Trabajo de Maestría:

Metodología de codiseño hardware-software para procesamiento de señales radar en sistemas embebidos

Autor: Felipe Andrés Silva Gómez

Directora: Alejandra María González Correal PhD

Mayo de 2017

Pontificia Universidad Javeriana

Contenido

1. INTRODUCCIÓN 7
2. DEFINICIÓN DEL PROBLEMA 9
3. OBJETIVOS 11
   3.1. OBJETIVO GENERAL 11
   3.2. OBJETIVOS ESPECÍFICOS 11
4. MARCO TEÓRICO 12
   4.1. Sistemas radar 12
      4.1.1. Procesamiento de señal en sistemas radar 13
   4.2. Estado del arte metodologías de codiseño hardware/software 16
      4.2.1. Metodologías basadas en modelado 17
      4.2.2. Metodologías de codiseño basadas en lenguajes de programación y/o orientadas por software 17
      4.2.3. Metodologías basadas en desarrollo ágil de software 18
      4.2.4. Mejoras a las metodologías tradicionales de codiseño 18
      4.2.5. Herramientas y algoritmos 19
      4.2.6. Casos de estudio 19
5. DESARROLLOS 20
   5.1. Análisis algoritmos de procesamiento radar 20
   5.2. Herramientas de desarrollo para codiseño hardware/software 21
   5.3. Metodología de codiseño para sistemas de procesamiento radar 22
      5.3.1. Bases de la metodología 22
      5.3.2. Flujo de diseño de la metodología COMRADES 24
   5.4. Herramienta COMRADES 27
      5.4.1. Arquitectura de la herramienta 27
      5.4.2. Modelamiento difuso de las reglas de diseño 34
   5.5. Implementación de la herramienta 38
      5.5.1. Patrón de radiación de la antena 38
      5.5.1. Interfaz gráfica 39
      5.5.2. Entorno de simulación 40
      5.5.1. Tareas de procesamiento 41
      5.5.1. Extracción de métricas 42
      5.5.2. Lógica difusa 43
      5.5.1. Codificación C y VHDL 43
   5.6. Evaluación de la metodología COMRADES 45
      5.6.1. Formulario de evaluación de metodología 46

5.6.2. Resultados de la evaluación de metodología 47
5.7. Caso de diseño de sistema de procesamiento radar mediante la metodología 50
6. ANÁLISIS DE RESULTADOS 54
6.1. Ventajas de la solución 54
6.2. Limitaciones de la solución 54
6.3. ¿Cómo se compara con otras soluciones? 56
7. CONCLUSIONES Y TRABAJOS FUTUROS 57
7.1. Trabajos futuros 58
8. BIBLIOGRAFÍA 59

Figuras

Figura 1. Esquema típico de un sistema radar primario. 7
Figura 2. Comparación flujo de diseño tradicional frente a flujo de codiseño que presentaría un ahorro en tiempo de desarrollo[1] 8
Figura 3. Diagrama típico de un sistema radar. 12
Figura 4. Búsqueda del pulso transmitido entre señal ruidosa, por medio del filtro adaptado 14
Figura 5. Concepto de procesamiento de señal durante un barrido, varios barridos y varias exploraciones (tomado de [8]) 15
Figura 6. Tendencias en el codiseño hardware-software 17
Figura 7. Arquitectura del sistema embebido objetivo para el procesamiento de señal radar 22
Figura 8. Flujo propuesto para el desarrollo de sistemas usando metodología COMRADES 24
Figura 9. Arquitectura general de la herramienta COMRADES 27
Figura 10. Herramientas utilizadas para modelado, simulación, compilación y síntesis de hardware. Todas pueden ser integradas al entorno de trabajo de Matlab 28
Figura 11. Pantalla principal de la interfaz de usuario para captura de parámetros del sistema radar 28
Figura 12. Recepción de ecos para formar la matriz radar. (a) Antes del procesamiento, es muy difícil diferenciar los blancos del ruido. (b) Luego del procesamiento, se puede diferenciar más fácilmente los blancos del ruido 29
Figura 13. Métricas de ejecución de las funciones f01 – f04, con los tiempos de ejecución y memoria 30
Figura 14. Procesamiento de la matriz radar por medio del filtro adaptado, en gráficas de acimut vs distancia, siendo el eje vertical la potencia de la señal recibida. a) Previo a la aplicación del filtro adaptado. b) Posterior a la aplicación del filtro adaptado. 31
Figura 15. El control de sensibilidad en el tiempo (STC) amplifica las señales que se encuentran más lejos y atenúa las que están más cerca del radar. 31
Figura 16. La integración de pulso suma las contribuciones de ecos del mismo blanco capturadas en pulsos diferentes. 32
Figura 17. El proceso de detección de pico consiste en analizar cuáles señales superan el plano del umbral, y en ellas hallar el punto máximo de elevación, en la gráfica presentada tan solo uno de los blancos superaría el umbral y allí se estimaría el pico para determinar el acimut del blanco. 32
Figura 18. Interfaz de codiseño del sistema 33
Figura 19. Resultado proceso de segmentación de codiseño 34
Figura 20. Funciones de pertenencia para frecuencia de repetición de pulso (PRF). Eje x en Hz. 35
Figura 21. Funciones de pertenencia para resolución en distancia. Eje x en metros. 35
Figura 22. Funciones de pertenencia para número de pulsos integrados. 36
Figura 23. Funciones de pertenencia para velocidad de la antena. Eje X en RPM. 36
Figura 24. Funciones de pertenencia para función de salida. En el eje x se presenta el grado de afinidad con hardware o software 37
Figura 26. Generación de un patrón de radiación pencil beam variable en acimut 39
Figura 27. Patrón de radiación, imagen de corte en acimut para dos diferentes anchos de antena (a) 3º (b) 15º 39
Figura 25. Entorno para creación de interfaz gráfica 40
Figura 28. Código simplificado de ciclo de simulación de blancos y rotación de antena 41
Figura 29. Llamado a las funciones de procesamiento desde el archivo principal 42
Figura 30. Uso de profiler para extracción de métricas 42
Figura 31. Uso de la herramienta Fuzzy Logic Designer 43

Figura 32. Pasos para la creación de código VHDL usando la herramienta HDL coder.....44
Figura 33. Creación de archivos C con la herramienta C Coder 45
Figura 34. Herramientas usadas por los encuestados, el listado presentado corresponde con las herramientas que tienen funcionalidades de codiseño hardware-software.....47
Figura 35. Lenguajes de programación usados por los encuestados. 48
Figura 36. Captura de requerimientos para el desarrollo de un sistema de procesamiento radar 50
Figura 37. Gráfica resultado de la simulación del sistema radar, para dos vistas diferentes. Se muestran tres picos, que corresponden a los tres blancos identificados por el radar. En a) se presenta una vista lateral, en donde se aprecia la diferencia en distancia respecto al radar de los diferentes blancos, en b) se presenta una vista frontal en donde se presenta la diferencia en acimut de los blancos. 51
Figura 38. Se presentan los resultados de las métricas de ejecución de las funciones radar: filtro adaptado, STC, integración de pulso y detección de pico.....51
Figura 39. Se presentan los resultados de las métricas de ejecución de las funciones radar: filtro adaptado, STC, integración de pulso y detección de pico.....52
Figura 40. Resultados de la evaluación para relación Velocidad/Área 52
Figura 41. Resultados de la evaluación para selección mediante barra de Velocidad y Área, priorizando la velocidad en un 75%. 53
Figura 42. Herramientas usadas en los diferentes trabajos de codiseño. a) El trabajo de Mischkalla y Mueller [57]. b) Herramienta COMRADES.....57

Tablas

Tabla 1. Descripción de técnicas de procesamiento radar 16
Tabla 2. Mapeo entre aplicaciones de sistema radar y tipos de radares usados 20
Tabla 3. Mapeo entre tipos de radares y técnicas de procesamiento digital.....21
Tabla 4. Principales herramientas encontradas para codiseño hardware software y diseño a nivel de sistema.....22
Tabla 5. Descripción de pasos de flujo para la metodología COMRADES.....27
Tabla 6. Conjunto de reglas de modelamiento difuso para asignación de funciones con modalidad hardware o software 38
Tabla 7. Cuestionario para validación de proyecto.....47

1. INTRODUCCIÓN

Históricamente, los sistemas radar han sido de vital importancia para aplicaciones militares y de aviación, debido a su capacidad de detección "todo tiempo", es decir, sin importar condiciones ambientales como día o noche, lluvia, nieve u oleaje. Esta gran ventaja hace que los radares hoy sean también usados en aplicaciones civiles como: sistemas de seguridad, sensores anticolisión para automóviles, detectores de velocidad, sistemas meteorológicos, tecnología espacial, y astronomía, entre otras. A nivel industrial y académico es una tecnología de gran interés, tanto así que la palabra "radar" se encuentra entre los 20 términos más buscados en la librería digital de la IEEE[1]. Adicionalmente, existe un auge en la implementación de soluciones usando tecnología radar debido a la baja de precio de estos sensores de radiofrecuencia, lo que da cabida a su uso en aplicaciones embebidas y de bajo costo.

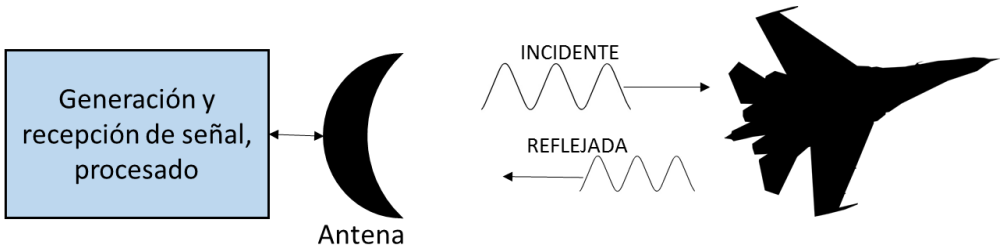


Figura 1. Esquema típico de un sistema radar primario.

En la Figura 1, se observa el esquema típico de un sistema radar. La tecnología radar está basada en la emisión de una onda electromagnética, esta onda se reflejará en los objetos que se encuentren en su camino, tanto los que se desean detectar como los que no, luego esta onda reflejada se capturará de nuevo en el sistema y a través del procesamiento se podrán separar las reflexiones deseadas (blancos), de las que no son de interés (*clutter*). Tradicionalmente, todos los componentes de un radar, que van desde la generación de la señal hasta el procesamiento, han sido analógicos, sin embargo, con la evolución de los sistemas digitales se han ido transformando para usar la señal digital tanto como sea posible. Anteriormente sólo se implementaba en tecnología digital el módulo de procesado de señal, hoy en día los sistemas de procesamiento digital de señal radar deben realizar tareas como la generación digital de la señal, modulación y demodulación en frecuencia intermedia, filtro adaptado, entre otros. Todas estas tareas deben realizarse bajo un requisito de tiempo real. Estas tendencias hacen que actualmente el núcleo y factor diferencial de un sistema radar sea su procesamiento digital.

Para la implementación de un sistema de procesamiento digital de señal radar, en la práctica se suelen usar dispositivos electrónicos programables por software como microprocesadores, microcontroladores, DSPs; así como dispositivos de hardware configurable como FPGAs y

[1] Año 2016

CPLDs. Una de las principales ventajas de los dispositivos de hardware configurable es la paralelización de los cálculos, por lo que se hace útil en aplicaciones que requieren alta velocidad, mientras que los dispositivos programables por software ofrecen flexibilidad. Dependiendo de las características del sistema de procesamiento que se diseñe, se usa hardware configurable para ciertas funciones y dispositivos programables por software para otras. En sistemas radar con requerimientos exigentes en tiempo y cantidad de datos, como en el caso de las aplicaciones militares, se usa una combinación de ambos tipos de dispositivos.

La decisión con respecto a la segmentación, es decir, qué funciones de procesamiento se realizan en componentes de hardware configurable, y qué funciones llevar a cabo en dispositivos programables por software, no es una decisión trivial, pues, puede tener implicaciones en el tiempo de desarrollo y los costos asociados. Teniendo en cuenta esta premisa, las metodologías de codiseño hardware/software, o metodologías de diseño a nivel de sistema, han propuesto un flujo de desarrollo que consiste en diseñar de forma simultánea los componentes hardware y software de un sistema usando un modelado de alto nivel para luego tomar la determinación con respecto a la segmentación. En la Figura 2 se presenta el diagrama comparativo entre las metodologías de diseño tradicionales y el codiseño.

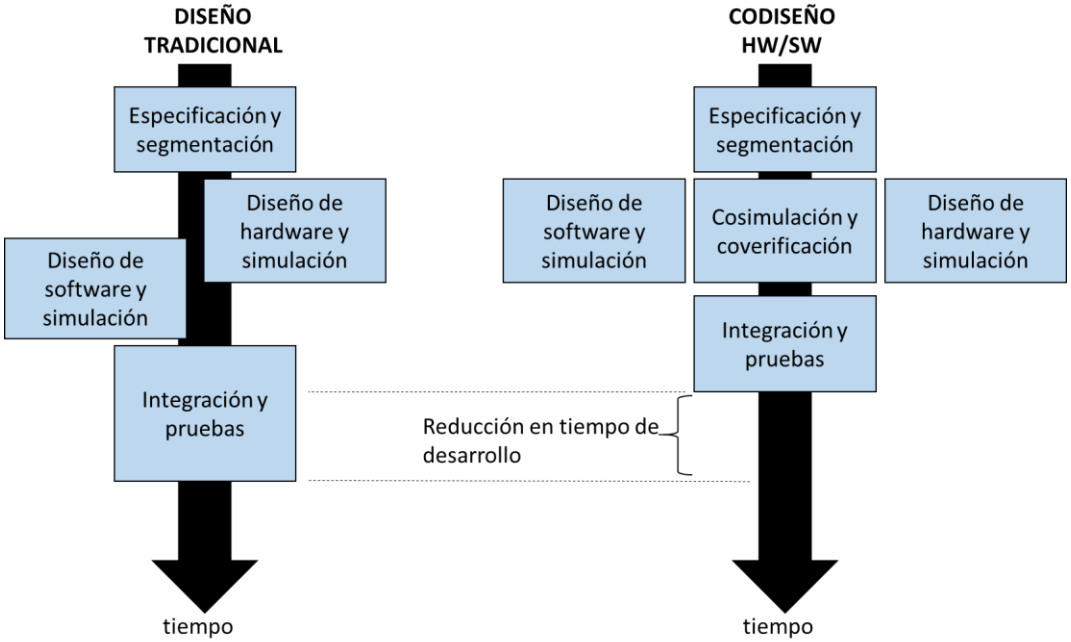


Figura 2. Comparación flujo de diseño tradicional frente a flujo de codiseño que presentaría un ahorro en tiempo de desarrollo[1].

A pesar de la tendencia de la evolución del diseño de sistemas embebidos hacia codiseño hardware/software o ESL (Electronic System Level) *design*, su uso aún no está extendido por parte de los desarrolladores de sistemas embebidos [2]. Esto se debe, en parte, a que, a pesar de presentar un modelo eficiente, existe una dificultad técnica en su implementación, pues requiere de nuevas herramientas de desarrollo e implica un cambio de paradigma que conlleva a una curva de aprendizaje y sus tiempos asociados.

Con el propósito de acortar la brecha entre las metodologías de diseño a nivel de sistema y su implementación real, en la presente investigación se propone una metodología de codiseño hardware/software o ESL, aplicada a un campo particular del desarrollo de sistemas de procesamiento radar, campo en el que no se han encontrado estudios ni trabajos de referencia previos y que podría ser utilizada por la creciente industria de desarrollo de soluciones usando tecnología radar.

2. DEFINICIÓN DEL PROBLEMA

Las metodologías de codiseño hardware/software o diseño a nivel de sistema electrónico (ESL) plantean una mejora en el tiempo de desarrollo de un sistema embebido frente al modelo convencional, sin embargo, su uso aún no está extendido en la industria. Al ser un cambio de paradigma, la adopción de estas metodologías, conlleva tiempo de apropiación, debido a que esta perspectiva puede implicar cambios en la filosofía de diseño y métodos de programación. Esta nueva tendencia de codiseño se enfoca en descripción, diseño y modelado, más que simplemente en programación de un sistema embebido. Debido al codiseño, hoy en día, existen variedad de lenguajes de alto nivel o métodos de especificación gráfica, así como diferentes herramientas de diseño, que no son de uso habitual por parte de desarrolladores de sistemas embebidos. Esta variedad y falta de estandarización en los métodos de entrada, representa un obstáculo para la adopción de sistemas embebidos en la industria [3].

Existe una tendencia creciente en el uso de sensores tipo radar, tecnología que anteriormente sólo era usada para aplicaciones militares, ahora gracias a la miniaturización y masificación de sus de sus componentes, es usada en productos de bajo costo como sistemas de detección de intrusos y sensores anticolisión, entre otros [4]. El procesamiento de estas señales radar, presenta un reto pues requiere de una variedad de algoritmos que por lo general tienen requisitos de ejecución en tiempo real. Los subsistemas radar de transmisión y recepción en radiofrecuencia se ofrecen habitualmente como dispositivos COTS (Commercial off-the-shelf), mientras que el procesamiento de señal radar, es el encargado de brindar valor agregado según la aplicación específica, representando por lo tanto el núcleo de la tecnología.

