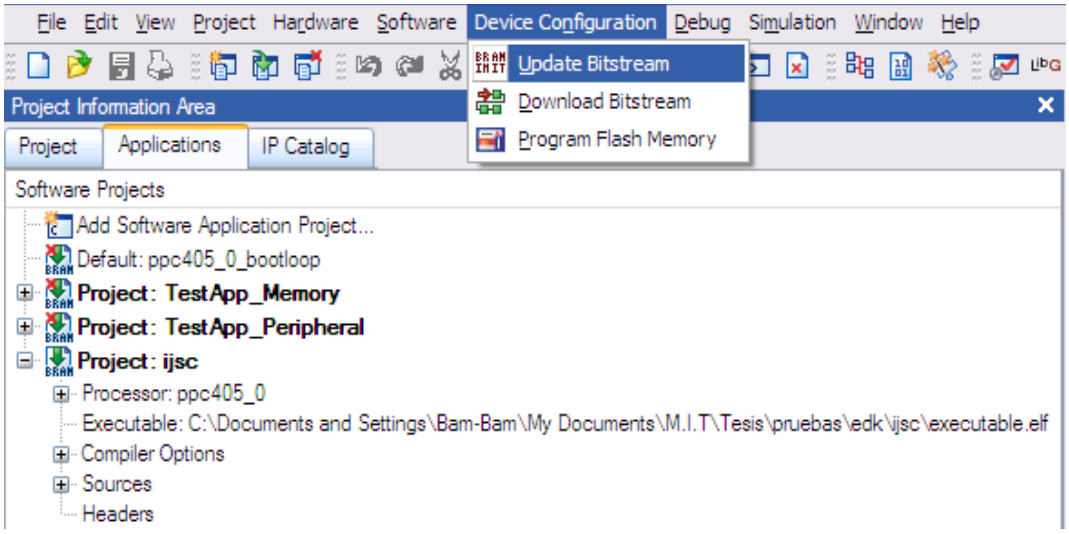



*Figura 4.34 Acciones para iniciar la generación del nuevo archivo .bit.*

Esta operación hará que se modifique el archivo de programación del PPC, para que la memoria BlockRam esté inicializada con el código del programa que se va a ejecutar.

El archivo de programación generado se identifica con el nombre inicial de *donwload.bit*, y se encuentra en el subdirectorio llamado *implementation*, dentro del directorio donde se ha realizado el proyecto. Este nombre puede ser cambiado por uno que identifique al proyecto y para que el programador lo localice con mayor facilidad.

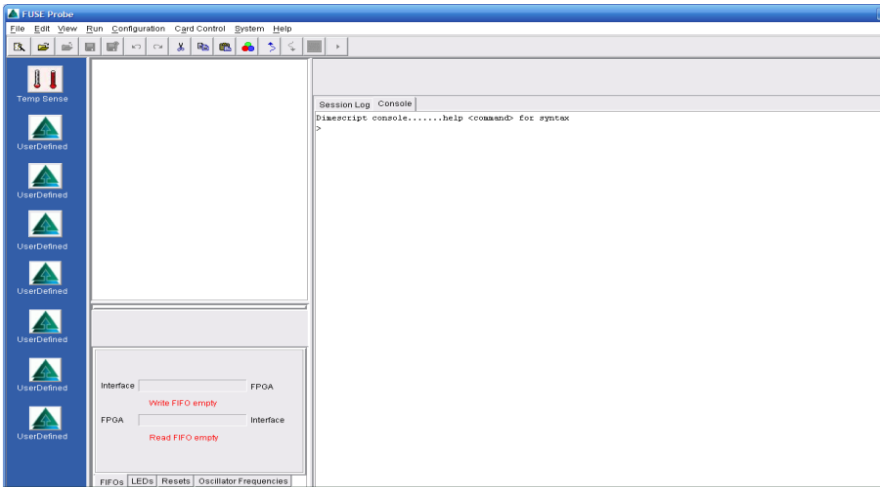
**Descarga del Archivo a la Tarjeta.**

La aplicación de EDK permite descargar al archivo *donwload.bit* por medio del dispositivo *Jtag* de Xilinx; sin embargo la tarjeta del kit XtremeDSP no cuenta con los conectores para realizar la

descarga a través de éste puerto. Debido a este problema se necesita encontrar la manera de poder continuar con el desarrollo de la aplicación, por lo mismo se debe recurrir al uso de una herramienta que sustituya esta tarea.

Con la adquisición de el kit de trabajo se obtiene la herramienta FUSE Probe de Nallatech, que brinda la oportunidad de descargar la aplicación a través del puerto USB. Con esto se logra eliminar las restricciones existentes.

Con la ventaja que representa el uso del puerto USB se arrancan las funciones de la herramienta.
En la ventana principal (fig.4.35) encontramos las diferentes opciones y acciones, con las que cuenta esta aplicación. En una de estas opciones se inicia el trabajo, donde se inicializa el uso de la tarjeta ocupada.




*4.35 Ventana inicial del ambiente FUSE Probe.*

En los diferentes menús de este ambiente gráfico, se localiza la pestaña para el control de la tarjeta (fig.4.36); así que desde aquí se debe realizar la tarea de acceder a la tarjeta y al ambiente de trabajo.

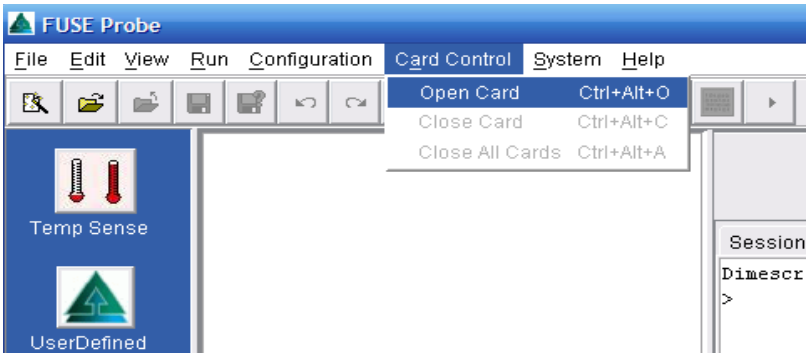



*Figura 4.36 Menú con la opción para abrir la tarjeta dentro del ambiente.*

Se debe hacer que la computadora, a través de esta herramienta logre reconocer al dispositivo. El primer paso para abrir la tarjeta, requiere de especificar el modo en que la tarjeta y la computadora están interactuando, para que sea de este modo como pueda ser descargado el programa en el FPGA.

Como ya se mencionó la ventaja que brinda esta herramienta es la utilizar al puerto USB para la descarga del archivo *donwload.bit*, así que éste debe ser especificado.

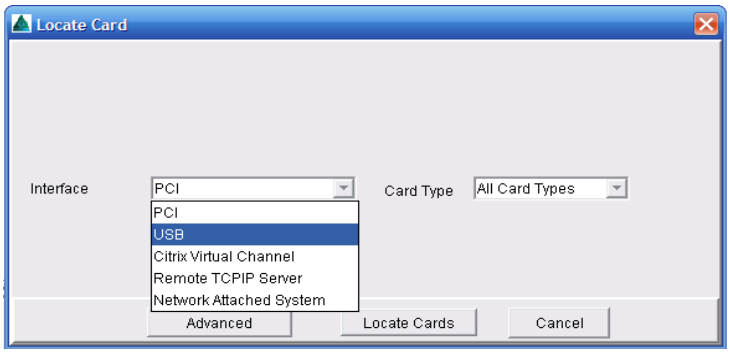



*Figura 4.37 Especificación del puerto USB para la identificación de la tarjeta.*

Después de ingresar la opción como se muestra en la figura 4.37, se debe realizar el reconocimiento de la tarjeta, con el icono de *locate cards* se comienza la búsqueda.

Justo después de realizar la operación, se mostrará una lista de todas las tarjetas que pudiesen estar conectadas por la misma vía, mostrada en la figura 4.38. Así que se elije la tarjeta utilizada por ésta aplicación y abrir su contenido.

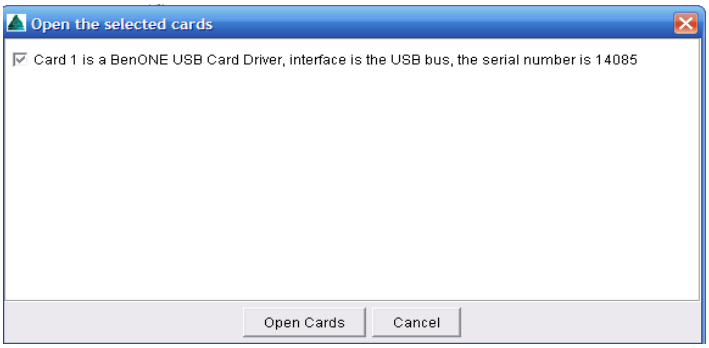



*Figura 4.38 Lista de las tarjetas reconocidas a través del puerto USB.*

Ahora en la ventana principal pueden ser observados los componentes que integran la tarjeta y algunas de sus características. Figura 4.39.

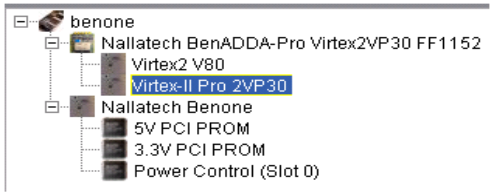



*Figura 4.39 Vista de la ventana principal con la información de la tarjeta conectada.*

Para poder descargar el archivo *donwload.bit*, desarrollado en la herramienta anterior, simplemente se abre el menú de *Configuration*, y en la opción de *Assign Bit file* se realiza la operación. Fig. 4.40.

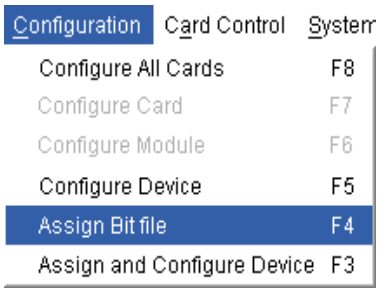



*Figura 4.40 Opciones en el menú para asignar al archivo .bit*

El resultado de esta acción se puede notar en la rama con las características de la tarjeta mostrada en la figura 4.41.

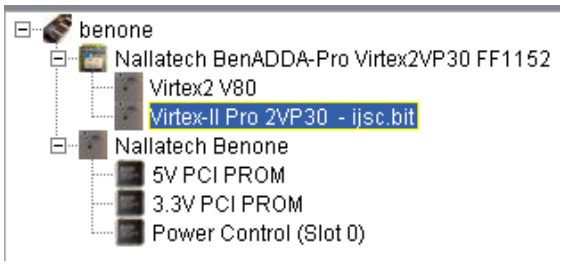



*Figura 4.41 Vista del proyecto al asignar el archivo .bit.*

Como tarea final se debe configurar al dispositivo para finalmente obtener la aplicación en la tarjeta. Para esto se debe almacenar la información de este desarrollo, con la finalidad de que cada vez que se abra la tarjeta en este u otro ambiente se pueda acceder a los datos obtenidos del proyecto. Para realizar esta acción se efectúa la tarea mostrada en la figura 4.42.

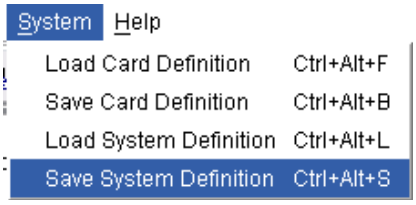



*Figura 4.42 Opción para almacenar la aplicación en la tarjeta*

Esto permite completar el proceso de implementación de los algoritmos de adquisición de señales GPS sobre el kit XtremeDPS, que contiene un FPGA VirtexII-PRO con un PowerPC405.

El proceso ha requerido del estudio, simulación y adaptación de estos algoritmos; sin embargo se ha completado con el objetivo principal, que es el de contenerlos en un ambiente de evaluación para tiempo real.

Ahora se debe realizar el análisis de los resultados obtenidos conforme al desarrollo del proyecto, identificando los detalles de todo el proceso realizado.

# Capítulo 5

# PRUEBAS Y RESULTADOS

Esta sección presenta los resultados obtenidos de la implementación de los algoritmos, en la tarjeta del kit XtremeDSP, desarrollada en la sección anterior. Se interpretan los datos arrojados por los ambientes de trabajo, por los cuales se fue transitando durante la elaboración del proyecto. Este análisis se divide en dos etapas; primero analizando los resultados obtenidos con el desarrollo de los algoritmos en MatLab y su migración al lenguaje C y una evaluación de su funcionamiento y desempeño en el microprocesador PPC del FPGA contenido en el kit XtremeDSP.

## 5.1 Datos de referencia

Con el propósito de verificar la correcta operación de los algoritmos y el sistema desarrollado, se generó un archivo de datos con la terminal de entrada RF. Este dispositivo estuvo instalado en el laboratorio de Radiocomunicaciones del edificio Z4 y fue usado en todas las pruebas realizadas.

A través del ambiente gráfico que está integrado a la terminal de entrada de RF, se capturó el archivo binario, con información de 32 MB de señal GPS. La ventana de esta aplicación es la mostrada en la figura 5.1.

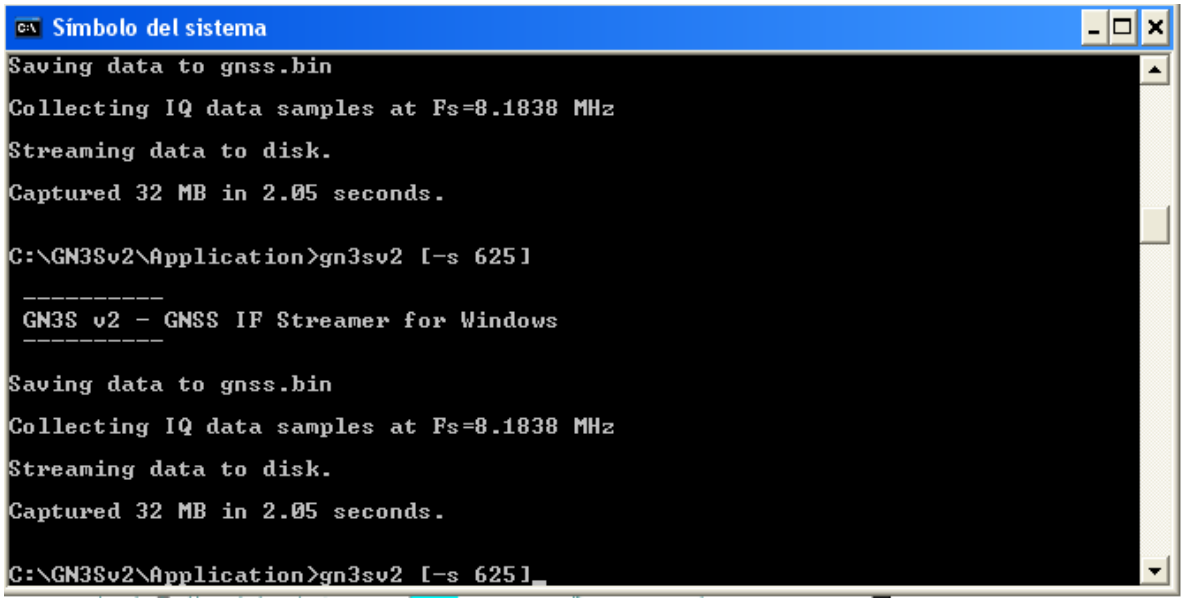



*Figura 5.1 Ventana con la información de la captura de la señal GPS en un archivo binario.*

Además para corroborar la validez de los datos capturados por la terminal, se recurrió a tomar información en línea de la situación en tiempo real del sistema GPS. La figura 5.2 muestra esta información.

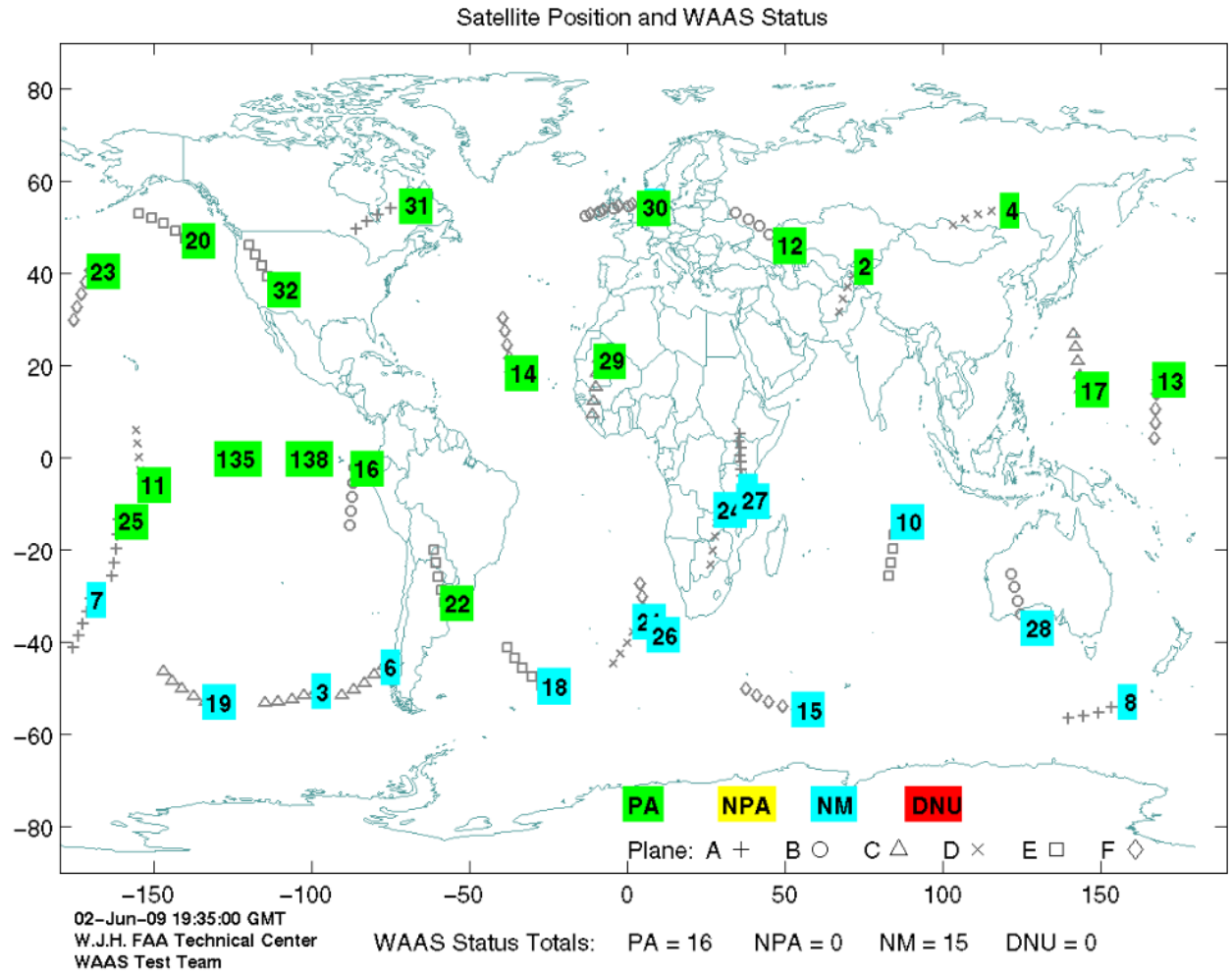
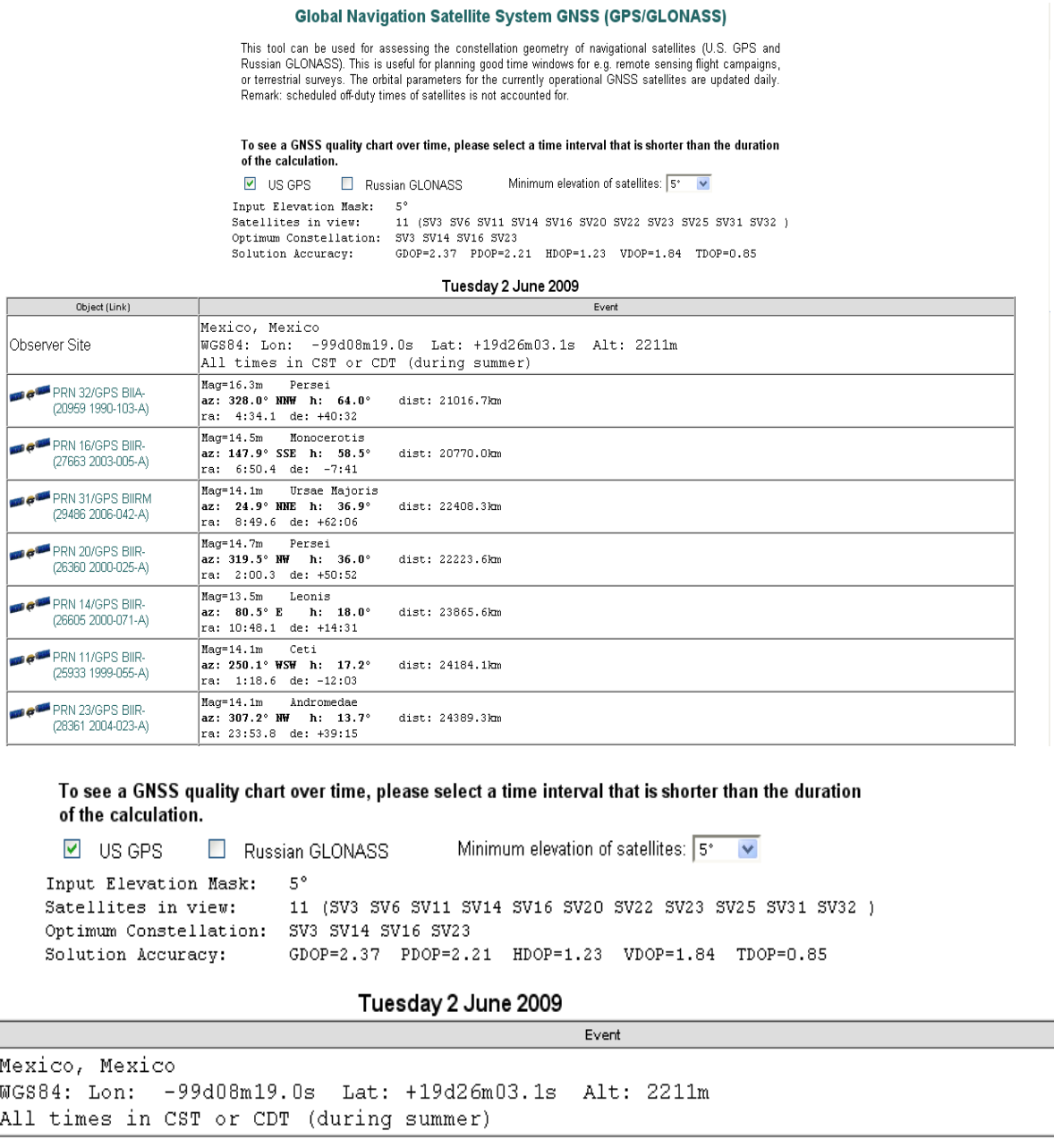