Un sistema de procesamiento de señales radar, puede implicar la combinación de tecnologías de procesamiento digital basadas en hardware reconfigurable como FPGA [5], así como en dispositivos programables por software como microprocesadores o DSP. La ventaja principal de los dispositivos hardware está en la capacidad de paralelización, lo que agiliza los tiempos de cómputo frente a los dispositivos programables por software que realizan las tareas de forma secuencial. Como inconveniente está que los dispositivos hardware al hacer tareas concurrentes también requerirán mayor área o cantidad de recursos como celdas lógicas y puertos de entrada/salida. En el diseño de un sistema de procesamiento radar particular, se habrá de elegir el mejor esquema de segmentación, es decir, decidir qué funciones del sistema se ejecutarán en hardware y qué funciones en software [6]. Esta decisión de segmentación no es trivial ya que tiene una influencia directa en el tiempo de desarrollo y costos del sistema.

En la revisión bibliográfica, se han encontrado diversos esquemas de metodologías y herramientas de codiseño y desarrollos en procesamiento de señales radar, sin embargo, no se han encontrado

aportes metodológicos de codiseño hardware/software orientados a sistemas de procesamiento radar, que corresponde al campo que se aborda en el presente proyecto.

3. OBJETIVOS

3.1. OBJETIVO GENERAL

- Proponer una metodología de codiseño hardware/software aplicada a sistemas de procesado de señales radar.

3.2. OBJETIVOS ESPECÍFICOS

- Identificar los diferentes elementos presentes en el desarrollo de un sistema de procesado radar y las metodologías y herramientas de codiseño.
- Evaluar diferentes herramientas de desarrollo disponibles mediante la implementación de módulos de procesamiento radar.
- Diseñar la metodología de codiseño aplicada a procesamiento radar y su método de validación.
- Realizar una validación de la metodología propuesta para evaluar sus beneficios frente a metodologías convencionales.

4. MARCO TEÓRICO

4.1. Sistemas radar

La Unión Internacional de las Telecomunicaciones (UIT) define al radar como "Sistema de radiodeterminación basado en la comparación entre señales radioeléctricas reflejadas o retransmitidas desde la posición a determinar". La palabra radar corresponde al término en inglés *Radio Detection And Ranging*, que se puede entender como detección y medición de distancia por medio de ondas radio. A pesar que el término se ha mantenido, hoy en día los radares pueden entregar mucha más información que la detección y la distancia, como: posición, velocidad, altitud, tipo de blanco, entre otros.

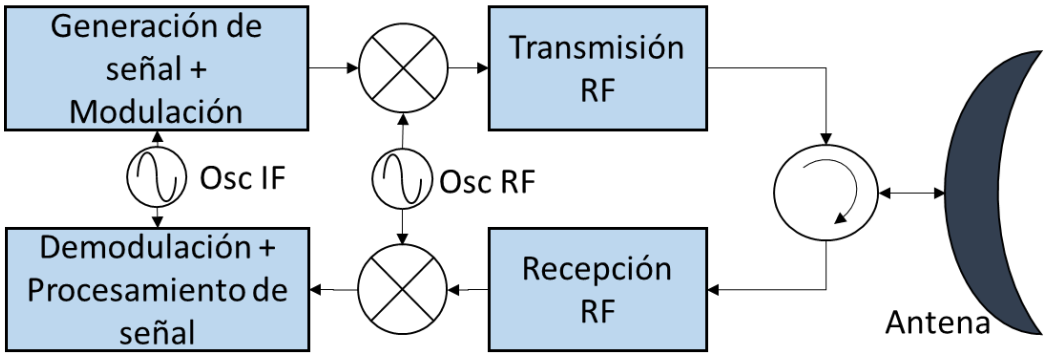


Figura 3. Diagrama típico de un sistema radar.

En la Figura 3 se presenta un esquema típico de radar. La señal se genera a una frecuencia intermedia (IF) en el orden de los *MegaHertz*, luego se sube a la frecuencia de operación del sistema (RF) en el orden de los *GigaHertz*, allí se amplifica y se envía a la antena. Los ecos de la señal de radiofrecuencia que han impactado en el blanco y en otros objetos se reciben a través de la antena, allí se amplifican y se bajan a IF. Con la señal en IF se realiza la demodulación de la señal, a la que se le aplica el procesamiento, este es el encargado de emplear las técnicas necesarias para extraer información sobre los blancos. La información de los blancos, puede ser almacenada, transferida o presentada al usuario a través de una interfaz gráfica [7].

Los principales retos que plantea la señal radar para la extracción de información, son por una parte el ruido, que está presente en todos los dispositivos electrónicos, y por otra parte el *clutter* que corresponde a ecos recibidos de blancos no deseados [8]. En la práctica, el *clutter* presente como las montañas, árboles y construcciones, dificultan la extracción de los blancos, puesto que típicamente presentan mayor sección radar que los blancos, lo que se traduce en ecos de señales de mayor amplitud que la proveniente de los objetos de interés. Para eliminar el *clutter*, se usan técnicas digitales que aprovechan las diferencias presentes entre el movimiento de los blancos y la quietud del entorno, o en términos generales, la diferencia entre las características del blanco y el *clutter* de fondo.

Para el procesamiento de señal radar existen varias alternativas, como los arreglos de circuitos analógicos, o la tecnología digital que es ampliamente usada hoy en día. Los principales componentes digitales usados para el procesamiento de señales son los microprocesadores y las

FPGA (*Field Programmable Gate Array*) [5], [9]–[11]. De los primeros, se destaca su versatilidad, costo y tamaño reducido, mientras que, de las FPGAs se destaca su capacidad de procesado en paralelo, que le permite tener un buen rendimiento en velocidad al realizar diversas operaciones de forma simultánea. En general, no se puede admitir que una tecnología sea mejor que la otra, debido a que depende de la aplicación que se realice, pero la tendencia es la implementación de tecnologías hibridas, aprovechando las ventajas de cada tipo de dispositivo.

En el estado del arte, se encuentran investigaciones y desarrollos recientes en el periodo de 2009 a 2016 de procesamiento radar sobre tecnologías FPGA [5], [9]–[12]. Esta tendencia indica la importancia del procesamiento hardware para acelerar los algoritmos necesarios en el tratamiento de la señal radar más allá del uso de los microprocesadores y DSP. Desde el punto de vista del codiseño hardware/software aplicado a procesamiento radar, los trabajos recientes, se enfocan principalmente en reportes de casos de estudio de sistemas desarrollados [13], [14]. Sin embargo, hasta el momento de desarrollo del proyecto (2016-2017), no se encuentra información sobre aportes metodológicos al procesamiento radar desde el punto de vista del codiseño hardware/software, el cual es el campo en el que se enfoca el proyecto.

4.1.1. Procesamiento de señal en sistemas radar

El procesamiento de la señal radar, es el que nos permite extraer información útil de las señales recibidas. En los sistemas radar, la señal debe viajar desde el transmisor hasta el blanco y regresar al sistema, esto genera que el alcance de un sistema radar sea proporcional a la raíz cuarta de la potencia transmitida [15], lo que implica una rápida atenuación de la señal, que se traduce en señales recibidas ruidosas y difíciles de diferenciar del ruido. El procesamiento de señal, por lo tanto, debe usar todas las información posibles que le permitan extraer dicha información: Por una parte aprovechar el hecho de que se conoce la forma de la señal esperada, adicionalmente tener en cuenta las características del blanco (velocidad, sección radar) permiten generar filtros que lo diferencien del ruido de fondo, por último, los valores estadísticos del ruido y el *clutter* para la aplicación específica permiten sintonizar las técnicas para mejorar la relación señal a ruido

En la Figura 4 se presenta un ejemplo de procesamiento de señal radar mediante el uso de filtro adaptado. En la parte superior se presenta el pulso transmitido por el radar modulado en frecuencia. En la segunda imagen se observa el eco que recibe el radar, en esta señal es muy difícil poder diferenciar la señal del ruido. En la imagen inferior se observa que, una vez se aplica el filtro adaptado, es posible obtener una representación que nos indica el lugar en el que se identificó el pulso transmitido dentro de la señal recibida, de esta información se puede extraer la posición del blanco.

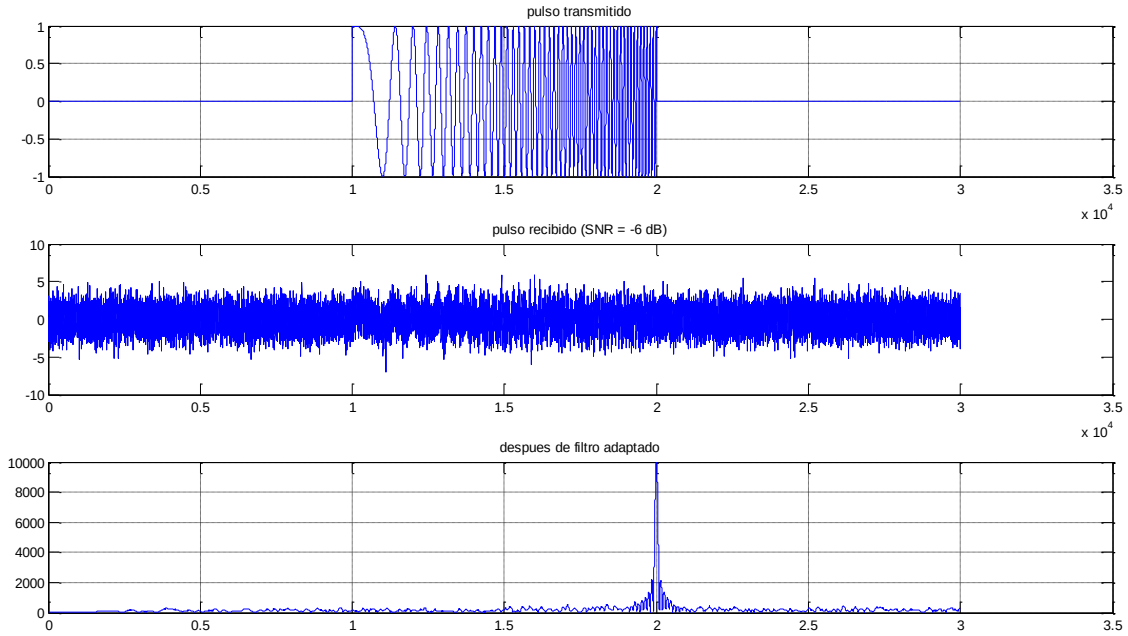


Figura 4. Búsqueda del pulso transmitido entre señal ruidosa, por medio del filtro adaptado

Existe una amplia variedad de técnicas de procesamiento aplicables a un sistema radar. Algunas de estas técnicas, requieren ser aplicadas a cada uno de los ecos de la señal recibida, es decir, en "tiempo rápido", mientras que otras se aplican combinando información de varios ecos recibidos. En un esquema típico de vigilancia, se tiene un sistema radar cuya antena tiene un movimiento giratorio que le permite explorar en varias direcciones, en este caso, cada una de las señales de ecos recibidos se va almacenando, conformando así un matriz radar de "datos crudos" a los que posteriormente se les aplicará procesamiento. En la Figura 5 se observa la matriz radar con los diferentes tipos de procesamiento: por barrido (señal de ecos de un pulso), procesado de varios barridos (acimut), procesado entre varias exploraciones.

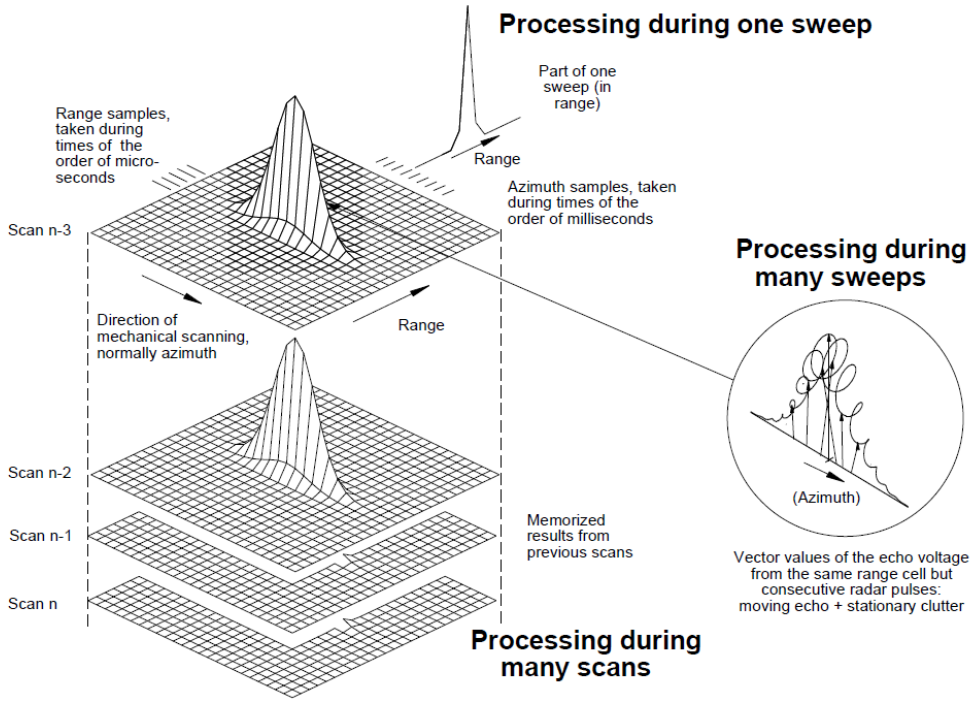


Figura 5. Concepto de procesamiento de señal durante un barrido, varios barridos y varias exploraciones (tomado de [8])

En la Tabla 1 se resumen algunas de las principales técnicas de procesamiento radar, tomando en cuenta el alcance en la matriz radar que tengan.

Técnica	Descripción	Alcance
Filtro adaptado	El filtro adaptado maximiza la relación señal a ruido, teniendo en cuenta que se conoce la forma de la señal esperada. Se realiza aplicando una convolución con el conjugado complejo de la señal transmitida.	Barrido
Detección de envolvente	Convierte la señal de una frecuencia intermedia en señales de video clásicas como en un receptor superheterodino. La señal de frecuencia intermedia tiene modulación de amplitud y modulación de fase. Puede hacerse detección coherente o detección no coherente [8].	Barrido
STC (*Sensitivity Time Control*)	Es una técnica que sirve para evitar errores de saturación debido al gran margen dinámico de la señal. Actúa con un control logarítmico atenuando en mayor medida las señales que se encuentren muy cerca al sistema radar.	Barrido
Función de enventanado	El enventanado en el tiempo permite mejorar los problemas ocasionados por el alto nivel de los lóbulos laterales presentes en el espectro de frecuencia de la señal.	Barrido
CFAR (*Constant False Alarm Rate*)	Genera una medición de los niveles de ruido para ajustar el nivel de umbral sobre el cual se deben aprobar las detecciones. Estos umbrales se asignan de tal forma que se preserve la tasa de falsa alarma del sistema.	Varios barridos
Integración de pulsos	Consiste en aprovechar el hecho que se pueden recibir varios ecos del mismo blanco, de tal forma que se suman sus aportes para mejorar el nivel de la señal sobre el ruido	Varios barridos

Integración binaria	Consiste en integrar blancos que ya han pasado el umbral de recepción de forma que se mejore la probabilidad de falsa alarma	Varios barridos
Staggering	Consiste en el cambio de la frecuencia de repetición de los pulsos, lo que permite en la recepción de señales que pueden interpretarse con ambigüedad en distancia	Varios barridos
Procesamiento *Doppler*	Utiliza el efecto *Doppler* que implica variaciones en la frecuencia de la señal recibida de acuerdo a la velocidad radial de los blancos, para determinar esto, se analiza la variación de la fase de la señal recibida	Varios barridos
MTD (*Moving Target Detection*)	Genera un banco de filtros con el propósito de identificar la velocidad de los blancos y poder así diferenciarlo del clutter. Para esto realiza análisis en frecuencia de la señal	Varios barridos
Estimación de acimut	Permite identifica el acimut sobre el cual fue recibida la detección. La tipología más típica para extracción de acimut es la detección de ventana mediante acumulador. La técnica consiste en recorrer la matriz de detecciones a un intervalo constante, variando acimut con una ventana de tamaño N. N corresponde al número de celdas en acimut que ocuparía un haz de la antena.	Varios barridos
Scan Conversion	Consiste en convertir la información de datos radar de coordenadas polares, a coordenadas cartesianas para presentarse en una pantalla. El problema es que no hay correspondencia de un pixel a un pixel en los esquemas de presentación.	Exploración
MTI (*Moving Target Indicator*)	Proceso que realiza una comparación de exploración a exploración para poder filtrar detecciones que no estén en movimiento, que usualmente corresponden a *clutter*	Varias exploraciones

Tabla 1. Descripción de técnicas de procesamiento radar

4.2. Estado del arte metodologías de codiseño hardware/software

La innovación constante en los sistemas digitales ha generado la necesidad de una mejora continua en las herramientas que facilitan su diseño. Por esta razón, desde comienzos de la década de los noventas se viene trabajando en la disciplina del codiseño hardware/software para poder manejar la complejidad asociada a las nuevas tecnologías[3],[6]. Estas herramientas en general, buscan realizar el diseño de un sistema, un escalón por encima del software y del hardware, es decir, con un mayor nivel de abstracción. De la misma forma que al abordar el diseño de un circuito, no se piensa inmediatamente en si se van a usar resistencias o condensadores, sino que primero se analiza y modela el problema hacia la búsqueda de una arquitectura, para luego seleccionar los componentes, así mismo, la idea del codiseño es definir, modelar y describir un sistema electrónico para luego definir su arquitectura, independientemente que se desarrolle en hardware, en software o en ambos. Hoy en día, el codiseño hardware-software también se relaciona con el término ESL *design (Electronic System Level design*) que consiste en una metodología que se enfoca en diseñar desde el mayor nivel de abstracción de un sistema electrónico, hasta el nivel de componente. En la Figura 6 se presenta un resumen de las tendencias en codiseño hardware software.