*Fig. 5.2 Información obtenida del almanaque de FAA que muestra la ubicación en tiempo real de la constelación GPS.*

En esta imagen se puede observar la situación actual (casi en tiempo real) de los satélites para la fecha y hora señaladas. En este caso nos interesa identificar a los satélites del sistema visibles a México, con respecto a la hora en que es tomada la muestra de la señal en la terminal de entrada de RF. Estos datos son importantes ya que la ejecución de los algoritmos desarrollados debería reportar la adquisición de los satélites que de acuerdo a esta imagen están visibles a México.

Esta información se asume válida ya que es proporcionada por la FAA (Federal Aviation Agency) de los Estados Unidos, la cual es la entidad encargada de administrar y operar el sistema GPS. Esta información puede ser consultada a través de la página web de esta institución.

De forma adicional se usó la información de otra organización que ofrece en línea datos en tiempo real de la posición de los satélites de diferentes constelaciones. Esta herramienta además de mostrar los satélites a la vista, proporciona información referente a la ubicación geográfica donde se toma la muestra de la señal.



**Global Navigation Satellite System GNSS (GPS/GLONASS)**

This tool can be used for assessing the constellation geometry of navigational satellites (U.S. GPS and Russian GLONASS). This is useful for planning good time windows for e.g. remote sensing flight campaigns, or terrestrial surveys. The orbital parameters for the currently operational GNSS satellites are updated daily. Remark: scheduled off-duty times of satellites is not accounted for.

**To see a GNSS quality chart over time, please select a time interval that is shorter than the duration of the calculation.**

☑ US GPS ☐ Russian GLONASS Minimum elevation of satellites: 5°

```
Input Elevation Mask:   5°
Satellites in view:     11 (SV3 SV6 SV11 SV14 SV16 SV20 SV22 SV23 SV25 SV31 SV32 )
Optimum Constellation:  SV3 SV14 SV16 SV23
Solution Accuracy:      GDOP=2.37  PDOP=2.21  HDOP=1.23  VDOP=1.84  TDOP=0.85
```

**Tuesday 2 June 2009**

| Object (Link) | Event |
|---|---|
| Observer Site | Mexico, Mexico<br>WGS84: Lon: -99d08m19.0s Lat: +19d26m03.1s Alt: 2211m<br>All times in CST or CDT (during summer) |
| PRN 32/GPS BIIA-(20959 1990-103-A) | Mag=16.3m Persei<br>**az: 328.0° NNW h: 64.0°** dist: 21016.7km<br>ra: 4:34.1 de: +40:32 |
| PRN 16/GPS BIIR-(27663 2003-005-A) | Mag=14.5m Monocerotis<br>**az: 147.9° SSE h: 58.5°** dist: 20770.0km<br>ra: 6:50.4 de: -7:41 |
| PRN 31/GPS BIIRM(29486 2006-042-A) | Mag=14.1m Ursae Majoris<br>**az: 24.9° NNE h: 36.9°** dist: 22408.3km<br>ra: 8:49.6 de: +62:06 |
| PRN 20/GPS BIIR-(26360 2000-025-A) | Mag=14.7m Persei<br>**az: 319.5° NW h: 36.0°** dist: 22223.6km<br>ra: 2:00.3 de: +50:52 |
| PRN 14/GPS BIIR-(26605 2000-071-A) | Mag=13.5m Leonis<br>**az: 80.5° E h: 18.0°** dist: 23865.6km<br>ra: 10:48.1 de: +14:31 |
| PRN 11/GPS BIIR-(25933 1999-055-A) | Mag=14.1m Ceti<br>**az: 250.1° WSW h: 17.2°** dist: 24184.1km<br>ra: 1:18.6 de: -12:03 |
| PRN 23/GPS BIIR-(28361 2004-023-A) | Mag=14.1m Andromedae<br>**az: 307.2° NW h: 13.7°** dist: 24389.3km<br>ra: 23:53.8 de: +39:15 |

**To see a GNSS quality chart over time, please select a time interval that is shorter than the duration of the calculation.**

☑ US GPS ☐ Russian GLONASS Minimum elevation of satellites: 5°

```
Input Elevation Mask:   5°
Satellites in view:     11 (SV3 SV6 SV11 SV14 SV16 SV20 SV22 SV23 SV25 SV31 SV32 )
Optimum Constellation:  SV3 SV14 SV16 SV23
Solution Accuracy:      GDOP=2.37  PDOP=2.21  HDOP=1.23  VDOP=1.84  TDOP=0.85
```

**Tuesday 2 June 2009**

| Event |
|---|
| Mexico, Mexico<br>WGS84: Lon: -99d08m19.0s Lat: +19d26m03.1s Alt: 2211m<br>All times in CST or CDT (during summer) |

*Figura 5.3 Información obtenida de los satélites visibles al receptor.*

En el cuadro de la figura 5.3 se observan los parámetros en hora y fecha de la ubicación y funcionamiento del GPS. Además se puede obtener una lista de los posibles satélites visibles a esta área, entregando la lista de la una óptima adquisición de satélites.

La información de estos dos sitios web es registrada para la misma fecha y hora en que fue capturado el archivo binario de la terminal de entrada RF, mostrando una lista de los satélites visibles de acuerdo a estos sitios para el día martes 2 de Junio del 2009 a las 19:35 GMT (14:35 hrs de la Ciudad de Mexico).

Toda esta información será usada para verificar que el sistema desarrollado funciona correctamente.

## 5.2. Simulación en MatLab.

La primer etapa desarrollada fue la obtenida en la implementación de los algoritmos en un ambiente de simulación en lenguaje C/MatLab.

### 5.2.1 Adquisición en Búsqueda Serial

El primer algoritmo implementado fue el de búsqueda serial, en su análisis se distinguieron sus características; sin embargo ahora se denotan las ventajas o desventajas en las que se puede incurrir usando este algoritmo.

Las figuras. 5.4 y 5.5 muestran los dos casos para la adquisición en este método según el desempeño del algoritmo cuando un satélite es adquirido por el receptor. Esto quiere decir que a través del parámetro de umbral, se identifica a los satélites mejor ubicados para la medición.

En la figura 5.4 se observan gráficas con picos pronunciados que significan valores altos en la operación de comparación (correlación) de los códigos localmente generados con los recibidos a través de la señal GPS capturada. Ahí mismo se puede identificar al número de código PRN que se está adquiriendo

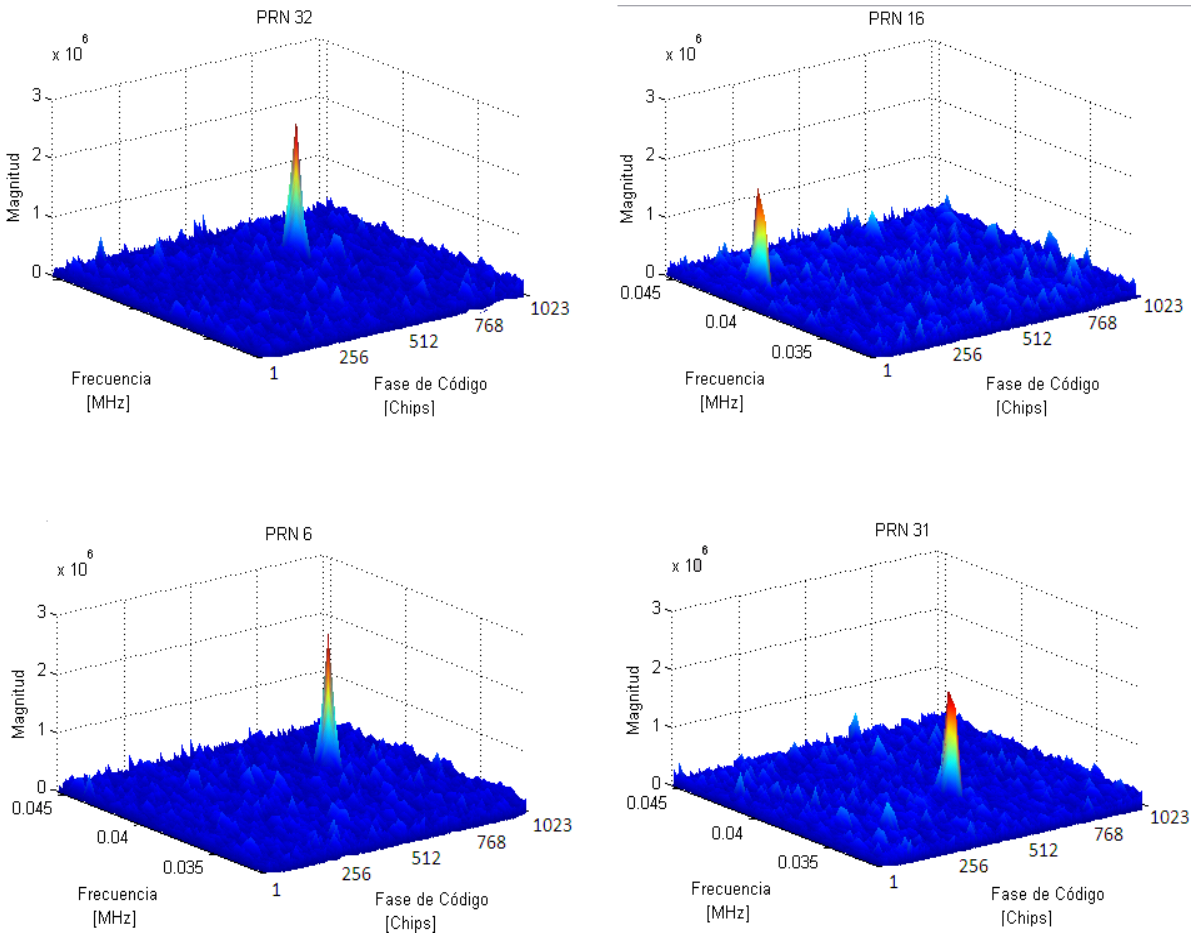



*Fig.5.4 Salida para la adquisición serial, se muestra que los VS 6, 16, 31, 32, presentan picos significativos en su función de correlación. Por lo tanto están siendo adquiridos.*

La figura 5.5 muestra el desempeño del algoritmo cuando un satélite no está siendo adquirido y por tanto esta fuera de la línea de vista del receptor. Las gráficas muestran a la función de correlación sin ningún pico pronunciado, solo con valores de bajo rendimiento y poca captación. Lo cual significa que estos satélites no están siendo adquiridos.

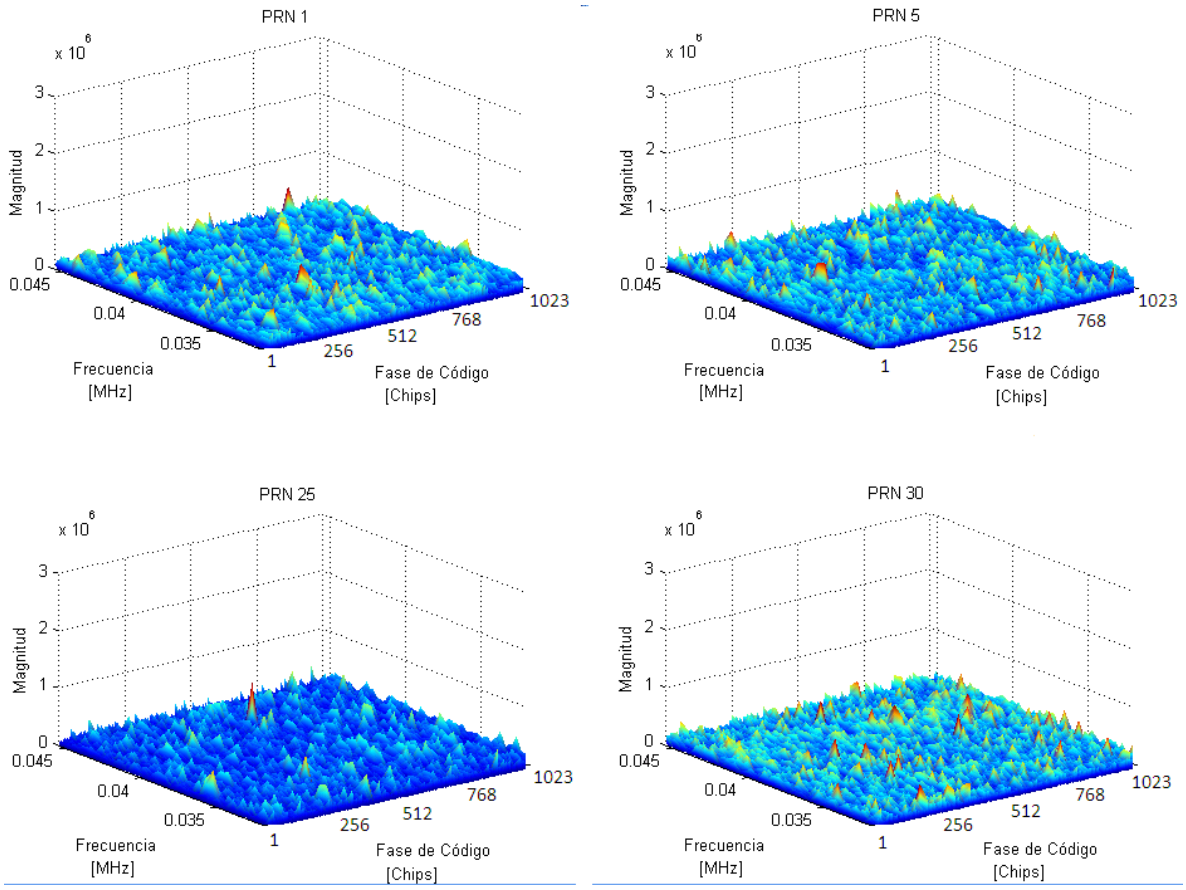



*Fig.5.5 Salida para la adquisición serial. Los satélites no presentan valores significativos en su función de correlación, por lo tanto no son visibles al receptor.*

Estas gráficas obtenidas y el número de código PRN que representa a los satélites, comparados con los obtenidos con el almanaque GPS, a la fecha y hora de la toma de muestra de la señal, aseguran que la adquisición es válida. Sin embargo hay que tomar en cuenta algunas otras características del algoritmo.

El método de búsqueda serial hace dos barridos en su búsqueda: Un barrido de la frecuencia, que debe hacerse sobre todas las frecuencias portadoras posibles de IF ±10kHz en pasos de 500Hz. Además de realizar un barrido sobre las fases del código, esto quiere decir que hará un recorrido sobre 1023 diferentes fases de los códigos PRN. Todo el trabajo de la búsqueda serial se resume en un total de: 41943 combinaciones posibles. Obviamente, éste es un número muy grande de combinaciones. Por lo que esta rutina de búsqueda es ardua y tiende a ser la debilidad principal de la adquisición de búsqueda serial. Debido a que su funcionamiento requiere del consumo de muchos cálculos y por consecuencia del consumo de un cierto tiempo de cómputo.

### 5.2.2. Búsqueda Paralela en el Espacio de la Frecuencia

Como se acaba de observar, el método serial de adquisición demostró que es un procedimiento muy desperdiciador de tiempo al buscar secuencialmente todos los valores posibles para los dos parámetros frecuencia y fase del código.

Recordemos que este segundo método hace uso de la transformada de Fourier para hacer una búsqueda en el espacio de la frecuencia. Cuando la señal de entrada se alinea al código generado localmente, la salida de la FFT muestra un pico pronunciado.

Para encontrar la frecuencia del pico se calcula el valor absoluto de todos los componentes. La figura 5.6 muestra los casos en que la señal de un satélite es adquirida, mostrando un pico pronunciado.

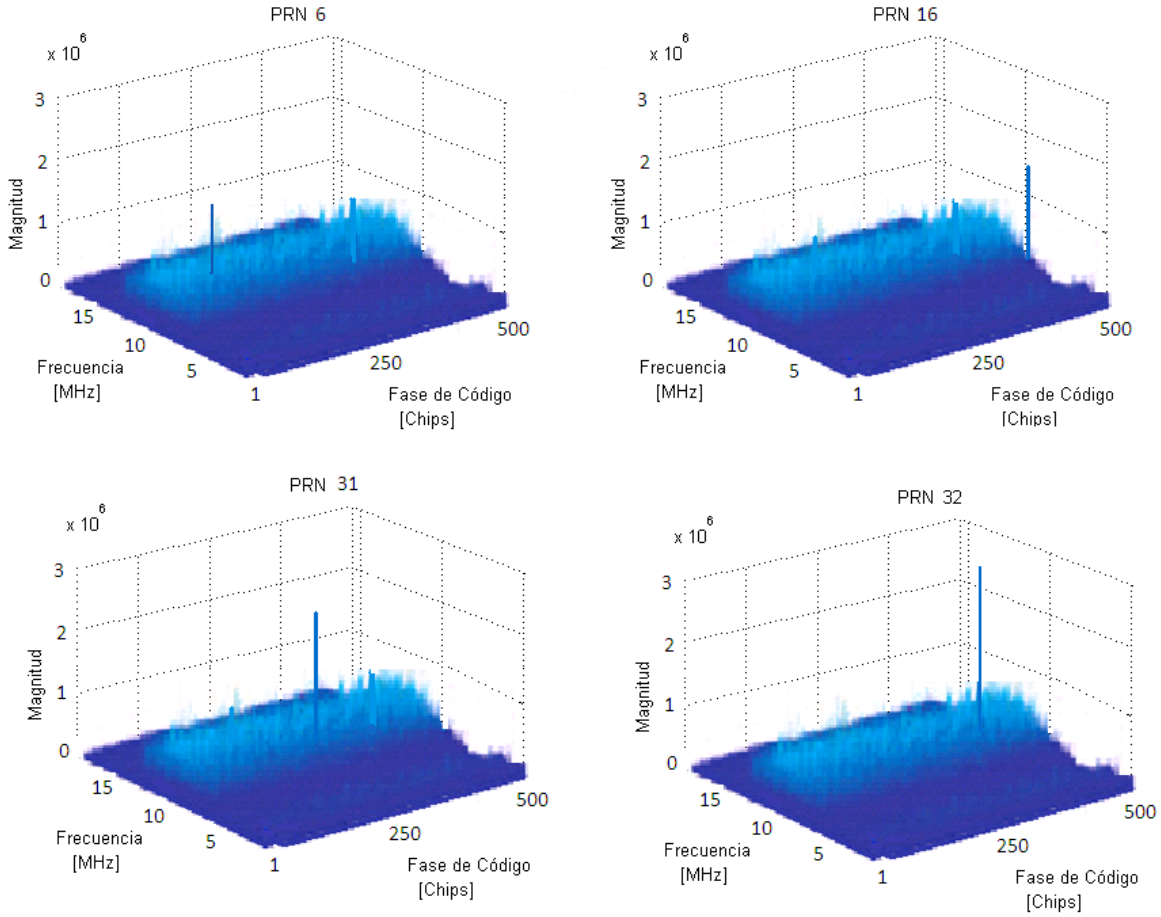



*Fig. 5.6 Salida del método de búsqueda paralela en el dominio de la frecuencia. La salida muestra picos significantes en la adquisición para satélites visibles.*

Para el caso en que las señales no contienen picos pronunciados, como en la figura 5.7. Los valores no contienen la suficiente información para indicar que los satélites están a la vista del receptor.