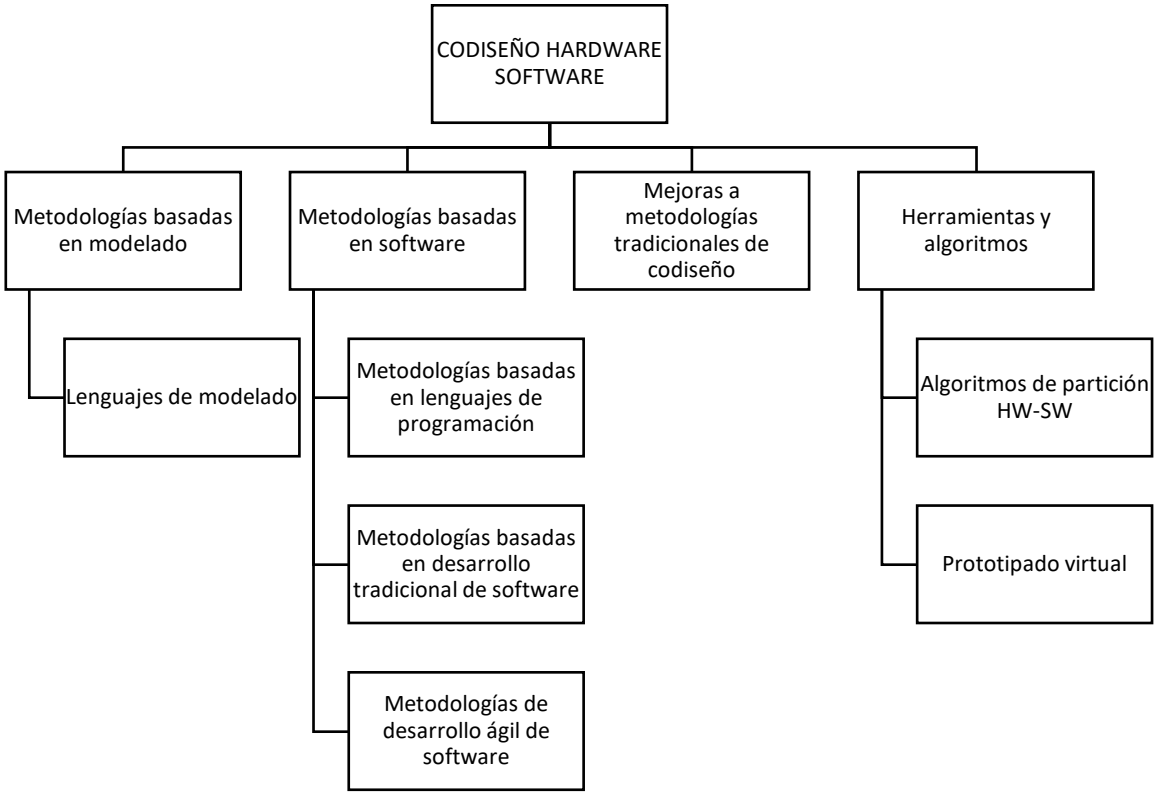


Figura 6. Tendencias en el codiseño hardware-software

4.2.1. Metodologías basadas en modelado

Corresponden a metodologías que se basan o incluyen una fase de modelado, utilizando herramientas estándar, como por ejemplo UML (*Unified Modeling Language*) [16], [17]. Dicha herramienta se utiliza para modelar el comportamiento completo del sistema para más adelante tomar la decisión de segmentación de diseño, bien sea de forma automática o a criterio del diseñador [18]. Están basadas en el concepto de *Model Driven Engineering* (MDE) el cual es conocido por ser una metodología de desarrollo de software conducida por el modelado [19]. Para codiseño hardware-software, se usa el perfil de UML orientado a sistemas en tiempo real conocido como UML-MARTE (*Modeling and Analysis of Real-Time and Embedded Systems*).

4.2.2. Metodologías de codiseño basadas en lenguajes de programación y/o orientadas por software

Así como existen metodologías basadas en modelado, también existen las que se basan en un lenguaje determinado, existen casos en los cuales se trata de un nuevo lenguaje de programación que permita abordar la complejidad a nivel de sistema, o los que se basan en lenguajes tradicionales de software y agregan extensiones o herramientas para llevarlos a hardware.

Teniendo en cuenta que el software presenta un mayor nivel de abstracción que el hardware, se proponen metodologías de codiseño en las que el hardware sólo se usa cuando se requiere la aceleración de alguno de los procesos software[20], [21]. Para ello se requiere alguna interfaz que permita comunicar el software con el hardware, en muchos casos también se proponen lenguajes como herramientas para solucionar problemas de segmentación (*Language Based Specification*).

Un mecanismo es usar especificaciones de lenguajes de diseño de interfaz (IDL, *Interface Design Language*) [20]. Se proponen también metodologías que hacen uso de herramientas de prototipado virtual para una validación temprana de la partición del diseño [22].

La compañía IBM, por su parte, se encuentra en el desarrollo de un nuevo lenguaje de programación llamado *Lime* para tratar de direccionar el problema planteado por el codiseño. *Lime* sería un lenguaje de desarrollo compatible con Java y tendría características que facilitarían la síntesis de hardware de alto nivel [23, p. 32].

Adicionalmente, se propone también como solución el lenguaje de programación BCL (*Bluespec Codesign Language*), el cual es una extensión del lenguaje (*Bluespec SystemVerilog*) que le permite al diseñador realizar la partición de hardware o software en el código fuente[24], [25].

Otro concepto originalmente del software que se pretende orientar a codiseño es el de Hardware API (*Application Programming Interface*), la cual sería una representación de los elementos de hardware a un nivel software. La idea es realizar un análisis estático de código para poder generar una estructura de hardware[26], [27].

4.2.3. Metodologías basadas en desarrollo ágil de software

Una tendencia en el desarrollo de software es el reemplazo de metodologías tradicionales (requerimientos, diseño, implementación, pruebas, despliegue) por metodologías que permitan llegar más rápido al mercado y permitir más flexibilidad en el desarrollo. Se propone entonces la forma de migrar estas tecnologías para hacerlas extensibles al desarrollo de sistemas embebidos hardware-software[28]. Metodologías de diseño ágiles como SCRUM se plantean para su implementación en codiseño [29].

Se ha encontrado que las metodologías ágiles son compatibles con el desarrollo de sistemas embebidos, pero aún deben adaptarse a las restricciones que plantean este tipo de sistemas, que no son habituales en el desarrollo de software de alto nivel [30].

4.2.4. Mejoras a las metodologías tradicionales de codiseño

El codiseño hardware software se ha estudiado y desarrollado por más de dos décadas, por lo que ya se ha establecido una base de trabajo [31]. Por lo tanto, un número de estudios relacionados con codiseño no plantean grandes cambios estructurales, sino mejoras o inclusión de nuevos elementos.

Un campo que se encuentra en algunos estudios, es el de particularizar el codiseño a un campo de aplicación específico, como el caso de una metodología de codiseño orientada al procesamiento digital de señales, en esta metodología se menciona el hardware el software y el firmware a través de las diferentes etapas de desarrollo: requerimientos, desarrollo de arquitectura, segmentación, mapeado, síntesis integración y pruebas[32].

Una de las mejoras a las metodologías actuales en relación al tiempo de diseño y verificación es el uso de componentes hardware débilmente programables (*weakly programmable*) que pueden mejorar el desempeño y agilizar el proceso de desarrollo[33]. Estos componentes débilmente programables, pueden verse como procesadores implementados en FPGA, con un set de instrucciones muy reducido.

En los estudios, se ha encontrado, una gran dependencia existente entre las herramientas de diseño con los productos que se obtienen. Por esta razón se propone que las herramientas de diseño sean vistas junto con el sistema embebido como un todo [34] en la fase de diseño, modificando las metodologías de desarrollo.

A futuro, se prevén también: la integración del mundo analógico al codiseño, codiseño orientado a la modificación de la arquitectura de los procesadores para su optimización, codiseño de sistemas adaptativos en tiempo de ejecución [31].

4.2.5. Herramientas y algoritmos

Las herramientas usadas para el diseño e implementación de sistemas embebidos han demostrado tener tanta o igual importancia que los sistemas a desarrollar, por lo tanto, un amplio campo de investigación son algoritmos y mejoras en los procesos de automatización de estas herramientas[35], [20].

El concepto de prototipado virtual se presenta como una metodología para la validación temprana de segmentación hardware-software [22]. Se ha propuesto una plataforma virtual para el diseño y simulación de sistemas embebidos en Linux [36].

A nivel de algoritmos, se han usado sistemas basados en búsquedas binarias y algoritmos genéticos para optimizar el proceso de segmentación hardware software [37]. También se proponen técnicas SMT (*Satisfiability Modulo Techniques*) para esta segmentación [38]. El concepto de agentes BDI (*Belief, Desire, Intention*) también se ha estudiado para la toma de decisiones en diseños hardware/software [39].

4.2.6. Casos de estudio

Esta categoría hace referencia a investigaciones que ejemplifican casos de estudio para codiseño de sistemas hardware software, algunos de los trabajos de interés corresponden a uso del codiseño para:

- Desarrollo de procesador y memoria caché de IBM [40].
- Implementación de algoritmo FFT en microprocesador y FPGA [41].
- Sistema de seguimiento de objetos en visión por computador [42].
- Aplicaciones de video [33].
- Procesadores digitales de video [43].

5. DESARROLLOS

En el contexto del proyecto de desarrollo de una metodología de codiseño hardware/software para sistemas de procesamiento radar, se presenta en este capítulo, inicialmente la identificación de los principales algoritmos de procesado de señal usados para diferentes tipos de aplicaciones radar, adicionalmente, se describe la base de la metodología, con la cual se exponen sus conceptos básicos, su representación y descripción, por otra parte, se presenta la herramienta de validación creada para llevar a la práctica la metodología, y en la parte final se presenta la forma como se evalúa la efectividad de la metodología.

5.1. Análisis algoritmos de procesamiento radar

El proyecto COMRADES (*Codesign Methodology for Radar in Embedded Systems*), busca sentar bases para el uso de herramientas de codiseño hardware/software, en el diseño de sistemas de procesado de señal radar, en la industria. Para ello, es de importancia identificar las principales aplicaciones, tipos de sistemas radar y algoritmos usados, de tal forma que sirva como base para la construcción y mejora de la herramienta de diseño.

En la Tabla 2 se identifican las principales aplicaciones de sistemas radar, y se presentan los tipos de radar asociados.

APLICACIONES	TIPOS DE RADAR
Seguridad vigilancia	Radar pulsado Radar pulsado Doppler Radar de onda continua con modulación en frecuencia
Seguridad seguimiento	Radar de onda continua con modulación en frecuencia (seguimiento)
Seguridad presencia	Radar de onda continua
Tránsito velocidad	Radar de onda continua
Radioaltímetro	Radar de onda continua con modulación en frecuencia
Automóviles anticolisión	Radar de onda continua con modulación en frecuencia
Detección de obstáculos	Radar de onda continua con modulación en frecuencia
Detección a través de paredes	Radar de onda continua con modulación en frecuencia
Detección de minas	Radar de onda continua con modulación en frecuencia
Pronostico del clima, detección de nubes	Radar pulsado Radar pulsado Doppler
Radares imagen (SAR, ISAR)	Radar pulsado Doppler + Procesado SAR (ISAR)

Tabla 2. Mapeo entre aplicaciones de sistema radar y tipos de radares usados

Para cada uno de los tipos de radar mostrados en la tabla anterior, existe un conjunto de técnicas de procesamiento digital de señal aplicables a cada tipo. Las principales técnicas de procesamiento se presentan en la Tabla 3.

Tipo de radar	Principales Técnicas de Procesamiento Digital
Radar pulsado	Generación de la señal, Filtro adaptado, MTI no coherente, CFAR, Log FTC, Integración de pulsos, STC, *staggering*, *Tracking*, *scan conversion*
Radar pulsado *Doppler*	Generación de la señal, compresión de impulsos, Filtro adaptado, MTI coherente, (Procesamiento digital de pulsos) MTD, CFAR, Procesador monopulso, Integración de pulsos, Log FTC, STC, *staggering*, *Tracking*, *scan conversion*
Radar de onda continua	Generación de la señal, DFT, CFAR, Integración de pulsos, t*racking*
Radar de onda continua con modulación en frecuencia	Generación de la señal, filtro adaptado, DFT, CFAR, Integración de pulsos, *tracking*, *scan conversion*
Radar de onda continua con modulación en frecuencia (seguimiento)	Generación de la señal, filtro adaptado, DFT, CFAR, Integración de pulsos, seguimiento angular, seguimiento en distancia, *scan conversion*

Tabla 3. Mapeo entre tipos de radares y técnicas de procesamiento digital

5.2. Herramientas de desarrollo para codiseño hardware/software

En el contexto del proyecto COMRADES se identifican las principales herramientas de codiseño hardware/software o diseño a nivel de sistema electrónico (*Electronic System Level*, *ESL*) presentes. Se ha encontrado en la revisión de bibliografía, la alta incidencia que tienen éstas en los tiempos y los flujos de desarrollo [35]. Las principales herramientas que han sido identificadas se resumen en la Tabla 4.

Nombre de la herramienta	Descripción	Lenguaje de entrada	Tipo de licencia
Artisan Studio [44]	Herramienta para simulación a nivel de sistema, basada en desarrollo orientado por modelo	SysML	Pago
Space studio codesign [45]	Herramienta para codiseño hw/sw que permite generación automática de código	C, C++	Pago
Matlab + HDL coder + C coder [46]	Herramienta que a partir de lenguaje Matlab genera código en C y en VHDL	Matlab	Pago / Académica
Vivado HLS (High Level Synthesis) [47]	Herramienta del fabricante Xilinx para desarrollo sobre FPGA	C, C++, SystemC	Pago / Libre con restricciones
Open Virtual Platform [48]	Herramienta para virtualización y simulación de sistemas, se usa principalmente para múltiples procesadores. No permite generación automática de código	C, C++, SystemC	Libre
SolidThinking Embed [49]	Entorno visual para desarrollo basado en modelo para sistemas embebidos. Generación de código en C, a partir de entrada visual	Bloques visuales, UML	Pago
Mentor Graphics Vista [50]	Prototipado virtual usando diseño a nivel de sistema electrónico ESL	Modelo a nivel de	Pago

Nombre de la herramienta	Descripción	Lenguaje de entrada	Tipo de licencia
		transacción TLM	
ARM SoC designer [51]	Plataforma para la creación de prototipos virtuales	Gráfico, VHDL, Verilog, System C	Pago
SystemVue Electronic System Level (ESL) Design Software [52]	Herramienta de la empresa Keysight, para diseño a nivel electrónico enfocado en la parte de radiofrecuencia y comunicación inalámbrica	C++	Pago

Tabla 4. Principales herramientas encontradas para codiseño hardware software y diseño a nivel de sistema

5.3. Metodología de codiseño para sistemas de procesamiento radar

5.3.1. Bases de la metodología

La metodología COMRADES ha sido desarrollada teniendo en cuenta las técnicas de codiseño hardware/software y ESL (*Electronic System Level*) aplicando estos conceptos en el desarrollo de sistemas de procesamiento radar.

El proyecto contempla sistemas con arquitectura de un único procesador, así como disponibilidad de hardware reconfigurable, como la que se plantea en la Figura 7. Es importante notar que también se podría contemplar un sistema FPGA con un *soft core* embebido. La implementación de un mayor número de procesadores aportaría una complejidad adicional elevada, teniendo en cuenta la necesidad de sincronización entre tareas, por esta razón no se contempla en el alcance del proyecto.

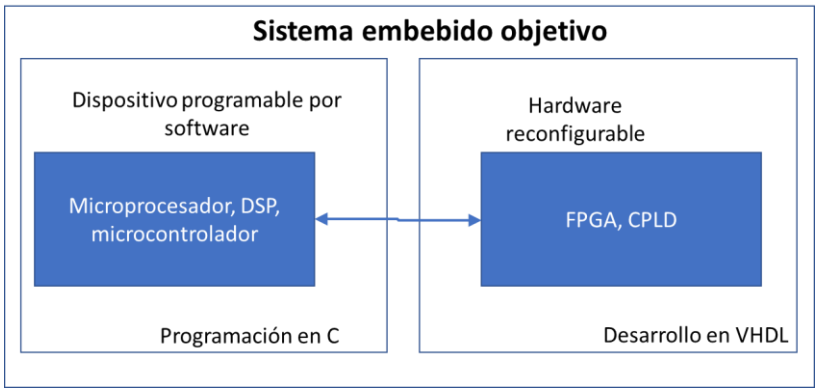


Figura 7. Arquitectura del sistema embebido objetivo para el procesamiento de señal radar

A continuación, se mencionan y describen las dos bases fundamentales de la metodología COMRADES.

Particularizar las metodologías de codiseño hw/sw al caso de procesamiento de sistemas radar

La premisa con respecto a esta primera base de la metodología es: “El hecho de conocer de antemano los diferentes algoritmos que puede tener un sistema radar, ofrece una ventaja con respecto a esquemas de segmentación generales”. Existen muchos métodos de segmentación

Las metodologías de codiseño ofrecen ventajas en tiempo de diseño y facilidad en el desarrollo con respecto a procesos de desarrollo convencionales, sin embargo, aún se presentan limitaciones para poder ser aplicadas en sistemas reales de la industria. Una base entonces de la metodología propuesta es poder solventar algunas limitaciones al particularizarla al caso de procesamiento de sistemas radar, buscando con esto hacerla más fácilmente aplicable a la industria.