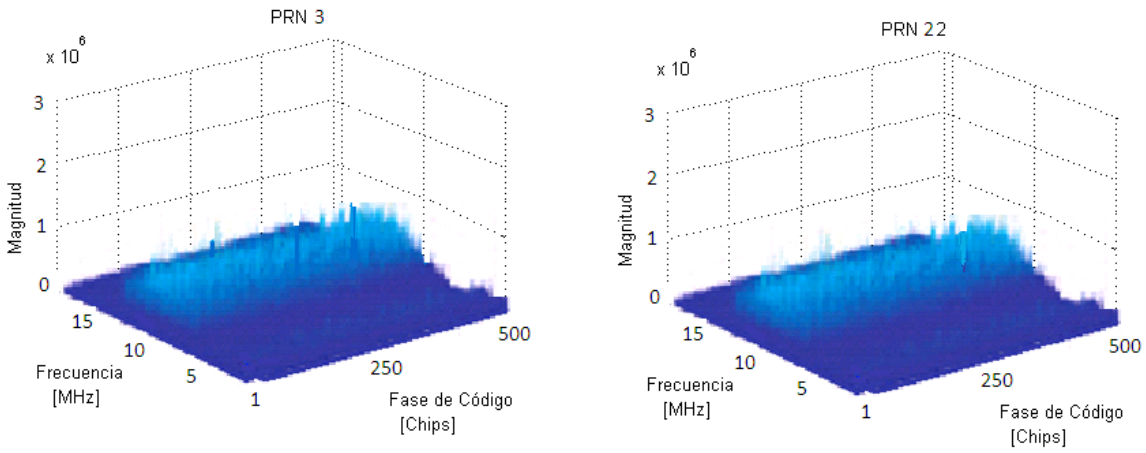



*Fig. 5.7 Salida del método de búsqueda paralela en el dominio de la frecuencia. La salida muestra picos insignificantes en la adquisición por lo que no son visibles al receptor.*

Los satélites visibles en este método coinciden con la lista obtenida en el método serial; sin embargo se debe hacer notar que las gráficas muestran una diferencia, pues en la búsqueda serial se hace una comparación en el dominio del tiempo y en este método la comparación se realiza en el espectro de la frecuencia; por el uso de la transformada de Fourier.

Como se ha observado cuando los pasos de la adquisición serial son a través de las posibles frecuencias y las fases de código, la adquisición paralela avanza solamente con las 1023 diversas fases de código.

Recordando que este método paraleliza la búsqueda y que esta acción viene con el costo de una transformación al dominio de la frecuencia de cada fase de código. Entonces este método solo recorre o tiene 1023 iteraciones, pero se debe tomar en cuenta que el trabajo en cálculo aumenta debido a la ejecución de la FFT.

### 5.2.3 Búsqueda Paralela en el Espacio de la Fase de Código

Si en el espacio de frecuencia en el algoritmo anterior se hizo paralelismo, en este método aplica una segunda modificación. Recordemos el momento en que se estudio a este algoritmo, la diferencia hacia el método anterior radica en que ahora sólo se deben realizar 41 pasos comparados a los 1023 del algoritmo en el espacio de la frecuencia.

La búsqueda se realiza solo sobre las 41 diferentes frecuencias de portador posibles, de los satélites que pueden ser adquiridos.

Este es el método de búsqueda paralela en el espacio de la fase de código y los resultados para ambos casos se muestran en la figura 5.8, donde de igual forma que en los anteriores podemos observar un pico pronunciado para el caso en que un satélite está siendo adquirido.

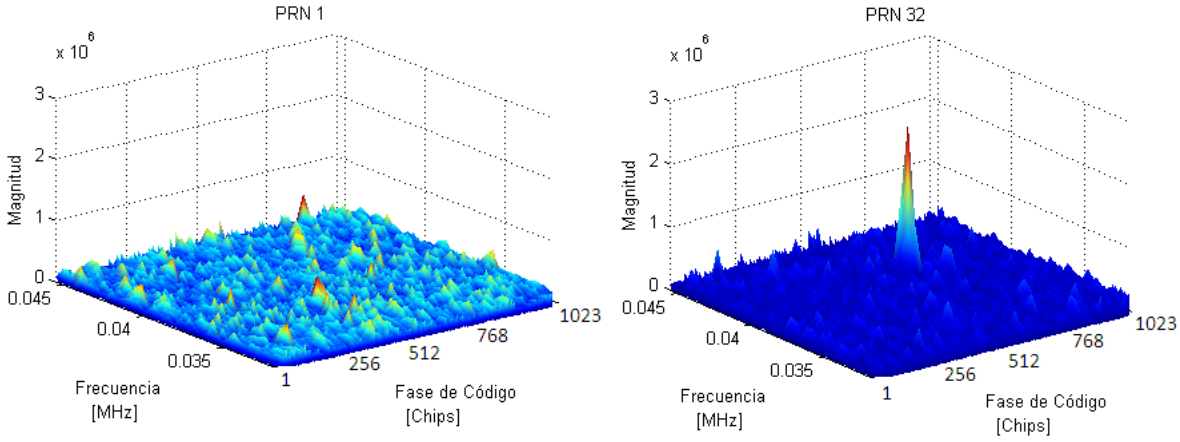



*Fig. 5.8 Salida al método de búsqueda paralela en el espacio del código de fase. a) Sv 1 no visible y con picos no presentes b) SV 32 es visible con un pico significante*

En esta ocasión solo se muestra a un par de graficas, sin embargo para este método se obtiene la misma lista de satélites adquiridos, mostrada posteriormente. En este método solo se recorre 41 veces al algoritmo; sin embargo tal y como al anterior método este corto recorrido viene acompañado de una mayor complejidad es su implementación y desarrollo. Pues este método debe realizar la función de transformada de Fourier, así como la trasformada inversa de Fourier. Esto requiere de aun más tiempo de cómputo para poder desarrollarse.

La figura 5.9 muestra los resultados a través de las ventanas en el ambiente de simulación con los resultados obtenidos para los tres algoritmos de adquisición.

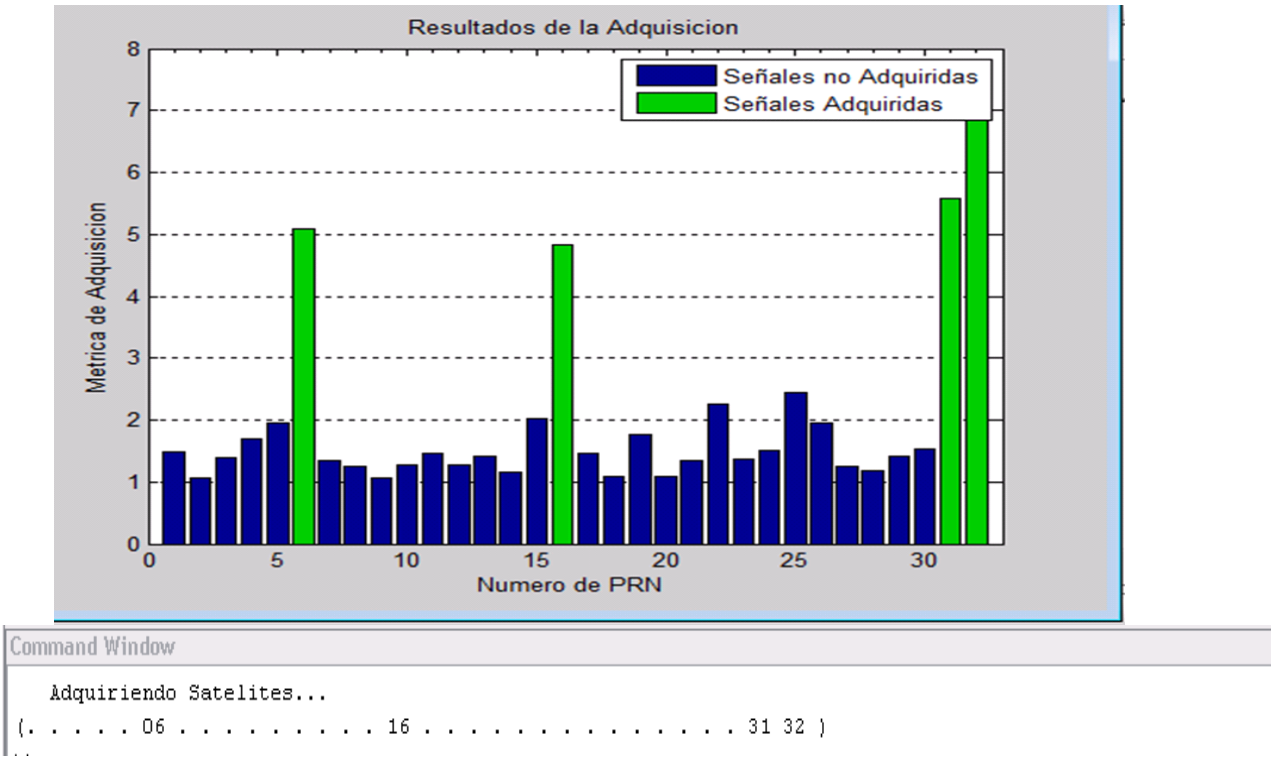



*Figura 5.9 Ventanas obtenidas en MatLab al ejecutar los algoritmos de adquisición.*

| Canal | PRN | Frecuencia | Doppler | Offset del Cod | Status |
|---|---|---|---|---|---|
| 1 | 32 | 4.09902e+004 | 2590 | 5240 | T |
| 2 | 31 | 3.67210e+004 | -1679 | 4404 | T |
| 3 | 6 | 4.03970e+004 | 1997 | 5861 | T |
| 4 | 16 | 4.13024e+004 | 2902 | 921 | T |
| 5 | --- | ---------- | ----- | ------ | Off |
| 6 | --- | ---------- | ----- | ------ | Off |
| 7 | --- | ---------- | ----- | ------ | Off |
| 8 | --- | ---------- | ----- | ------ | Off |

*Figura 5.10 Tabla que muestra los datos obtenidos en la etapa de simulación*

En la figura 5.10 se puede observar a la lista de satélites que estan siendo adquiridos, en la trama de señal GPS tomada para el desarrollo de la aplicación. Asi como los datos que son enviados a la etapa de seguimiento de la señal GPS.

En este punto se puede hacer una comparacion con la lista obtenida a través de las paginas web mostradas anteriormente. La tabla 5.1 muestra la similutud entre los datos obtenidos y los reportados por la FAA y el sitio CALSKY.

| Satélites visibles FAA | Lista optima de satélites adquiridos (CALSKY) | Satélites adquiridos por los algoritmos implementados |
|---|---|---|
| 3 | 3 | |
| 6 | | 6 |
| 11 | | |
| 14 | 14 | |
| 16 | 16 | 16 |
| 20 | | |
| 22 | | |
| 23 | 23 | |
| 25 | | |
| 31 | | 31 |
| 32 | | 32 |

*Tabla 5.1 Listas de los satélites segun la FAA, CalSky y los datos obtenidos por los algoritmos.*

Los datos de adquisición fueron alimentados a las etapas posteriores del procesamiento, desarrolladas en otros proyectos, para obtener la posición de longitud y latitud del lugar donde fue tomada la muestra.

Esta informacion de posición fue representada en la aplicación web Google Earth, tal como se muestra en la figura 5.11 y que sirve como validación de los datos adquiridos por los algoritmos.

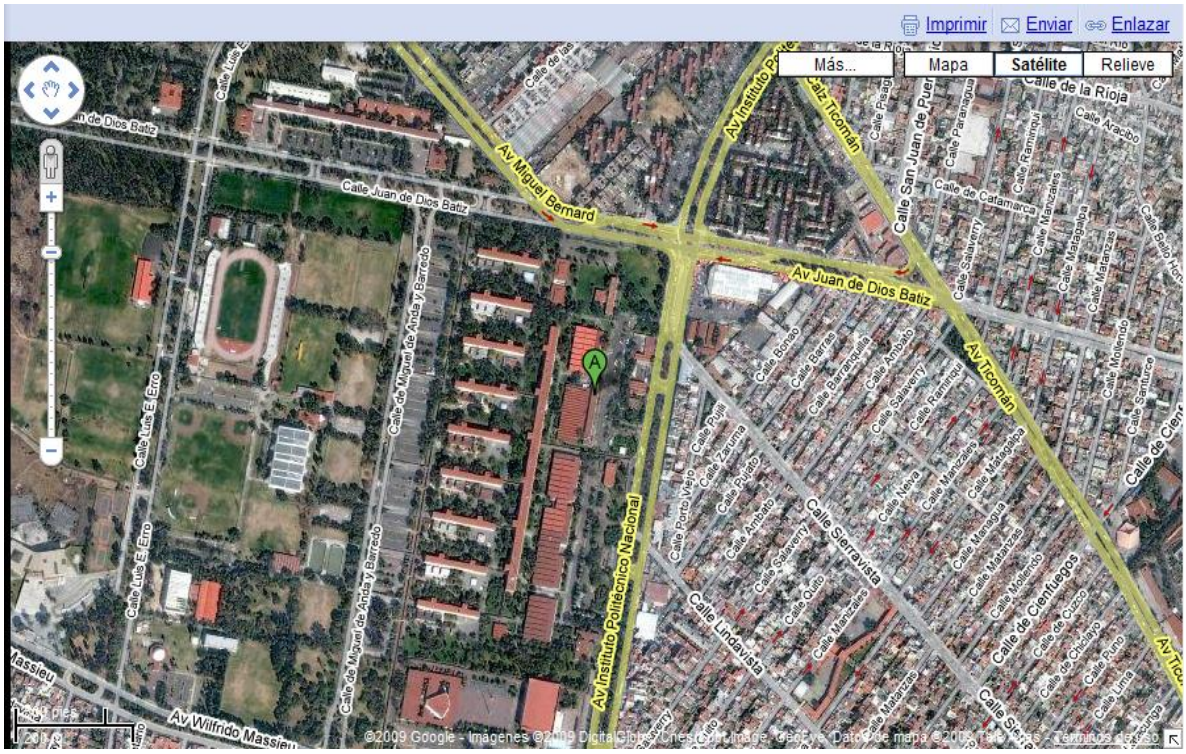



*Figura 5.11 Ventana de Google Earth que muestra la hubicacion del receptor con los datos de la adquisición de los algortimos.*

Estos resultados muestran concordancia con los valores extraídos del almanaque del sistema GPS y los obtenidos por el simulador.

Sin embargo recordemos que además de la implementación de los algoritmos otro objetivo de este trabajo es evaluar el desempeño de los mismos.

Así que en la tabla 5.2 se engloban las primeras conclusiones hasta esta etapa de simulación.

| *Algoritmo* | *Tiempo de Ejecución* | *Complejidad* | *Repeticiones* |
|---|---|---|---|
| **Búsqueda Serial** | Pendiente | Baja | *41943* |
| **Búsqueda Paralela en el espacio de la Frecuencia** | Pendiente | Media | *1023* |
| **Búsqueda Paralela en el Espacio del Código de Fase** | Pendiente | Alta | *41* |

*Tabla 5.2 Conclusiones de la primer etapa de simulación.*

Debido a que la implementación se desarrollo en diferentes computadoras, el tiempo de ejecución es un dato pendiente, pues por cada computadora se registra un tiempo de ejecución distinta, para esta etapa. Así que se determinará a ese tiempo, como el registrado en la etapa en que se desarrolla en la tarjeta con el FPGA.

## 5.3 Lenguaje C

El traslado al segundo ambiente de simulación, por el que debe pasar el proyecto es el ambiente de lenguaje C y la ventana obtenida al ejecutar el archivo compilado y ejecutado es la mostrada en la figura 5.12.

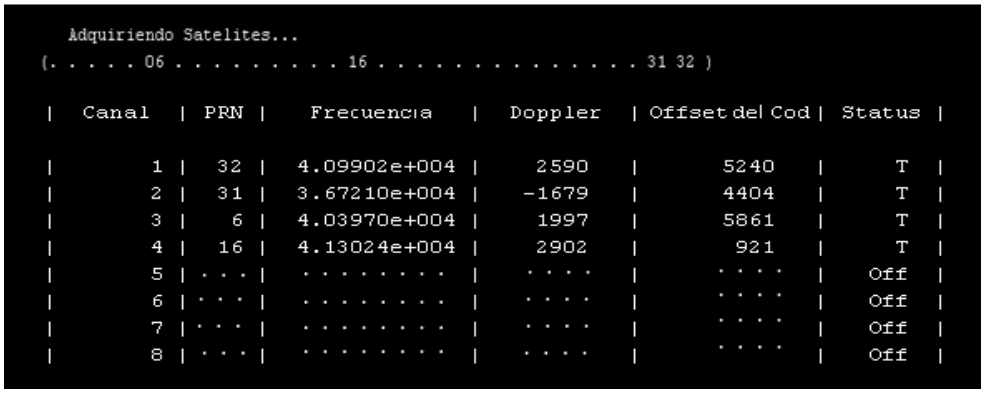



*Figura 5.12 Ventana del programa ejecutable desarrollado en lenguaje C.*

Los resultados son los mismos a los obtenidos en C/MatLab, por lo que la migración no ha influido en el resultado. Ahora que el desarrollo en este lenguaje ha implicado el aumento de las líneas de programación, debido a que las funciones utilizadas en MatLab han sido desarrolladas a su forma más primitiva, pensando en su reutilización para la etapa de implementación en el PPC.

No se debe perder de vista que el ambiente en lenguaje C es mucho más rápido en funcionamiento que el de MatLab, por lo que aun desarrolladas las funciones, el trabajo es realizado en menor tiempo. El único parámetro a distinguir en este ambiente, arroja conclusiones similares a la etapa anterior. La complejidad en el desarrollo aumenta conforma avanzamos en los métodos, el tiempo sigue siendo un parámetro no confiable debido a que se presenta el mismo fenómeno ocasionado por el cambio de computadora.

Algunos ejemplos del desarrollo de las funciones de MatLab a C se muestran a continuación.

```
for (PRN = 1;PRN<=32;PRN++)
      {
 int g2s[] =
{5,6,7,8,17,18,139,140,141,251,2
52,254,255,256,257,258,469,470,4
71,472,473,474,509,512,513,514,5
15,516,859,860,861,862,145,175,5
2,21,237,235,886,657,634,762,355
,1012,176,
603,130,359,595,68,386};
 int g2shift = g2s[PRN];
 int g1[1023]={0};
 int reg[10] = {-1};
 int acarr;
 for (int i=1;i<=1023;i++)
     {
      g1[i]= reg[10];
      acarr= reg[3]*reg[10];
      reg[10]=reg[9];
      reg[9]=reg[8];
      reg[8]=reg[7];
      reg[7]=reg[6];
      reg[6]=reg[5];
      reg[5]=reg[4];
      reg[4]=reg[3];
      reg[3]=reg[2];
      reg[2]=reg[1];
      reg[1]= acarr;
     }
int g2 [1023]={0};
int reg2[10] = {-1};
```

Este es el desarrollo de los registros de corrimiento para la obtención de los diferentes códigos PRN. Recordemos que este segmento código es utilizado en varias ocasiones durante los algoritmos y en comparación a las tres líneas desarrolladas por MatLab para la misma tarea, se nota una diferencia en el trabajo con ambos lenguajes.