En el caso de un sistema de procesamiento radar, se sabe que el cambio en las especificaciones del sistema, tendrá un impacto bien sea en los algoritmos a utilizar o en el desempeño de los mismos, ocasionando que puedan requerir más o menos recursos, por lo tanto, alterando la segmentación hardware/software que debe tener el sistema.

La decisión con respecto a la segmentación HW/SW se puede tomar con respecto a las métricas y con respecto a información experta

Una segmentación adecuada para un sistema de procesamiento radar, la puede dar un diseñador con suficiente experiencia, es por esto que la metodología captura dicha experiencia en forma de reglas, lo que se modela mediante lógica difusa. Adicionalmente, basándose en métricas de desempeño como tiempo de ejecución, memoria y área, es posible determinar un esquema adecuado de segmentación.

5.3.2. Flujo de diseño de la metodología COMRADES

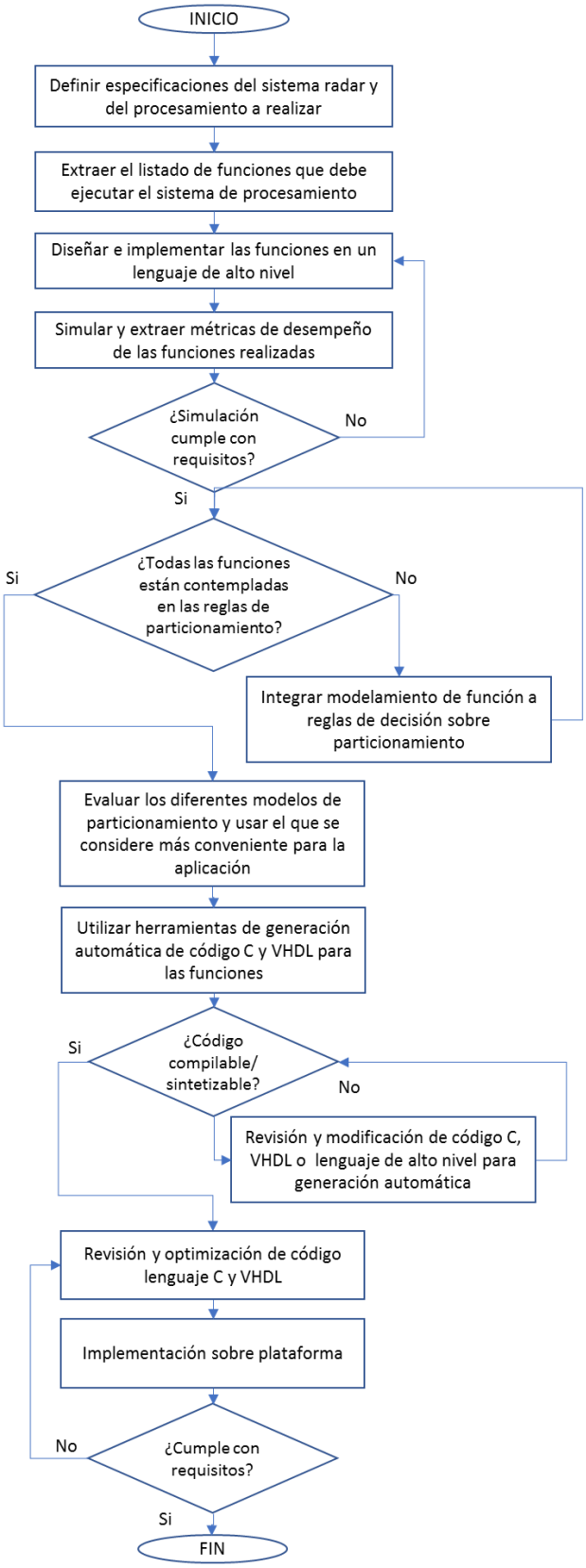


Figura 8. Flujo propuesto para el desarrollo de sistemas usando metodología COMRADES

En la Figura 8, se propone el flujo general propuesto por el proyecto para el desarrollo de sistemas de procesamiento radar usando codiseño hardware/software. En la Tabla 5, se detallan cada uno de los pasos representados en el diagrama.

Paso	Texto	Descripción
1	Definir especificaciones del sistema radar y del procesamiento a realizar	Fase de especificación a nivel de sistema de requerimientos funcionales y no funcionales del sistema radar, definiendo la aplicación específica que tendrá el sistema radar, identificando así las necesidades del sistema de procesamiento. *Continuar con el paso 2*
2	Extraer el listado de funciones que debe ejecutar el sistema de procesamiento	Dependiendo de lo especificado en el paso 1, la primera fase de diseño consiste identificar cuáles son los algoritmos de procesamiento digital necesarios para satisfacer los requisitos del sistema. *Continuar con el paso 3*
3	Diseñar e implementar las funciones en un lenguaje de alto nivel	Una vez identificadas las funciones necesarias para el procesamiento, es necesario en primera instancia revisar si ya están implementadas en lenguaje de alto nivel, en caso contrario se deberán implementar. Para la herramienta COMRADES, el lenguaje utilizado es Matlab. *Continuar con el paso 4*
4	Simular y extraer métricas de desempeño de las funciones realizadas	Con la implementación en lenguaje de alto nivel de todas las funciones necesarias para el funcionamiento del sistema, se utiliza una herramienta de simulación que permita identificar si con el procesamiento implementado se pueden cumplir con los requisitos de la aplicación. Para la herramienta COMRADES, se utiliza el simulador que cuenta con la posibilidad de definir diferentes tipos de sistemas radar y ubicar blancos de prueba. Al final de la simulación se obtienen las métricas de tiempo de ejecución y recursos utilizados. *Continuar con el paso 5*
5	Evaluar si la simulación cumple con requisitos	Realizar la evaluación, si con la implementación en alto nivel, se observa el cumplimiento de los requisitos mediante la simulación. *Continuar con el paso 6* en caso de ser afirmativo, o *continuar al paso 3* en caso de no cumplir con los requerimientos, para realizar ajustes a la implementación de las funciones
6	Evaluar si todas las funciones están contempladas en las reglas de segmentación	El diseñador debe evaluar si todas las funciones a usar están contempladas en las reglas de segmentación. En caso afirmativo *continuar con el paso 7*, de lo contario *continuar al paso 8*
7	Evaluar los diferentes modelos de segmentación y usar el que se considere más conveniente para la aplicación	En este paso se toma la decisión sobre la segmentación del sistema, para ello se plantean tres alternativas: • Método de hibridación que consiste en que el diseñador basándose en las métricas toma la decisión con respecto a la asignación que debe tener cada una de las funciones, bien sea hardware o software • Método automático, este método da prelación a priorizar la velocidad de ejecución de las

Paso	Texto	Descripción
		funciones o a priorizar el uso eficiente de recursos • Método automático que busca un compromiso entre velocidad de ejecución y recursos usados, aplicando las reglas de segmentación de lógica difusa Continuar con el paso 9
8	Integrar modelamiento de función a reglas de decisión sobre segmentación	En el caso que las funciones a implementar no se hayan contemplado anteriormente dentro del modelo de decisión sobre la segmentación hardware/software, se debe incluir la función a las reglas de segmentación, para ello será necesario la validación por parte de un experto. Una vez se haya integrado, *continuar con el paso 6*
9	Utilizar herramientas de generación automática de código C y VHDL para las funciones	La ventaja de la utilización de un lenguaje de alto nivel, es que en una etapa posterior nos permita generar fácilmente el código que será implementado sobre las plataformas de hardware reconfigurable y procesador. Para el caso de la herramienta COMRADES será VHDL o C según el proceso de segmentación. Es importante notar, que sólo se puede generar código VHDL o C, para un conjunto de funciones del lenguaje de alto nivel, además, el hecho que lo genere, no implica que pueda ser compilado o sintetizado, para ser utilizado en un dispositivo real, por lo que en algunos casos será necesario la reescritura de código en lenguaje de alto nivel. *Continuar al paso 10*
10	Evaluar si el código es compilable / sintetizable	Se debe evaluar, si el código generado es compilable o sintetizable, lo que daría lugar a utilización por parte de un sistema real. En caso afirmativo, *continuar al paso 12*, de lo contrario continuar al paso 11
11	Revisión y modificación de código C, VHDL o lenguaje de alto nivel para generación automática	El código que ha sido generado de forma automática tiene insuficiencias, por lo que habrá que revisar si es posible hacer modificaciones a este código generado, o en dado caso, modificar la implementación de la función en lenguaje de alto nivel para que el código generado sea susceptible de ser implementado. Regresar al paso 10
12	Revisión y optimización de código lenguaje C y VHDL	Es habitual que, en el código generado automáticamente por las herramientas de diseño, se encuentren posibilidades de mejoras en tiempo de ejecución y recursos, por lo que el diseñador deberá ajustarlo según su experiencia. Una vez ajustado se *continuar con el paso 13*
13	Implementación sobre plataforma	Se realiza la implementación del código sobre las plataformas (microprocesador, FPGA, etc). *Continuar con el paso 14*
14	Evaluar si cumple con requisitos	Se verifica si la implementación sobre la plataforma es funcional y se valida si cumple con los requisitos establecidos, para este caso es conveniente tener un entorno de pruebas que permita evaluar el funcionamiento del sistema y el cumplimiento de los

Paso	Texto	Descripción
		requerimientos. En caso de cumplir con los requisitos se considerará el final del proceso, en caso contrario, *regresar al paso 12*

Tabla 5. Descripción de pasos de flujo para la metodología COMRADES

5.4. Herramienta COMRADES

Con el propósito de verificar la validez de la metodología propuesta, además de constituirse como una base para la construcción de un sistema de diseño de procesamiento radar, se desarrolla una herramienta de software en Matlab, cuya arquitectura y descripción se presenta a continuación. Esta herramienta ha sido desarrollada en Matlab, debido a que es una herramienta de uso común en ingeniería, por lo que en muchos casos no supone el aprendizaje de un nuevo lenguaje o método de modelado o descripción de sistema.

5.4.1. Arquitectura de la herramienta

Los principales componentes lógicos de la herramienta COMRADES, se presentan en la Figura 9. Adicionalmente, las herramientas empleadas para el desarrollo se presentan en la Figura 10.

El usuario ingresa los parámetros de entrada desde la interfaz de usuario, allí puede iniciar la simulación del diseño, en la cual el motor de simulación del sistema se ejecutará y entregará resultados gráficos, que permitan demostrar el correcto funcionamiento del sistema con los parámetros de entrada, así como extraer y entregar al usuario las métricas de desempeño del sistema. El usuario también podrá desde la interfaz, seleccionar las herramientas de codiseño con las cuales podrá determinar la segmentación para el desarrollo del sistema de procesamiento y tendrá acceso al código generado por el sistema.

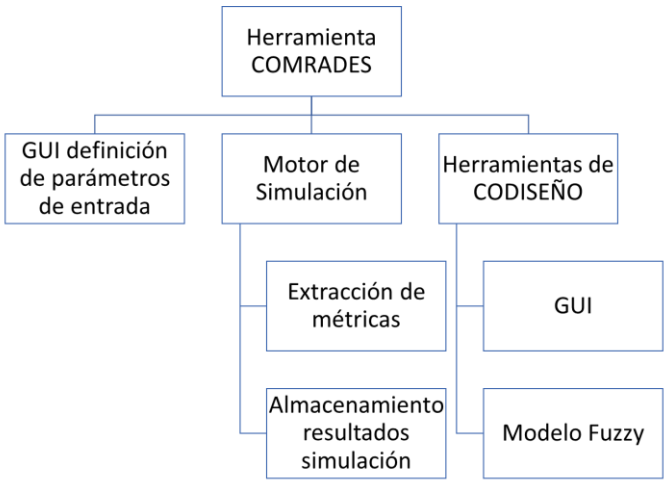


Figura 9. Arquitectura general de la herramienta COMRADES

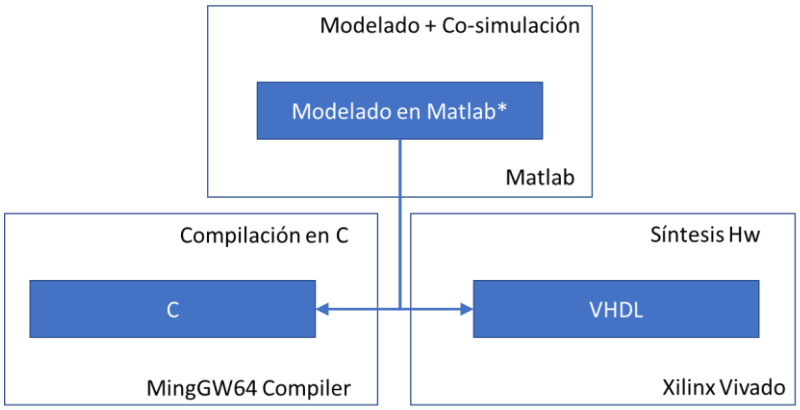


Figura 10. Herramientas utilizadas para modelado, simulación, compilación y síntesis de hardware. Todas pueden ser integradas al entorno de trabajo de Matlab

GUI definición de parámetros de entrada

Esta interfaz gráfica de usuario (*Graphical User Interface*, GUI) es el primer contacto que tiene el diseñador con la herramienta, allí puede configurar los principales parámetros del sistema radar a desarrollar, generar blancos de prueba e iniciar la simulación o la herramienta de codiseño. En la Figura 11 se observa la pantalla principal de la herramienta.

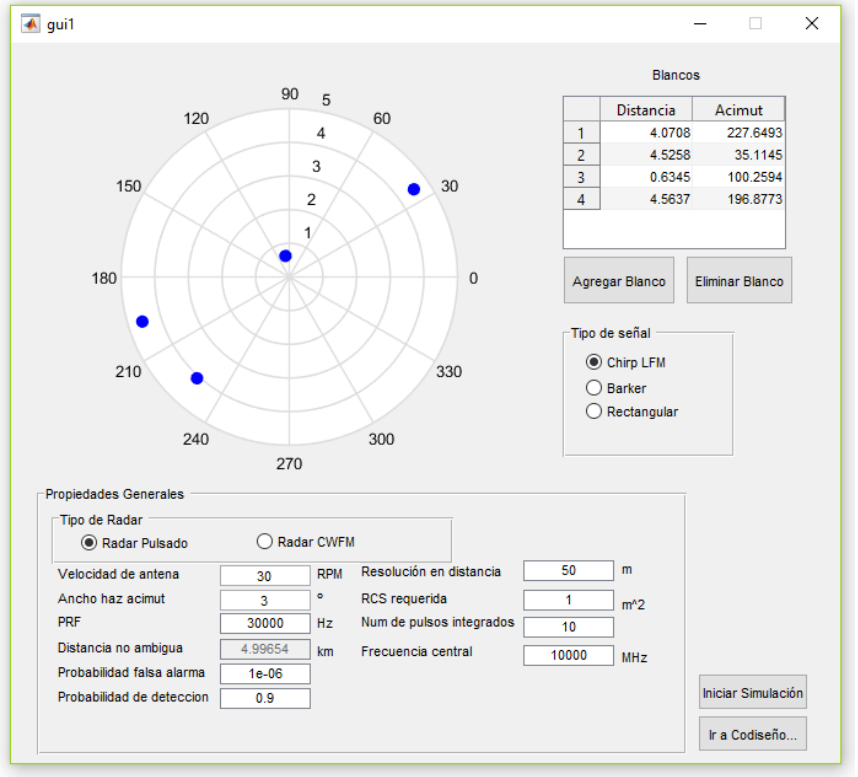


Figura 11. Pantalla principal de la interfaz de usuario para captura de parámetros del sistema radar

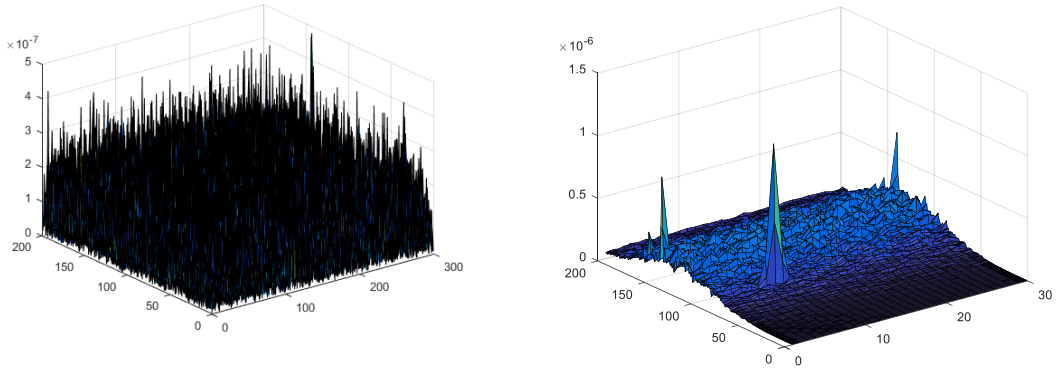
Motor de simulación y extracción de métricas

Este componente se comienza a ejecutar al pulsar la opción de Iniciar Simulación de la pantalla principal (ver Figura 11). Consiste en ejecutar la simulación, basándose en los parámetros seleccionados. Los principales elementos de simulación son:

- Patrón de radiación de antena
- Propagación de la onda (se asume un ambiente de propagación de espacio libre)
- Forma de la onda
- Características del receptor del sistema, contemplando su ganancia y figura de ruido
- Características de transmisión del sistema, contemplando su ganancia y potencia de transmisión
- Característica de los blancos de prueba
- Funciones de procesamiento de la señal

La simulación, se hace mediante un muestreo de las señales. Se simulan todos los pulsos que tiene el sistema radar en una vuelta de antena. A pesar que algunos parámetros no son modificables desde la interfaz gráfica, puesto que sólo se contemplaron los principales parámetros de entrada, todos los parámetros son modificables desde el código en Matlab. Este proceso entrega:

- Gráficas de resultado del proceso de simulación para verificar si el sistema cumple con los requerimientos solicitados, (ver Figura 12).
- Resultados de métricas de ejecución que permitirán tomar decisiones con respecto a la segmentación del sistema, (ver Figura 13).
- Se almacenan los resultados en archivos, puesto que, dependiendo de los parámetros de entrada, la simulación puede ser un proceso demorado y dichos archivos se usan para tomar datos de simulaciones pasadas y ser usados en el bloque de codiseño.