Otro ejemplo claro es el desarrollo de la función FFT que en MatLab se representa por una sola línea, en lenguaje C que da como:

```
# define pi 3.14159265358
for (n=2; n<=N; n <<= 1)
{   w=2*pi/n;
  for (m=0; m<N; m+=n)
    {
    for (k=0; k<n/2; k++)
      {
      y=Out[m+k];
      z=Out[m+k+n/2] * exp(-ikw);
      Out[m+k]=(y+z)/2;
      Out[m+k+n/2]=(y-z)/2;
       }
    }
```

Entonces debido a que la migración de lenguaje de programación no altera los resultados, el siguiente paso es la evaluación de la implementación en el FPGA.

## 5.4 Tiempo real

Hasta este punto los algoritmos han sido probados, evaluados y confirmados por dos lenguajes de programación y sus ambientes de trabajo.

Ahora debemos analizar su funcionamiento y desempeño en la plataforma en la que finalmente se desenvolverán en tiempo real, al final del proyecto. Una vez implementado y visualizado el desarrollo del proyecto en el capítulo anterior, en la misma plataforma de trabajo XPS, se realiza la prueba estándar en tiempo para los tres algoritmos.

Esta prueba se enfoca a evaluar a los algoritmos con la finalidad de saber que tan rápidos son para el trabajo en tiempo real, el cual será desarrollado al completarse y conjuntarse las distintas etapas del receptor GPS.

Esta prueba consta de una toma de tiempo en desempeño, con lo algoritmos descargados al PPC. La prueba se realiza a través del mismo ambiente, la ventana que guía esta tarea es la mostrada en la figura 5.13.

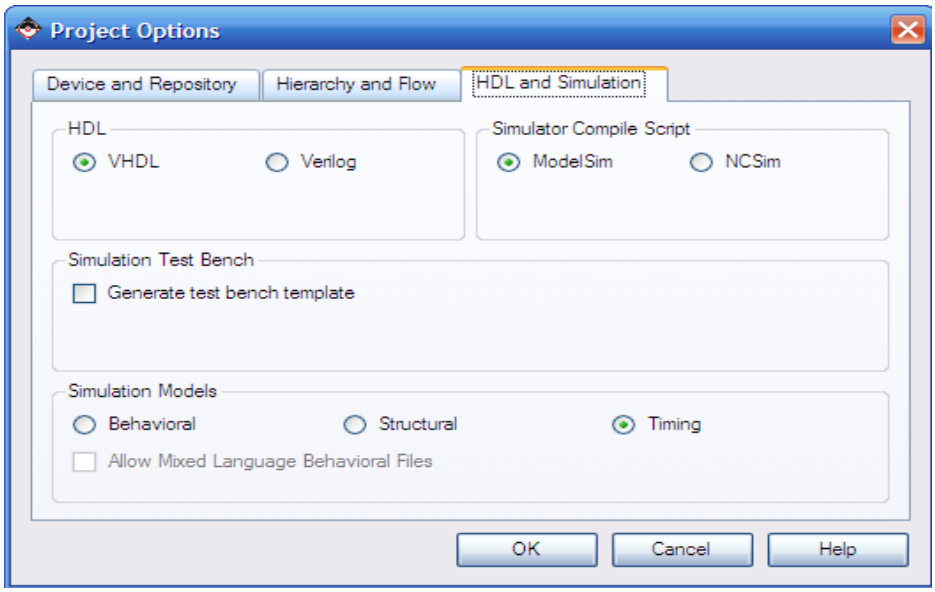



*Figura 5.13 Ventana que muestra el inicio de la prueba de tiempo para los algoritmos.*

En esta ventana se observan las opciones para realizar la prueba, se requiere especificar que simulador será el encargado de realizar la tarea. De acuerdo al funcionamiento del ISE, el simulador ModelSim es el indicado para realizar pruebas en tiempo real; así que este debe ser elegido.

Para cada algoritmo evaluado los resultados pueden ser monitoreados en la ventana de interacción. Mostrada en la figura 5.14.

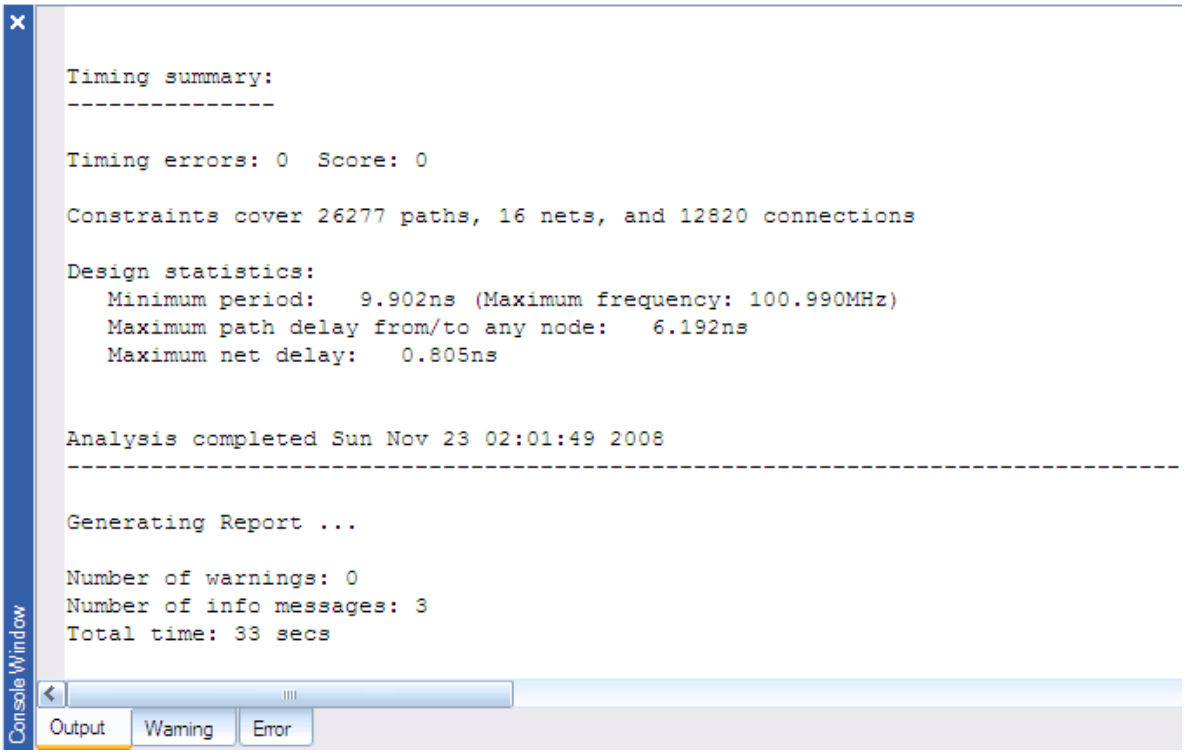



*Figura 5.14 Reporte en tiempo de la prueba realizada a un algoritmo.*

Esta ventana muestra los resultados de las pruebas realizadas a los tres algoritmos de adquisición de señales GPS, por lo que se consigue terminar la evaluación de éstos.

Al finalizar las pruebas para los tres métodos de adquisición se puede completar la tabla mostrada al final de las pruebas de simulación, finalizando el análisis de los algoritmos.

La tabla 5.3 muestra los valores finales queda de la siguiente manera:

| *Algoritmo* | *Tiempo de Ejecución* | *Complejidad* | *Repeticiones* |
|---|---|---|---|
| **Búsqueda Serial** | 1984 us | Baja | *41943* |
| **Búsqueda Paralela en el espacio de la Frecuencia** | 1845 us | Media | *1023* |
| **Búsqueda Paralela en el Espacio del Código de Fase** | 865 us | Alta | *41* |

*Tabla 5.3 Resultados obtenidos a través de los ambientes de desarrollo.*

Los valores finales muestran que el algoritmo que se ejecuta más rápidamente es el de búsqueda paralela en el espacio de la fase de código y que tarda menos de la mitad de tiempo que los otros dos algoritmos. Aunque la programación e implementación de este algoritmo es más complicada, se ha obtenido una ganancia muy importante en el tiempo de ejecución. Es importante señalar que los tres algoritmos ofrecen los mismos resultados y que la diferencia entre ellos radica en la complejidad de su implementación y la velocidad de ejecución. Estos aspectos deben ser considerados al incorporar uno de los algoritmos al receptor GPS completo.

Con la tabla completa se termina el análisis y evaluación de los algoritmos de adquisición de señales GPS. Los datos obtenidos a través del trabajo, han sido validados principalmente con la etapa de simulación en los ambientes de programación de MatLab y lenguaje C. La concordancia en los valores obtenidos, garantiza que el trabajo de implementación y desarrollo es satisfactorio. Además se observan los parámetros más significativos que se han dispuesto a evaluar, se hace evidente la funcionalidad de los algoritmos en referencia a la complejidad de implementación y al registro en tiempo real de su desempeño, aunque una métrica básica de comparación de desempeño es el tiempo de ejecución, es importante evaluar el consumo de recursos del FPGA al ejecutar estos tres algoritmos.

# CONCLUSIONES

Al final del trabajo de evaluación e implementación de algoritmos de adquisición de una señal GPS, en un dispositivo reconfigurable FPGA, se obtienen las siguientes conclusiones:

Para desarrollar la etapa de adquisición de un receptor GPS definido por software, se ha requerido del estudio del procesamiento de las señales entregadas por la constelación de satélites pertenecientes a este sistema; así como de los dispositivos encargados de adaptar a la señal en banda base para iniciar este procesamiento.

Además de obtener la información referente al sistema GPS, se requirió estudiar y documentar las características y metodología de desarrollo usando FPGA, para la implementación de los algoritmos. Las herramientas con el ambiente de desarrollo para este dispositivo se obtuvieron a través del sitio web de Xilinx.

Con la información recabada se consideraron tres diferentes algoritmos para la adquisición, que fueron evaluados a través de diferentes ambientes de programación. El primer ambiente de desarrollo fue MatLab, debido a que fueron utilizadas las funciones especializadas para el procesamiento de señales, incluidas en sus librerías de desarrollo. Después de implementar cada uno de estos algoritmos se obtuvo como resultado una lista similar de satélites adquiridos para los tres casos, por lo que en primer plano se asumió que su funcionamiento era correcto.