(a) (b)

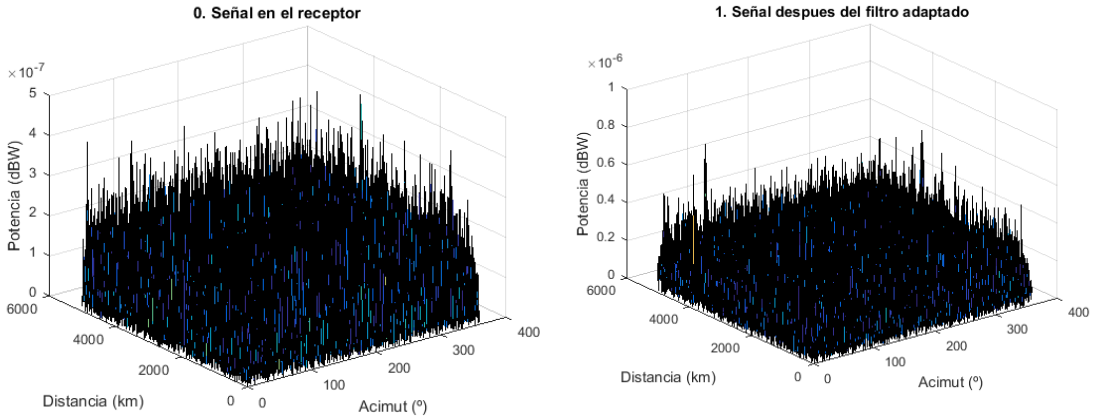
Figura 12. Recepción de ecos para formar la matriz radar. (a) Antes del procesamiento, es muy difícil diferenciar los blancos del ruido. (b) Luego del procesamiento, se puede diferenciar más fácilmente los blancos del ruido

Function Name	Function Type	Calls	Total Time	Allocated Memory	Freed Memory	Peak Memory	% Time	Time Plot
f03_3applyPulseIntegration	function	60000	0.524 s	0.00 Kb	0.00 Kb	0.00 Kb	32.9%	
f01_2applyMatchedFilter	function	300	0.326 s	496.00 Kb	0.00 Kb	496.00 Kb	20.5%	
f02_2applySensitivityTimeControl	function	60000	0.180 s	0.00 Kb	0.00 Kb	0.00 Kb	11.3%	
f04_3PeakDetection	function	6000	0.017 s	0.00 Kb	0.00 Kb	0.00 Kb	1.1%	
Self time (built-ins, overhead, etc.)			0.546 s	1456.00 Kb	0.00 Kb	496.00 Kb	34.2%	
Totals			1.593 s	1952.00 Kb	0.00 Kb	496.00 Kb	100%	

Figura 13. Métricas de ejecución de las funciones f01 – f04, con los tiempos de ejecución y memoria

Un sistema radar de vigilancia puede presentar un amplio conjunto de técnicas de procesado, como las analizadas en la Tabla 1. Como aproximación inicial, se contemplaron dentro del alcance de la herramienta COMRADES un conjunto de técnicas de procesamiento radar, que en Matlab se modelan como funciones, de tal forma que se muestren independientes del resto del código, para permitir la generación automática de código VHDL y C. A continuación, se describen:

- F1- Filtro adaptado: se aprovecha el hecho que la forma del pulso transmitido es conocida, con lo que realizando una operación de correlación o convolución permite mejorar la relación señal a ruido de los valores recibidos. En la Figura 14 se presenta el panorama antes y después de la aplicación del filtro adaptado. El cambio se puede observar analizando el cambio en la escala del eje de potencia de la señal entre la figura antes del filtro y después del filtro.
- F2- *Sensitivity Time Control*: debido a que las señales procedentes de blancos muy cercanos tendrán valores de potencia muy altos, y las señales procedentes de blancos lejanos, presentarán valores de potencia muy bajos, se genera un perfil de compensación para trabajar con niveles de potencia similar. Ver Figura 15.
- F3- *Pulse Integration*: Teniendo en cuenta que debido al ancho del haz de la antena en acimut y la PRF, de un mismo blanco se pueden recibir varios pulsos, se integran constructivamente estas contribuciones para permitir diferenciar los blancos del ruido. La imagen presentada en la Figura 12 b, corresponde a la forma de la señal, luego de haber realizado la integración de pulsos. Ver Figura 16.
- F4- *Peak detection*: Con el propósito de determinar si un pico de la señal se detecta como blanco o no, se debe revisar las señales que exceden el umbral y en ellas determinar el punto máximo de elevación de la señal (detección de pico). En la Figura 17, se presenta una representación gráfica de lo que busca esta tarea.



(a) (b)

Figura 14. Procesamiento de la matriz radar por medio del filtro adaptado, en gráficas de acimut vs distancia, siendo el eje vertical la potencia de la señal recibida. a) Previo a la aplicación del filtro adaptado. b) Posterior a la aplicación del filtro adaptado.

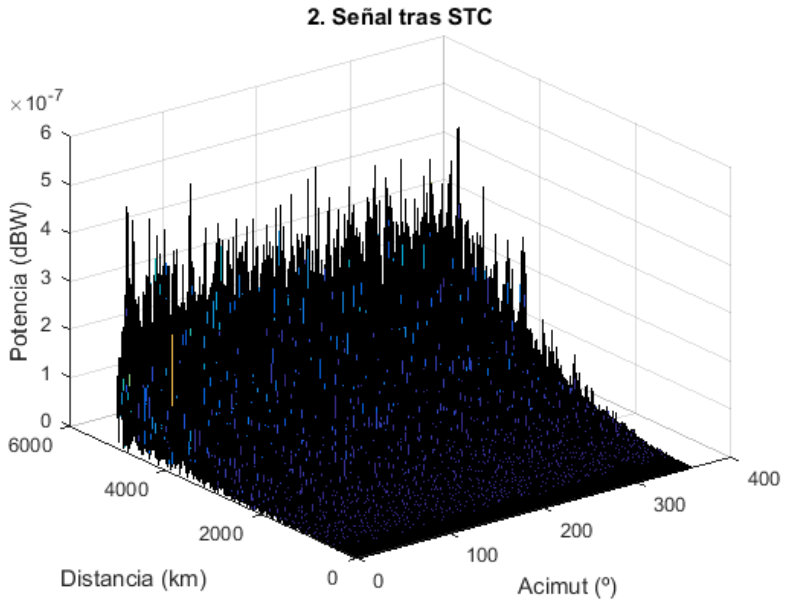


Figura 15. El control de sensibilidad en el tiempo (STC) amplifica las señales que se encuentran más lejos y atenúa las que están más cerca del radar.

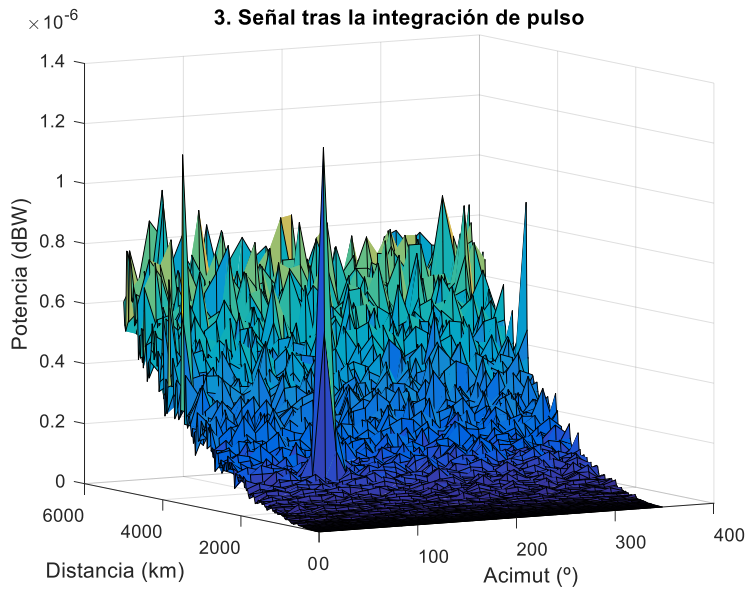


Figura 16. La integración de pulso suma las contribuciones de ecos del mismo blanco capturadas en pulsos diferentes.

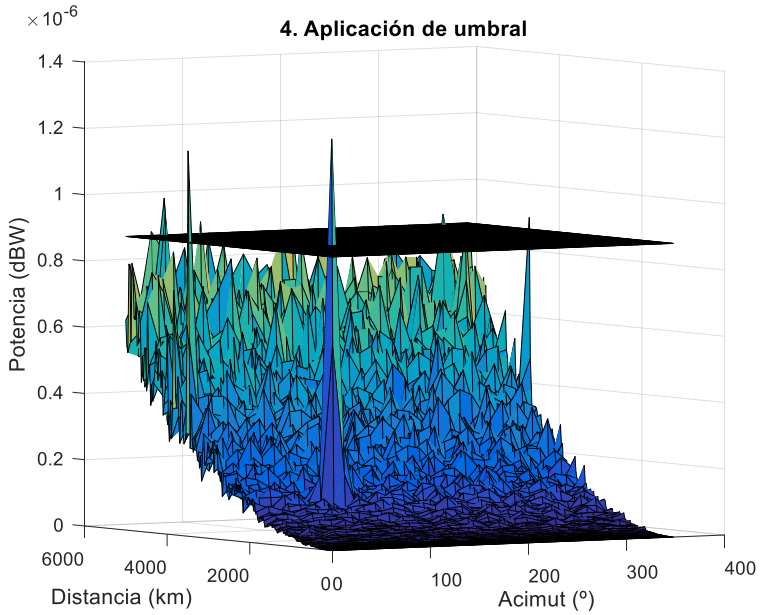


Figura 17. El proceso de detección de pico consiste en analizar cuáles señales superan el plano del umbral, y en ellas hallar el punto máximo de elevación, en la gráfica presentada tan solo uno de los blancos superaría el umbral y allí se estimaría el pico para determinar el acimut del blanco.

Herramientas de codiseño

Este componente se encarga de la segmentación del diseño en dispositivos basados en hardware configurable y dispositivos basados en programación de software. En la Figura 18, se presenta la interfaz de codiseño. Para iniciar será necesario el cargar el archivo resultado de la simulación Para llevar a cabo la segmentación, es posible elegir entre los siguientes métodos:

- Método de hibridación, en este método, el usuario del sistema, se basa en su experiencia y en las métricas de la simulación, y, elige de forma manual la asignación a cada una de las funciones.

- Método *slider* entre área y velocidad, aquí el diseñador selecciona la preferencia del diseño, al ubicar la barra completamente en velocidad, el sistema asignará las funciones a hardware ya que estos dispositivos al poder paralelizar procesos ofrecen velocidad, al ubicar el cursor completamente en área, se asignarán las funciones a software ya que los procesadores al ejecutar tareas de forma secuencial con una única unidad de procesado, requieren menos área en hardware.

- Método métricas, compromiso entre velocidad y recursos. Este método utiliza las reglas que de lógica difusa que han sido definidas, para que según sean los requerimientos del sistema a diseñar, se seleccione el que se infiere como el mejor esquema de segmentación del diseño.

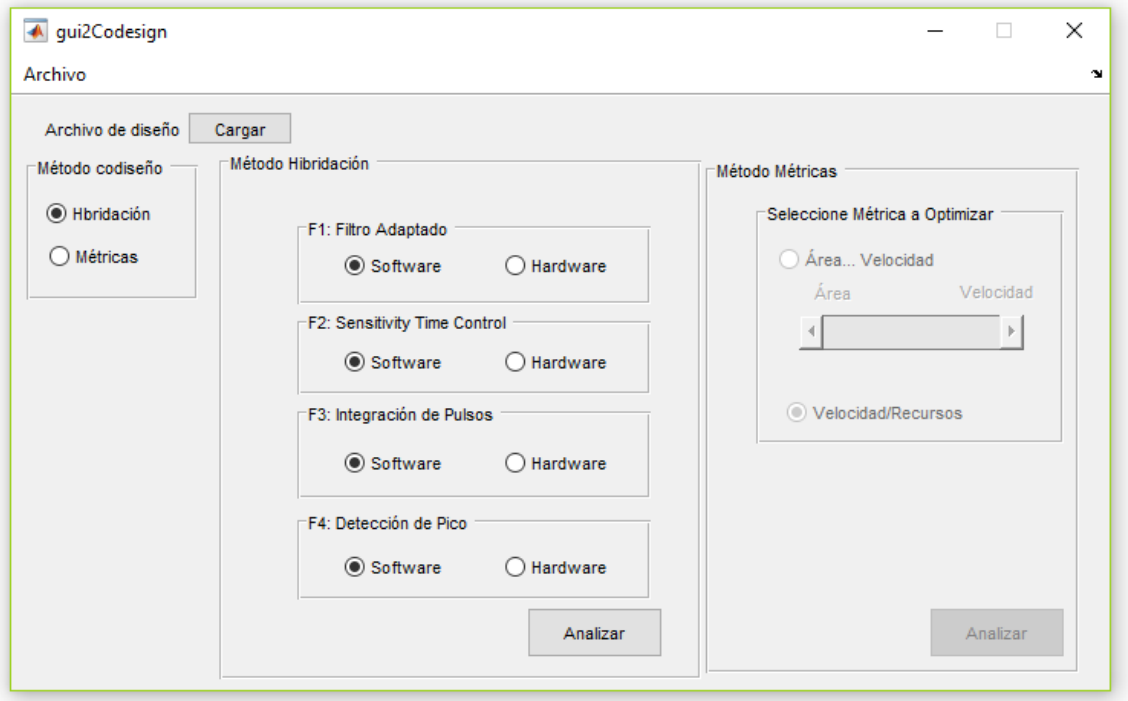


Figura 18. Interfaz de codiseño del sistema

Una vez concluido el análisis, el sistema arroja una ventana de resultados, esta ventana se observa en la Figura 19. De aquí se conocerá la segmentación sugerida y se podrá acceder a los archivos en lenguaje C y VHDL.

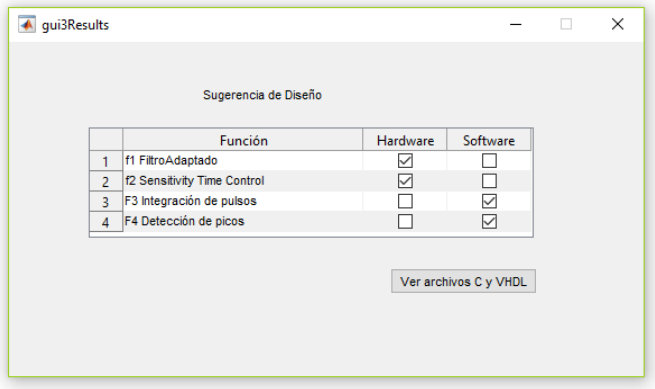


Figura 19. Resultado proceso de segmentación de codiseño

5.4.2. Modelamiento difuso de las reglas de diseño

La toma de decisiones sobre implementación, bien sea en hardware o en software de cada una de las funciones de procesamiento de señal, está basada además de en las métricas, en una heurística asociada a la experiencia pueden tener los diseñadores de estos sistemas, y que se pueden modelar mediante lógica difusa.

Las etapas para el modelado de un sistema difuso son principalmente:

- Definición de variables de entrada
- Definición de conjuntos difusos de entrada
- Definición de variables de salida
- Definición de conjuntos difusos de salida
- Definición de reglas y peso de las reglas

Para la herramienta COMRADES, a continuación, se describen las reglas usadas para la toma de decisión sobre segmentación para el caso de la selección del método de mejor relación velocidad/recursos.

Definición de Variables de Entrada
Las variables de entrada que pueden definir el comportamiento del sistema son:

- PRF
- Resolución en distancia
- Número de pulsos integrados
- Velocidad de antena

Definición de conjuntos difusos
Las funciones de pertenencia de los conjuntos difusos se representan en el eje y con valores que van desde 0 a 1, siendo 1 una pertenencia completa al grupo y 0 ninguna pertenencia.

PRF (Hz)

Las funciones de pertenencia alta y baja, para Frecuencia de Repetición de Pulso (Pulse Repetition Frequency, PRF) se presentan en la Figura 20.