A partir de ser implementados en este ambiente, se inició el análisis de los algoritmos, el parámetro de referencia en esta evaluación fue tomado en función a la complejidad de desarrollo en el ambiente de programación y se tuvo el primer parámetro de análisis para reconocer las cualidades de cada uno de los algoritmos.

Para realizar una validación de los datos obtenidos por los algoritmos, se requirió de buscar información del sistema GPS al día en que se hizo este análisis. La información de referencia mostrada fue tomada del sitio web de la FAA, que aportan información casi instantánea del almanaque del GPS. Se complementó a esta verificación con la integración en simulación de etapas posteriores en el procesamiento de la señal en un receptor GPS, obteniendo la imagen del ambiente Google Earth que muestra las instalaciones del I.P.N. en donde fue tomada la muestra para su análisis. Con los datos corroborados por diversas fuentes se garantizó el correcto funcionamiento de los algoritmos.

Para continuar con la evaluación de estos métodos, se requirió presentarlos en un lenguaje compatible con el dispositivo FPGA. Al momento de estudiar se identificó que el lenguaje C, era idóneo para este trabajo, así que se requirió de implementar a estos algoritmos sobre este lenguaje. El traspaso requirió de desarrollar las funciones de MatLab especializadas a un nivel de compatibilidad con la plataforma de trabajo. Esta tarea requirió del estudio de estas funciones para lograr la adaptación al nivel de compatibilidad deseado.

La implementación en el último ambiente de desarrollo fue en el FPGA y específicamente en su procesador interno PowerPC, que es un dispositivo especializado en el procesamiento de señales. En esta plataforma se buscó realizar una evaluación en tiempo real de los algoritmos y resaltar las ventajas que implicaba utilizar a un dispositivo como el FPGA. El trabajo se desarrolló con las herramientas que ofrece Xilinx, como el SDK y XPS, donde a partir del ModelSim se realizó la prueba en tiempo para los algoritmos.

Al obtener la tabla de resultados de las tres implementaciones, se puede identificar el tiempo de ejecución de cada uno de los algoritmos, aunque debe señalarse también que no se debe perder de vista la complejidad de desarrollo de estos métodos.

Los valores en los tres algoritmos de adquisición son similares y pueden ser utilizados sin alterar el funcionamiento general y los resultados esperados. Aunque es importante tener presente que solo un algoritmo de adquisición de señales desarrollado, es utilizado dentro de los módulos de procesamiento que deberán ser ejecutados en un receptor GPS completo. Entonces se debe tener una perspectiva de los requerimientos esperados por el receptor, para saber cuál de los algoritmos debe ser utilizado.

En cuestión al trabajo realizado sobre el FPGA, se debe mencionar que el objetivo era comprobar la flexibilidad de su arquitectura, esta característica ha sido aprovechada con el hecho de implementar a los tres diferentes algoritmos sobre la misma plataforma de trabajo, sin necesidad de tener que utilizar diversos dispositivos. A esto se suma el hecho de que esta plataforma garantiza que los tiempos de ejecución del proyecto obtenidos puedan ser competitivos con otros dispositivos que se ocupan de realizar las mismas tareas.

A partir de agregar a este trabajo las etapas posteriores de procesamiento de señales, y de realizar sus funciones bajo los sistemas de posicionamiento, se obtendrá una plataforma de radio definido por software capaz de incorporarse sobre diversos sistemas de comunicaciones inalámbricas empleadas en las telecomunicaciones y realizar pruebas e innovaciones que mejoren el desempeño de éstas. Con esto se hace notar la importancia que tiene el trabajo desarrollado.

# RECOMENDACIONES Y TRABAJOS FUTUROS

La capacidad del proyecto realizado esta basada directamente en el desempeño del PPC en el FPGA, así que a medida se mantenga en forma óptima el trabajo con este dispositivo, se logrará un desarrollo ideal para el trabajo de otras etapas de procesamiento que puedan ser implementadas dentro de este componente.

Para la elaboración de este proyecto se ha requerido de los recursos del kit de desarrollo. A nivel de hardware estos recursos no han sido consumidos en gran proporción; sin embargo para el trabajo en software se ha tenido un uso de por lo menos el 50% de los recursos disponibles, para optimizar el desempeño de este proyecto, se propone probar innovaciones a la etapa de desarrollo de los algoritmos, siempre existirán caminos diferentes para plasmar las ideas dedicadas a la adquisición de señales.

A nivel de hardware se puede hacer la consideración de adaptar funciones utilizadas en software, para realizar trabajos con más paralelismo y obtener una mejora en el desempeño de su funcionamiento.

Para completar una plataforma de radio definido por software, se requiere del desarrollo e implementación de las etapas posteriores a la adquisición, sobre el mismo microprocesador PPC.

También una alternativa de desarrollo interesante es analizar las nuevas señales GPS y Galileo para identificar e implementar los ajustes a la programación de tal manera que este desarrollo pueda convertirse en un receptor compatible para ese tipo de señales.

# REFERENCIAS

[1]"United States Naval Observatory ((USNO) - Block II Satellite Information". ftp://tycho.usno.navy.mil/pub/gps/gpsb2.txt.

[2] A. Brown and J. Nordlie, "Integrated GPS/TOA Navigation using a Positioning and Communication Software Defined Radio",

[3] Software Defined Radio: Architectures, Systems and Functions,M.Milliger et al., Eds., Wiley, New York, 2003.

[4] J. Mitola et al., "Cognitive radio: Making software radios more personal," IEEE Pers. Commun., vol. 1, no. 4, pp. 13–18, Aug. 1991.

[5] J. Mitola, "Cognitive radio: An integrated agent architecture for software defined radio," Doctor of Technology, Royal Inst. Technol. (KTH), Stockholm, Sweden, 1999.

[6] A. Brown and J. Nordlie, "Integrated GPS/TOA Navigation using a Positioning and Communication Software Defined Radio", Proceedings of IEEE/ION Position Location and Navigation Symposium, PLANS 2006, San Diego, CA, Apr. 2006

[7] Software Defined Radio: Architectures, Systems and Functions,M.Milliger et al., Eds., Wiley, New York, 2003.

[8] Software Defined Radio: Origins, Drivers, and International Perspectives, W. Tuttlebee, Ed., Wiley, New York, 2002.

[9] S. Haykin, Communication Systems, 4th ed. New York: Wiley, 2007.

[10] A. Brown and J. Nordlie, "Integrated GPS/TOA Navigation using a Positioning and Communication Software Defined Radio", Proceedings of IEEE/ION Position Location and Navigation Symposium, PLANS 2006, San Diego, CA, Apr. 2006

[11] GPS Wing Reaches GPS III IBR Milestone in InsideGNSS November 10, 2008

[12]"Navstar GPS and GLONASS: global satellite navigation systems". IEEE. http://ieeexplore.ieee.org/iel1/2219/7072/00285510.pdf?arnumber=285510.

[13] NASA Global Differential GPS System
http://www.gdgps.net/

[14] "United States Updates Global Positioning System Technology". America.gov. February 3, 2006.

[15]"NAVSTAR GPS USER EQUIPMENT INTRODUCTION" (PDF). US Government. http://www.navcen.uscg.gov/pubs/gps/gpsuser/gpsuser.pdf.

[16] GPS constellation status. Russian Space Agency. April 9, 2008

[17] GPS Wing Reaches GPS III IBR Milestone in InsideGNSS November 10, 2008

[18] Dietrich Schroeer, Mirco Elena (2000). Technology Transfer. Ashgate. pp. p80. ISBN 075462045X. http://books.google.com/books?lr=&id=I7JRAAAAMAAJ. Retrieved on 2008-05-25.

[19]M.S. Braasch, A. J. van Dierendonck, "GPS
receiver architectures and measurements",
Proceedings of the IEEE, vol. 87/1, Jan. 1999.

[20]"United States Updates Global Positioning System Technology". America.gov. February 3, 2006.http://www.america.gov/xarchives/display.html?p=washfile-
english&y=2006&m=February&x=20060203125928lcnirellep0.5061609.

[21]Alison K. Brown, Lynn Stricklan, and David Babich,
"Implementing a GPS Waveform Under the Software
Communication Architecture," 2006 Software

[22]Richard Langley (July/August 1991). "The Mathematics of GPS" (PDF). GPS World. http://gauss.gge.unb.ca/gpsworld/EarlyInnovationColumns/Innov.1991.07-08.pdf.

[23] David M. Lin, James B. Y. Tsui, Dana Howell,
"Direct P(Y)-Code Acquisition Algorithm for
Software GPS Receivers", Proc. ION GPS, 1999.

[24]Software Defined Radio: Origins, Drivers, and International Perspectives,
W. Tuttlebee, Ed., Wiley, New York, 2002.

[25]J. Mitola et al., "Cognitive radio: Making software radios more personal,"
IEEE Pers. Commun., vol. 6, no. 4, pp. 13–18, Aug. 1999.

[26]J. Mitola, "Cognitive radio: An integrated agent architecture for software
defined radio," Doctor of Technology, Royal Inst. Technol. (KTH),
Stockholm, Sweden, 2000.

[27] GPS SPS Performance Standard — The official Standard Positioning Service specification (2008 version).

[28]"Navstar GPS and GLONASS: global satellite navigation systems". IEEE. http://ieeexplore.ieee.org/iel1/2219/7072/00285510.pdf?arnumber=285510.

[29] David M. Lin, James B. Y. Tsui, Dana Howell,
"Direct P(Y)-Code Acquisition Algorithm for
Software GPS Receivers", Proc. ION GPS, 1999.

[30] Sklar, Bernard. Digital Communications Fundamentals and

Applications Second Edition. Prentice Hall 2001.

[31] M.S. Braasch, A. J. van Dierendonck, “GPS receiver architectures and measurements”, Proceedings of the IEEE, vol. 87/1, Jan. 1999.

[32]David M. Lin, James B. Y. Tsui, “Comparison of Acquisition Methods for Software GPS Receiver”, Proc. of ION GPS, 2000.

[33] “Global Positioning System Interface Control Document”, ICD-GPS-200c, available at http://www.spacecom.af.mil/usspace/gps_support/ documents/ICD-GPS-200RC-004.pdf

[34] Frigo, M. and S. G. Johnson, "FFTW: An Adaptive Software Architecture for the FFT," Proceedings of the International Conference on Acoustics, Speech, and Signal Processing, Vol. 3, 1998, pp. 1381-1384.