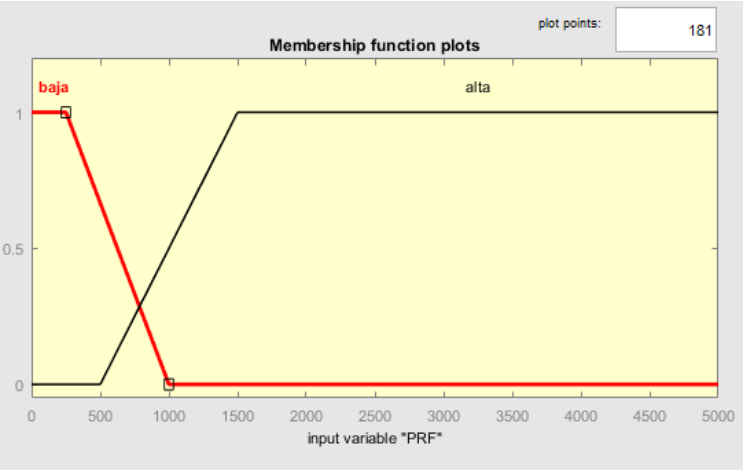


Figura 20. Funciones de pertenencia para frecuencia de repetición de pulso (PRF). Eje x en Hz.

Resolución en distancia (m)

Las funciones de pertenencia alta y baja, para resolución en distancia en metros, se presentan en la Figura 21.

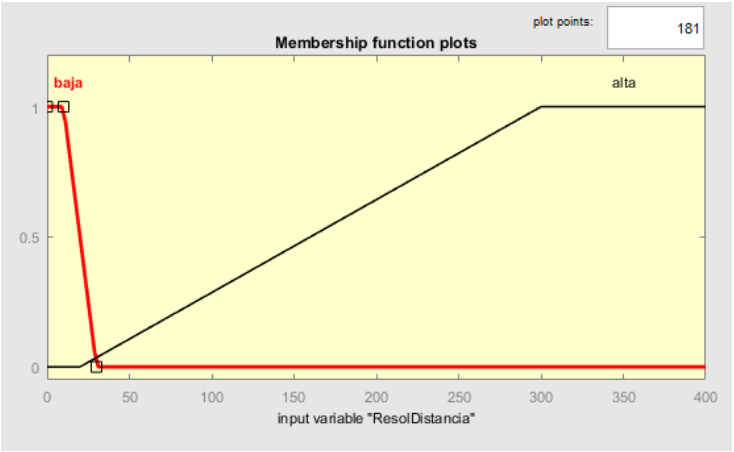


Figura 21. Funciones de pertenencia para resolución en distancia. Eje x en metros.

Pulsos integrados

Las funciones de pertenencia alta y baja, para el número de pulsos integrados, se presenta en la Figura 22.

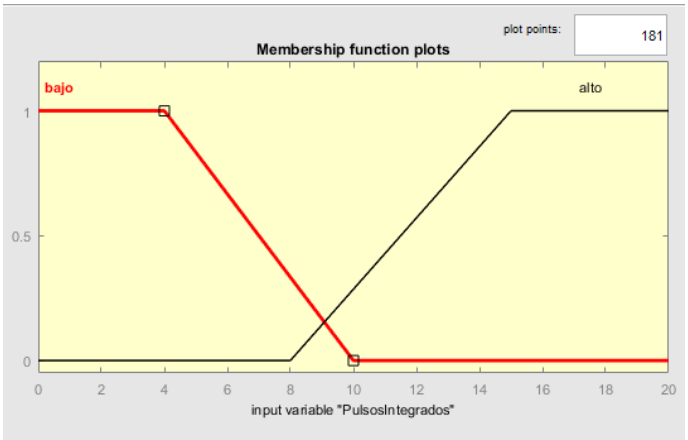


Figura 22. Funciones de pertenencia para número de pulsos integrados.

Velocidad de antena

Las funciones de pertenencia alta y baja para la velocidad de la antena en RPM, se presenta en la Figura 23.

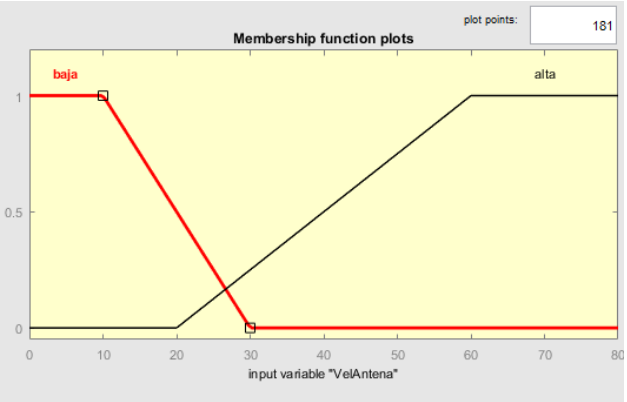


Figura 23. Funciones de pertenencia para velocidad de la antena. Eje X en RPM.

Definición de variables de salida

Las variables de salida nos aportarán un valor entre 0 y 1 indicando si la función debe implementarse en hardware o en software, por lo que los conjuntos serán:

- Implementación función 1, filtro adaptado, hardware o software.
- Implementación función 2, *Sensitivity Time Control* (STC), hardware o software.
- Implementación función 3, integración de pulsos, hardware o software.
- Implementación función 4, detección de pico, hardware o software.

Los conjuntos son idénticos para todos los casos y se representan como el que se muestra en la Figura 24.

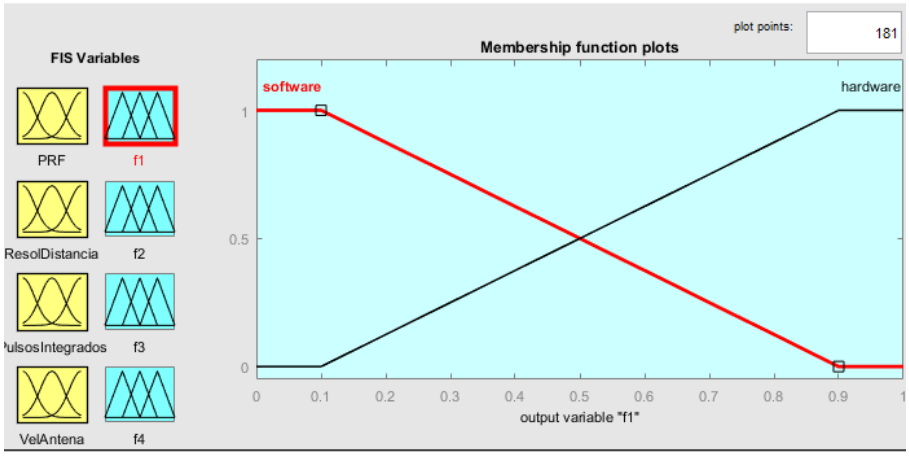


Figura 24. Funciones de pertenencia para función de salida. En el eje x se presenta el grado de afinidad con hardware o software

Definición de reglas

Las reglas forman parte del conocimiento experto que nos indicará la mejor asignación para cada una de las funciones dependiendo de los parámetros de entrada seleccionados en el sistema. En la Tabla 6 se presenta el conjunto de reglas de la herramienta COMRADES.

Número de la regla	Parámetro de entrada	Parámetro de salida
1	Si la PRF es alta	Filtro adaptado se asigna a hardware
2	Si la PRF es baja	Filtro adaptado se asigna a software
3	Si la resolución en distancia es baja	Filtro adaptado se asigna a hardware
4	Si la resolución en distancia es alta	Filtro adaptado se asigna a software
5	Si la PRF es alta	Sensitivity Time Control se asigna a hardware
6	Si la PRF es baja	Sensitivity Time Control se asigna a software
7	Si la resolución en distancia es baja	Sensitivity Time Control se asigna a hardware
8	Si la resolución en distancia es alta	Sensitivity Time Control se asigna a software
9	Si los pulsos integrados son altos	Integración de pulso se asigna a hardware
10	Si los pulsos integrados son bajos	Integración de pulso se asigna a software
11	Si la resolución en distancia es baja	Integración de pulso se asigna a hardware
12	Si la resolución en distancia es alta	Integración de pulso se asigna a software
13	Si los pulsos integrados son bajos	Detección de pico se asigna a software
14	Si los pulsos integrados son altos	Detección de pico se asigna a hardware
15	Si la PRF es alta	Detección de pico se asigna a hardware
16	Si la PRF es baja	Detección de pico se asigna a software
17	Si la velocidad de antena es baja	Detección de pico se asigna a hardware
18	Si la velocidad de antena es alta	Detección de pico se asigna a software

Tabla 6. Conjunto de reglas de modelamiento difuso para asignación de funciones con modalidad hardware o software

5.5. Implementación de la herramienta

La implementación de la herramienta del proyecto COMRADES (*Codesign Methodology for Radar in Embedded Systems*) fue desarrollada en Matlab, tomando algunos elementos como base para su desarrollo. En este capítulo se exponen los diferentes componentes que implicó el desarrollo de la herramienta, mencionando los elementos integrados, así como los que corresponden a desarrollo propio para el proyecto.

5.5.1. Patrón de radiación de la antena

La simulación del sistema radar, entre otros parámetros, contempla el patrón de radiación de la antena de transmisión y recepción. El *Phased Array System Toolbox* de Matlab permite la creación de patrones de radiación personalizables, bien sea definiendo los elementos radiantes o la ecuación de su patrón de radiación. Debido a que el ancho de haz en acimut es un parámetro importante en un sistema radar ya que define la resolución acimutal del sistema, se deja la opción en la interfaz gráfica de la aplicación, que el ancho a 3 dB de la antena sea modificable por el usuario.

El tipo de patrón de radiación usado es el haz tipo pincel (*pencil beam*) al que se le varía el ancho según los parámetros ingresados por el usuario.

Para la variación de este ancho de haz, se usa la ecuación propuesta en el libro Modern Antenna Desgin [53].

$$U(\theta) = \cos^{2N}(\theta/2)$$

Siendo U(θ) la función que describe la intensidad de radiación en función del ángulo. N, por su parte está estrechamente relacionada a la directividad del patrón de radiación.

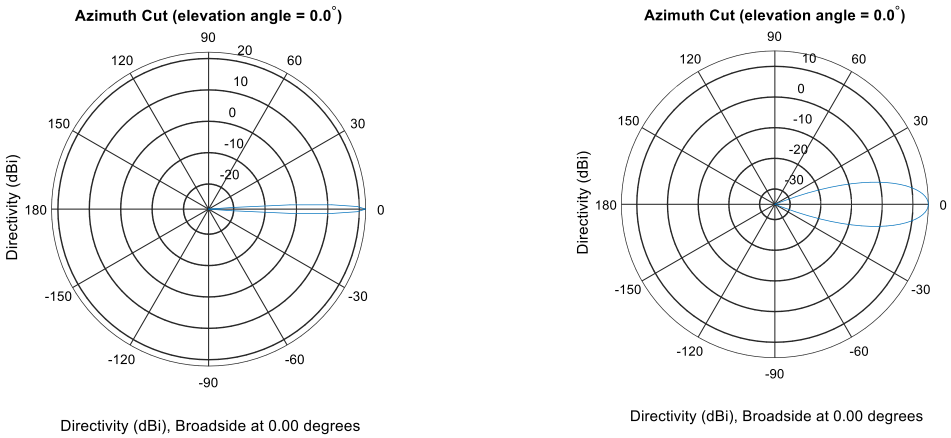
$$N = \frac{-\mathrm{Lvl(dB)}}{20\log[\cos(\mathrm{beamwidth}_{\mathrm{Lvl(dB)}}/4)]}$$

En el numerador, se pone el punto a partir del cual se determina el ancho de haz en dB que para caso de sistemas radar, por lo general se toma a -3 dB.

Con estas ecuaciones por lo tanto se puede parametrizar el ancho de haz de la antena en la simulación en Matlab como se observa en la Figura 25. En la Fig se observan dos diferentes patrones de radiación para ancho de haz 3º y 15º respectivamente.

```
% %Tomado del libro Modern Antenna Design.
%Thomas A Milligan Second Edition
% %1-8.1 Pencil Beam en Acimut

bw3dBrad=deg2rad(anchoHaz);
N=(-3)/(20*log10(cos(bw3dBrad/4)));

hant = phased.CustomAntennaElement;
hant.AzimuthAngles = -180:180;
hant.ElevationAngles = -90:90;
 az = [-180:180];
 el = [-90:90];
 [az_grid,el_grid] = meshgrid(az,el);

hant.RadiationPattern =...
        mag2db(abs((cosd(az_grid/2)).^(2*N)));

```

Figura 25. Generación de un patrón de radiación *pencil beam* variable en acimut



(a) (b)

Figura 26. Patrón de radiación, imagen de corte en acimut para dos diferentes anchos de antena (a) 3º (b) 15º

5.5.1. Interfaz gráfica

La interfaz gráfica a nivel de ventanas, paneles y controles que tiene la aplicación, fue completamente desarrollada para el proyecto. Se usó la herramienta GUIDE de Matlab que corresponde a un conjunto de elementos visuales que permiten una rápida creación de los componentes, como se puede observar en la Figura 27. Ya mediante código, se implementa la reactividad de los botones y controles utilizando sus funciones *callback*. También es importante considerar qu las variables que se crean en los archivos de interfaz gráfica no son visibles en archivos diferentes, razón por la cual se debe utilizar la función "`assignin`" para extender la visibilidad de estas variables a archivos Matlab diferentes.

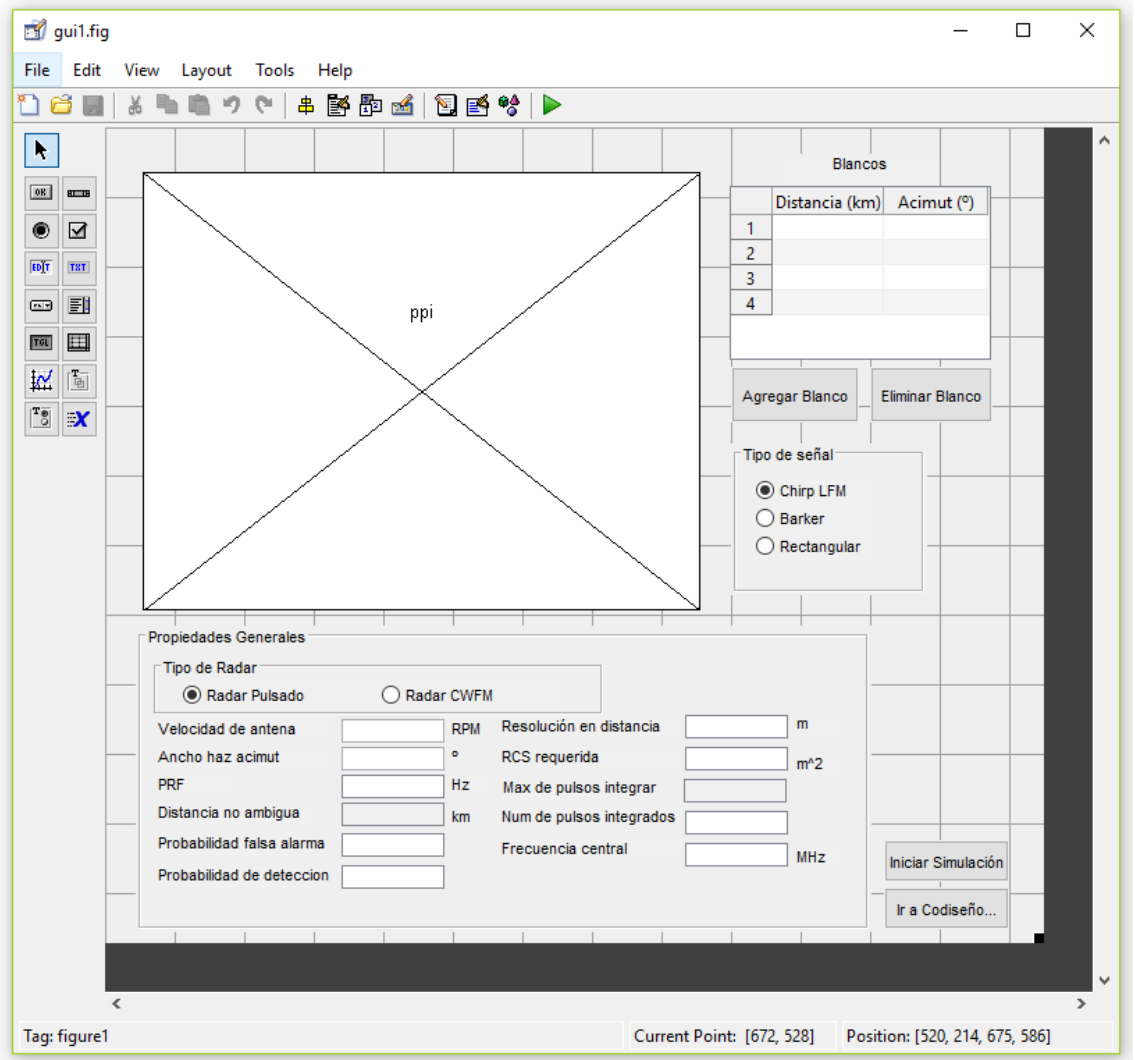


Figura 27. Entorno para creación de interfaz gráfica

5.5.2. Entorno de simulación

El entorno de simulación corresponde al conjunto de elementos en Matlab que permiten, por una parte, modelar el sistema y por otra analizar su comportamiento frente a diferentes entradas. El *Phased Array System Toolbox*, facilita la definición de varios componentes para sistemas radar:

- Creación de blancos especificando su sección radar (RCS) y posición, entre otros.
- Definición del canal de propagación de las ondas
- Parámetros de las antenas
- Características del receptor
- Definición de formas de onda del sistema
- Síntesis de la señal
- Procesamiento radar de la señal

Un punto de inicio para el proyecto se basó en el código de diseño propuesto de un radar básico monoestático de Matlab [54]. En este se especifican los parámetros de un radar, con la limitación

que está planteado para un radar unidimensional que sólo detecta rango, es decir, no tiene una antena giratoria y todos los blancos están ubicados en posiciones fijas a lo largo de una misma línea, tampoco cuenta con interfaz gráfica. Por lo tanto, el trabajo inicial desarrollado consistió en lograr extender esta simulación a un radar bidimensional con una antena girando a cierta velocidad.

En el código simplificado expuesto en la Figura 28, se representa el ciclo principal de simulación. La simulación, como se puede ver no está iterada por tiempo, sino de forma discreta en los ecos para cada uno de los pulsos o rampas que genere el radar en una vuelta. Dentro del ciclo principal, se contempla el efecto de la radiación de los pulsos o rampas en cada uno de los blancos, ya que para minimizar el tiempo de simulación, no vale la pena iterar cada punto del espacio sino sólo en aquellos donde se encuentra algún objeto de interés para nuestra simulación.

```
1
2   for npulso=0:N_pulsos_por_vuelta
3       for blanco=0:N_blancos
4           %analizar efecto de radiación de pulso
5           %en el blanco
6           analizar
7       end
8       %actualizar posición de antena
9       actualizar
10  end
```

Figura 28. Código simplificado de ciclo de simulación de blancos y rotación de antena

5.5.1. Tareas de procesamiento

El *Phased Array System Toolbox*, ofrece funciones de procesamiento para sistemas radar. Sin embargo, en el desarrollo del proyecto se pudo evidenciar que dichas funciones no son compatibles con los módulos de generación de código vhdl. Por esta razón estas tareas debieron reescribirse usando funciones básicas de Matlab.

Adicionalmente, como requisito para que un código pueda pasarse al generador de vhdl o C, se requiere que esté escrito en una función de Matlab en archivos independientes, en la Figura 29 se presenta la interacción entre los archivos de simulación y procesamiento.

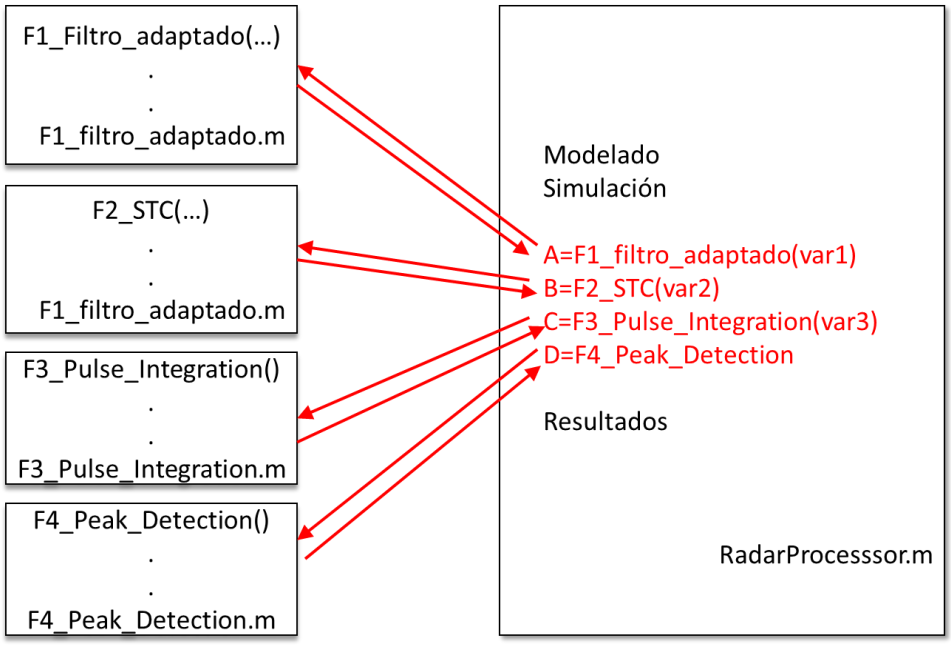


Figura 29. Llamado a las funciones de procesamiento desde el archivo principal

5.5.1. Extracción de métricas

Para la extracción de métricas de tiempo de procesamiento y memoria utilizada, se usó la herramienta *profiler* de Matlab. Sin embargo, es importante anotar, que no se desea extraer las métricas de todo el código, sino solamente de las funciones de procesado, ya que estas serían las que implementaría un procesador real, por lo tanto, se debe pausar la ejecución del *profiler* cuando no se estén ejecutando. En la Figura 30, se presenta un ejemplo de uso del profiler, en donde se observa que también se usa la opción `-memory on`, de tal forma que también se extraigan mediciones de memoria. Los resultados de estas mediciones se obtienen luego de la ejecución de la simulación, y se observan en la Figura 13.

```
432 -    profile -memory on
433 -    for i=1:1:num_acimuts
434 -    [output_signal(:,i)]= f01_2applyMatchedFilter(matchingcoeff, ...
435                              rx_pulses(:,i), weight);
436 -    end
437 -    rx_pulses=output_signal;
438 -    profile off
```

Figura 30. Uso de *profiler* para extracción de métricas

5.5.2. Lógica difusa

Una vez modelado los conjuntos y reglas difusas, para la codificación en Matlab, se utiliza la herramienta *Fuzzy Logic Designer*, ver Figura 31, el cual consta de una interfaz gráfica desde la cual se puede crear el modelo. Una vez creado el modelo, se guarda como un archivo, y al momento de usarlo desde el código, se usan los comandos `readfis` y `evalfis` para leer los datos del modelo guardado en el archivo y poderlos utilizar para evaluar la salida ante determinadas entradas.

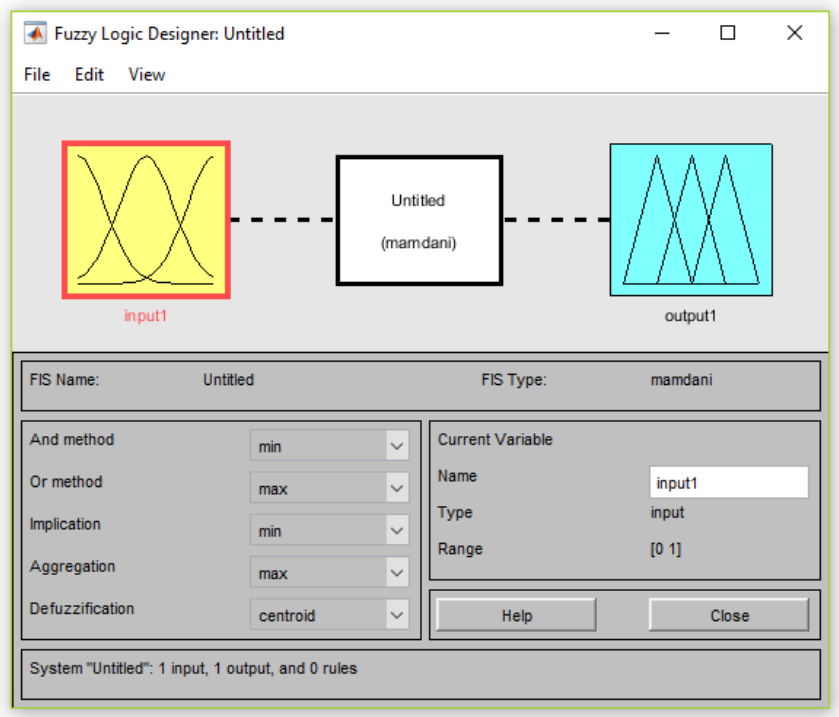


Figura 31. Uso de *la herramienta* Fuzzy Logic Designer

5.5.1. Codificación C y VHDL

La generación de código en C y VHDL a partir del código Matlab de alto nivel, se realiza utilizando las herramientas de Matlab C Coder y HDL Coder. Estas herramientas permiten generación automática de código, a partir de las funciones de procesado.

Los pasos para generar código en VHD usando HDL *Coder* se observan en la Figura 32. Estos son:

- Definición de tipo de datos de entrada. Este proceso tiene la ventaja que puede hacerse de forma automática tomando como base los valores del archivo de principal de simulación “RadarProcessor.m.
- Conversión a punto fijo. Se convierten las variables al formato de punto fijo usado en el HDL y se realiza una comprobación de funcionamiento.
- Selección de dispositivo. Aquí se hace uso de la conexión entre Matlab y el software de síntesis de hardware, que en este caso se realizó con el Xilinx Vivado.

- Generación de código. Se genera el código VHDL.
- Verificación. Opcionalmente se puede verificar el funcionamiento usando simuladores especializados como Modelsim. También tiene como ventaja que se genera el archivo VHDL *testbench*, con el que se puede realizar la simulación en la herramienta de desarrollo de FPGA.

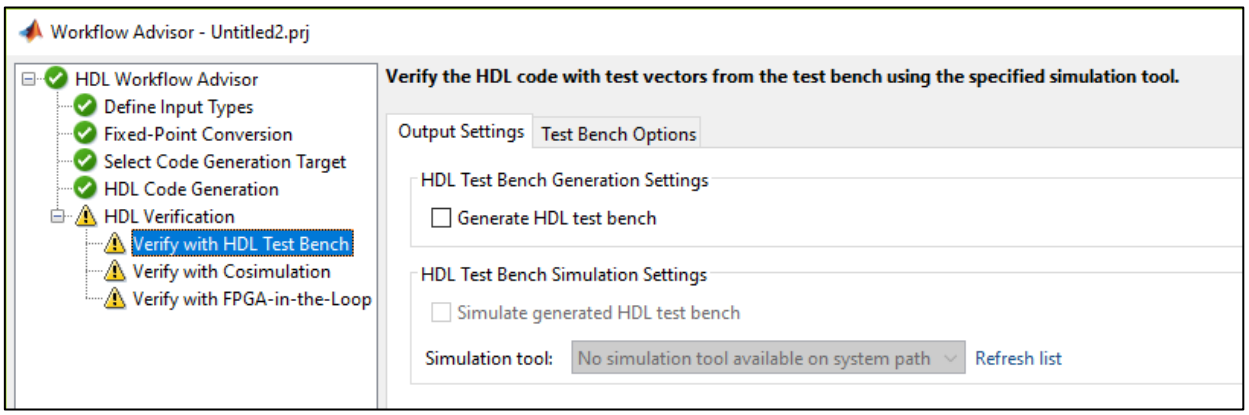


Figura 32. Pasos para la creación de código VHDL usando la herramienta HDL *coder*

El proceso para generar código C de la herramienta consiste en:

- Definición de tipo de datos de entrada. Se elige el tamaño y tipo de variables en C en los que se van a almacenar los datos de los parámetros de entrada. Usando el archivo principal "RadarProcessor.m" las entradas se definen automáticamente. En la Figura 33 se observa la interfaz de definición de tipo de variables de entrada para la generación de código C a partir de una función.
- Selección de plataforma. El código se puede generar en C para el mismo computador que está ejecutando Matlab, o para dispositivos de diferentes fabricantes como ARM, BeagleBone Black, entre otros.
- Generación de código.

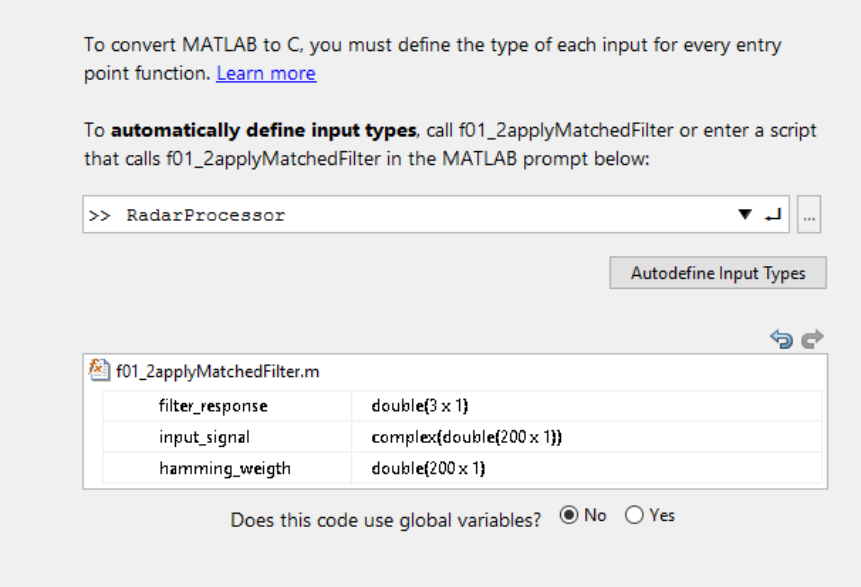


Figura 33. Creación de archivos C con la herramienta C Coder

5.6. Evaluación de la metodología COMRADES

Para la validación de la metodología de desarrollo, en el contexto del proyecto COMRADES: *Codesign Methodology for Radar in Embedded Systems*, se contemplaron aspectos a evaluar por parte de personas que han tenido experiencias en el diseño de sistemas de procesamiento de señal radar. A continuación, se enumeran y describen dichos aspectos que abarca la validación de la metodología.

1. Importancia del problema del codiseño en el desarrollo de sistemas de procesamiento radar. Para validar correctamente la metodología se considera necesario explicar inicialmente el contexto del trabajo desarrollado, esto es, el problema a tratar en el desarrollo del proyecto e identificar la prioridad que le da el evaluador a dicha problemática, en relación a la actividad de desarrollar un sistema de procesamiento de señal radar.
2. Enfoque hacia diseño asistido por herramientas que faciliten codiseño. Valoración de en qué medida las herramientas de diseño o codiseño permiten o dificultan el proceso y cómo se compara con la herramienta propuesta que demuestra la metodología de codiseño del proyecto.
3. Vista de la estructura de la herramienta propuesta. Consiste en analizar la forma cómo está estructurada la herramienta de ayuda para la metodología de codesarrollo con el propósito de identificar sus fortalezas y debilidades para posibilidades de mejora o rediseño. Esto incluye una revisión y realimentación sobre las funciones de procesamiento inicialmente propuestas.
4. Revisión de las reglas de lógica difusa para herramienta de codiseño. La lógica difusa se incorpora como aproximación heurística en el contexto del presente proyecto para sistematizar un conocimiento experto sobre una temática, sin embargo, precisamente ese conocimiento experto modelado es el que debe revisarse por parte de personas que trabajen en la temática, para validar que los conjuntos planteados y las reglas son acordes a la realidad del diseño.

5. Recopilación de sugerencias sobre expansiones o modificaciones para trabajos futuros. La idea es reunir los comentarios y sugerencias que puedan surgir del uso de la metodología para documentarlo.

5.6.1. Formulario de evaluación de metodología

Se realiza un cuestionario para aplicar a los diseñadores que evalúan el sistema con las preguntas que se observan en la Tabla 7.

Número	Pregunta
1	Documento de identificación
2	¿Con cuáles de las siguientes herramientas ha trabajado en ingeniería? • Artisan Studio • Space studio codesign • Matlab • Vivado HLS • Open Virtual Platform • SolidThinking Embed • Mentor Graphics Vista • ARM SoC Designer • SystemVue ESL design software
3	¿Con cuáles de los siguientes lenguajes de programación o modelado ha trabajado? • SysML • C/C++ • Matlab • SystemC • UML • TLM • VHDL • Verilog
4	¿Ha trabajado en desarrollo de sistemas o componentes de procesamiento de sistemas radar?
5	En el desarrollo de un sistema de procesamiento radar ¿Qué importancia tiene la implementación de algoritmos sobre plataformas hardware (FPGA)?
6	¿En qué medida considera que la partición hardware/software del sistema de procesamiento radar es una decisión que puede afectar el proceso de diseño y desarrollo de un sistema?
7	¿Ha usado herramientas de codiseño hardware/software? ¿En qué medida considera que facilitan la tarea de desarrollo de un sistema de procesamiento radar?
8	En caso de haber usado herramientas de codiseño ¿Cómo compararía la herramienta COMRADES con relación a estas?
9	¿Considera que la estructura de la herramienta propuesta facilita la tarea de desarrollo de sistemas de procesamiento radar o por lo contario implica trabajo adicional que puede repercutir en más tiempo de desarrollo?
10	¿Considera que las funciones de procesamiento de señal radar contempladas dentro de la herramienta son una muestra significativa para verificar el funcionamiento de la metodología? ¿Cuáles otra consideraría importantes?
11	Valore los conjuntos y cada una de las reglas de codiseño indicando si la considera válida o si tiene algún comentario sobre las mismas

Número	Pregunta
12	Qué aspectos considera que se pueden mejorar en la metodología y la herramienta propuesta ¿Qué nuevas funcionalidades se podrían contemplar?

Tabla 7. Cuestionario para validación de proyecto

5.6.2. Resultados de la evaluación de metodología

La evaluación es realizada por personas a las que se les presenta el proyecto y la herramienta de desarrollo. El perfil de las personas encuestadas corresponde a ingenieros con maestría en "Sistemas Radar, Tecnologías y Diseño de Sistemas", además de acreditar 3600 horas de trabajo práctico en el desarrollo de sistemas Radar con las empresas ART e INDRA.

Pregunta 2: A la pregunta de cuáles herramientas ha trabajado en la ingeniería, se observa que Matlab es la herramienta más usada con relación a otros softwares de desarrollo que involucran funcionalidades de codiseño.

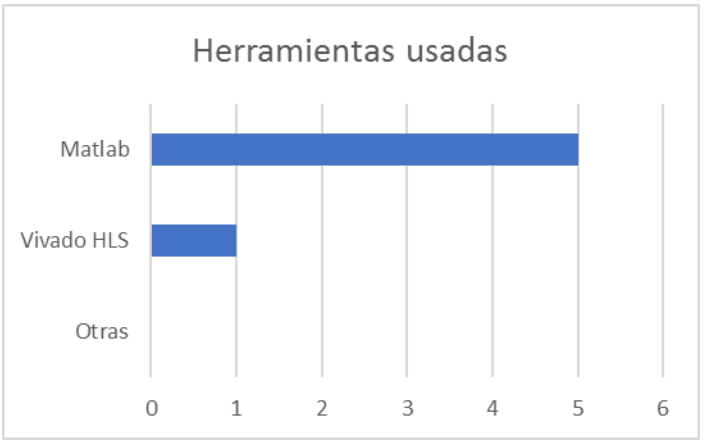


Figura 34. Herramientas usadas por los encuestados, el listado presentado corresponde con las herramientas que tienen funcionalidades de codiseño hardware-software.

Pregunta 3: En cuanto a lenguajes de programación se encuentra que los lenguajes más usados para el desarrollo de sistemas son Matlab, VHDL, C/C++ y en menor medida UML.

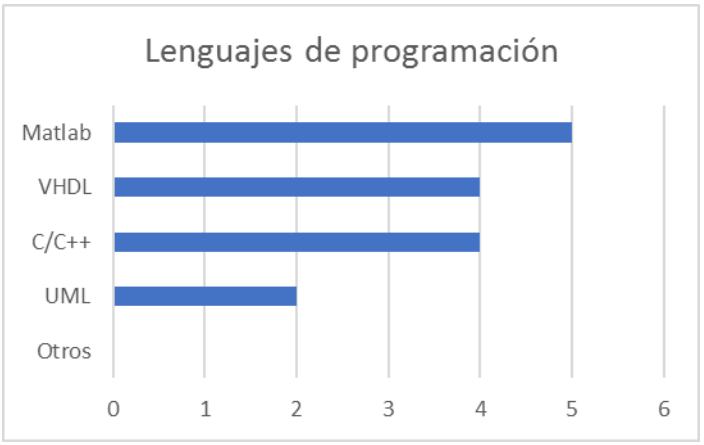


Figura 35. Lenguajes de programación usados por los encuestados.

Pregunta 4: La totalidad de los encuestados ha trabajado en el desarrollo de sistemas radar y procesamiento de señal radar.

Pregunta 5: Sobre la importancia de las FPGAs en el diseño de sistemas de procesamiento radar los comentarios se pueden resumir en los siguientes puntos:

- En su totalidad expresan la importancia de las FPGAs en el desarrollo de sistemas de procesamiento radar.
- Las principales características que se resaltan de las FPGAs son: velocidad, concurrencia, y uso en aplicaciones de tiempo real.

Pregunta 6: Con respecto a si la decisión sobre partición hardware-software tiene un impacto en el proceso de desarrollo de un sistema, las respuestas se resumen en los siguientes puntos:

- Afectan, principalmente en el tiempo de desarrollo y en los recursos del proyecto.
- La medida en la que la decisión sobre la partición del sistema afecta, depende también del tipo de proyecto (militar/civil, desarrollo único/fabricación en serie, corto plazo/largo plazo).

Pregunta 7: Sobre si han utilizado herramientas específicas de codiseño hardware-software, se encuentra que a pesar de que los encuestados han usado software que facilita el codiseño hardware-software como Matlab y Vivado HLS, en ninguno de los casos lo han usado en el contexto del codiseño hardware-software.

Pregunta 8: No es posible establecer una comparativa de la herramienta trabajada en este proyecto, con respecto a otras herramientas debido a que no han sido utilizadas por los participantes.

Pregunta 9: Con respecto a si la estructura de la herramienta, facilita el desarrollo de sistemas de procesamiento o por el contrario sería una tarea adicional, se obtiene que, en su totalidad, reconocen que el uso de la herramienta puede facilitar el desarrollo de un sistema de procesamiento radar, aunque en varios casos también identifican condicionantes para que esto se pueda dar:

- Alimentar las reglas de decisión sobre segmentación.
- Puede representar un trabajo adicional en un principio, sin embargo, el beneficio se vería en el contexto de un trabajo continuado en sistemas de procesamiento radar, con lo que a largo plazo el beneficio sería mayor.

Pregunta 10: Con respecto a si las funciones base para la demostración de la herramienta son una muestra significativa para validar el funcionamiento de la metodología, se encuentra que:

- Cuatro de los cinco encuestados consideran que son una muestra significativa, aunque también aportan otras funciones que podrían ser importantes.
- En uno de los casos se considera importante la inclusión de algoritmo de procesamiento CFAR (*Constant False Alarm Rate*) para que se pueda tomar como muestra significativa.
- Las otras funciones que son consideradas importantes por los encuestados son: algoritmos para tratamiento del clutter, CFAR, MTD, *scan conversion*, procesamiento Doppler, asociación de *blobs* y *plots*, generación de *tracks*.

Pregunta 11: Con respecto a la valoración de los conjuntos difusos y las reglas para la partición del diseño en hardware-software, se generan los siguientes comentarios:

- En tres de los cinco casos se considera que los conjuntos y las reglas están bien definidas, aunque también realizan comentarios para mejoras, estos se detallan en otro punto.
- En uno de los casos se considera que es necesario definir la aplicación específica del radar para la aplicación de las reglas, es decir, este modelamiento difuso es aplicable bajo las condiciones dadas de un contexto de aplicación, que podría incluirse en la interfaz de usuario.
- En otro de los casos se proponen dos reglas nuevas: si la velocidad de antena es baja, la integración de pulso se asigna a hardware, en caso contrario a software.
- En uno de los casos se plantea como mejora el identificar la relación de interdependencia que puede existir entre la asignación de funciones, por ejemplo: asignar la función B a hardware, puesto que la función A ha sido asignada a software.

Pregunta 12: Con respecto a las mejoras propuestas para la herramienta y la metodología se generan los siguientes comentarios:

- Aumentar la cantidad de algoritmos propuestos.
- Considerar el tiempo de comunicación entre procesadores y FPGA.
- Considerar modelos de *clutter*.
- Permitir personalizar la herramienta dependiendo del tipo de radar.
- Incluir elementos de costo en dinero, asociado a la asignación de las tareas en uno u otro dispositivo.

5.7. Caso de diseño de sistema de procesamiento radar mediante la metodología

Con el propósito de identificar las ventajas y desventajas de la metodología y las herramientas de diseño, se propone el desarrollo de un sistema radar con los siguientes requerimientos:

- Sistema radar pulsado de vigilancia
- Velocidad de rotación de la antena de 30 RPM
- Ancho de haz de la antena de 3º
- Frecuencia de repetición de pulso de 10 kHz
- Resolución en distancia de 100 m
- Blancos objetivo con sección radar (RCS) de 1 m^2
- Integración de 20 pulsos
- Frecuencia central de 10 GHz
- Alcance de 4.5 km

Estos requerimientos se capturan en la ventana de interfaz, y se ubican tres blancos de prueba en la gráfica de posición PPI, con diferente acimut (ángulo) y distancia del radar. El origen del radar corresponde al centro de la gráfica. En la Figura 36, se presenta la ventana de interfaz con el usuario.

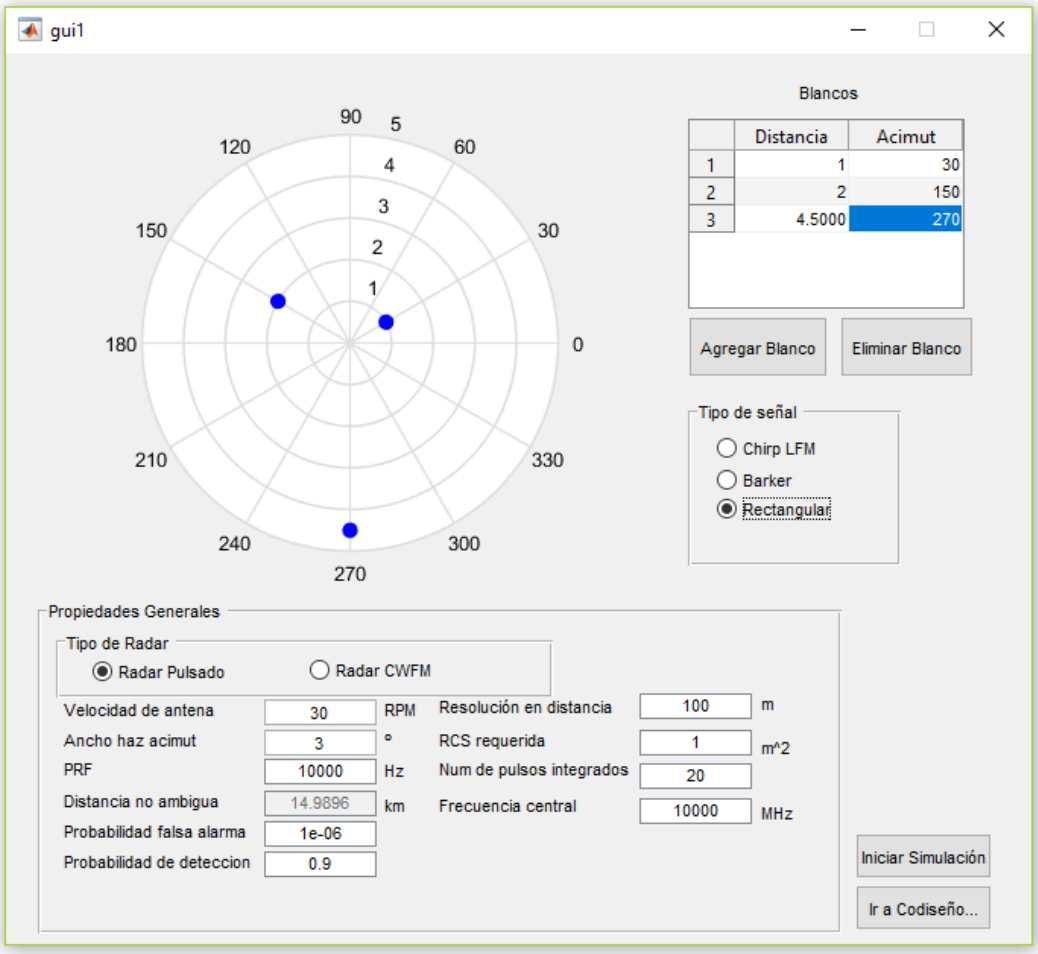
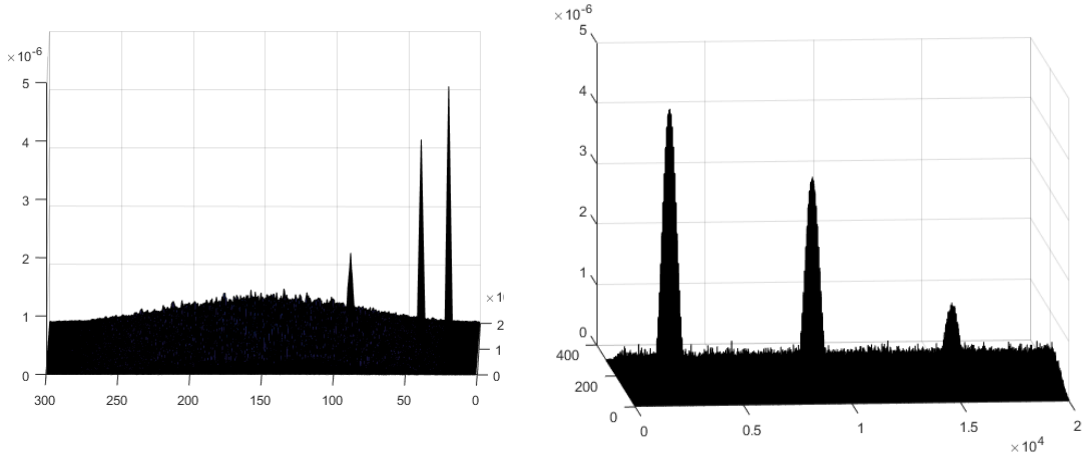


Figura 36. Captura de requerimientos para el desarrollo de un sistema de procesamiento radar

Al pulsar sobre simulación, se generan gráficas y resultados que me permiten evaluar el funcionamiento del sistema. En la Figura 37 se presenta el gráfico que me permite determinar si es posible la identificación de los blancos con los parámetros dados. Se observan tres picos que corresponden a la identificación de cada uno de los blancos para la vista lateral y frontal. Los ejes corresponden a la matriz de procesamiento y el valor de los picos está dado por el nivel en voltaje de la señal recibida. Con esta gráfica es posible identificar que los ecos de las señales pueden ser correctamente recibidos en esta configuración. Adicionalmente se recibe un resultado con las métricas de ejecución de este proceso indicando el número de llamados a cada función, la memoria que requieren y el tiempo total de ejecución. Este resultado se presenta en la Figura 38. Se observa que la función que más consume tiempo de ejecución es la de integración de pulsos y en segunda medida la de STC.



(a) (b)

Figura 37. Gráfica resultado de la simulación del sistema radar, para dos vistas diferentes. Se muestran tres picos, que corresponden a los tres blancos identificados por el radar. En a) se presenta una vista lateral, en donde se aprecia la diferencia en distancia respecto al radar de los diferentes blancos, en b) se presenta una vista frontal en donde se presenta la diferencia en acimut de los blancos.

Children (called functions)

Function Name	Function Type	Calls	Total Time	Allocated Memory	Freed Memory	Peak Memory	% Time	Time Plot
f03_3applyPulseIntegration	function	6000000	67.396 s	0.00 Kb	0.00 Kb	0.00 Kb	49.2%	
f02_2applySensitivityTimeControl	function	6000000	23.454 s	68.00 Kb	36.00 Kb	16.00 Kb	17.1%	
f04_3PeakDetection	function	300000	0.745 s	0.00 Kb	0.00 Kb	0.00 Kb	0.5%	
f01_2applyMatchedFilter	function	20000	0.737 s	0.00 Kb	0.00 Kb	0.00 Kb	0.5%	
Self time (built-ins, overhead, etc.)			44.778 s	141812.00 Kb	225328.00 Kb	46968.00 Kb	32.7%	
Totals			137.110 s	141880.00 Kb	225364.00 Kb	46968.00 Kb	100%	

Figura 38. Se presentan los resultados de las métricas de ejecución de las funciones radar: filtro adaptado, STC, integración de pulso y detección de pico.

El siguiente paso corresponde a pasar a la fase de codiseño. En la fase de codiseño se presenta la ventana que se observa en la Figura 39. Allí se carga el archivo generado anteriormente en la fase de simulación y elegimos el método que nos permite la mejor relación entre Velocidad y Recursos. Para este caso específico el sistema indica que es preferible la implementación de todas

las tareas en hardware, como se observa en la Figura 40. Si el método de codiseño hardware-software es el del slider entre área y velocidad, se obtienen los resultados sugeridos por el sistema que se presentan en la Figura 41.

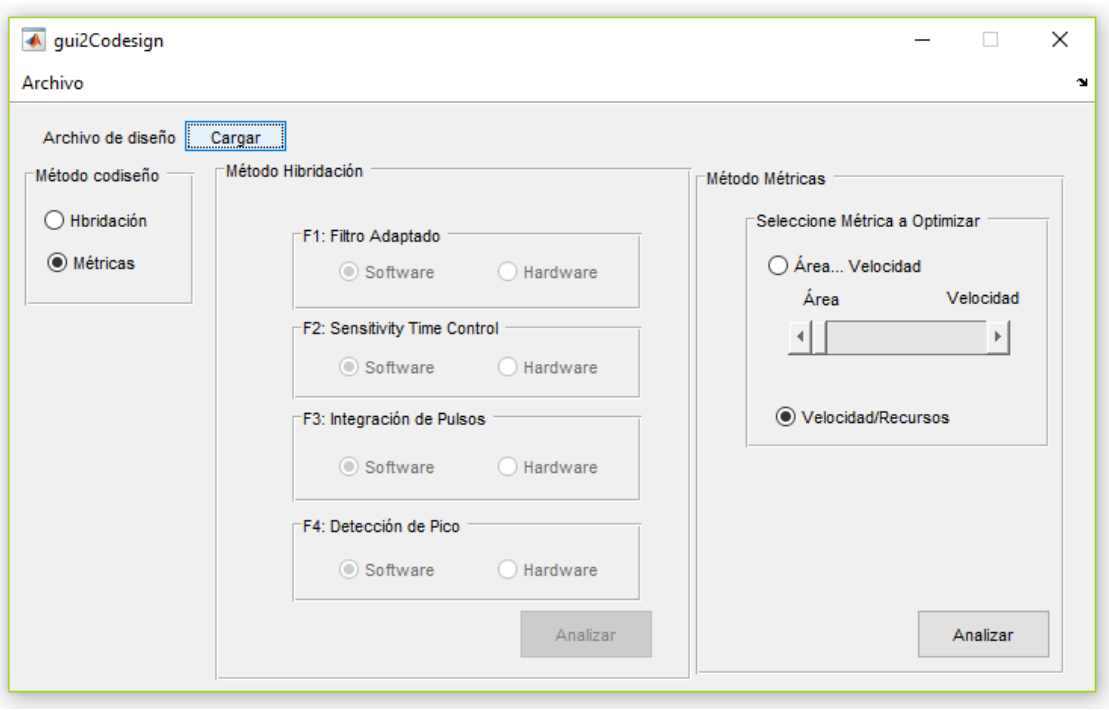


Figura 39. Se presentan los resultados de las métricas de ejecución de las funciones radar: filtro adaptado, STC, integración de pulso y detección de pico.

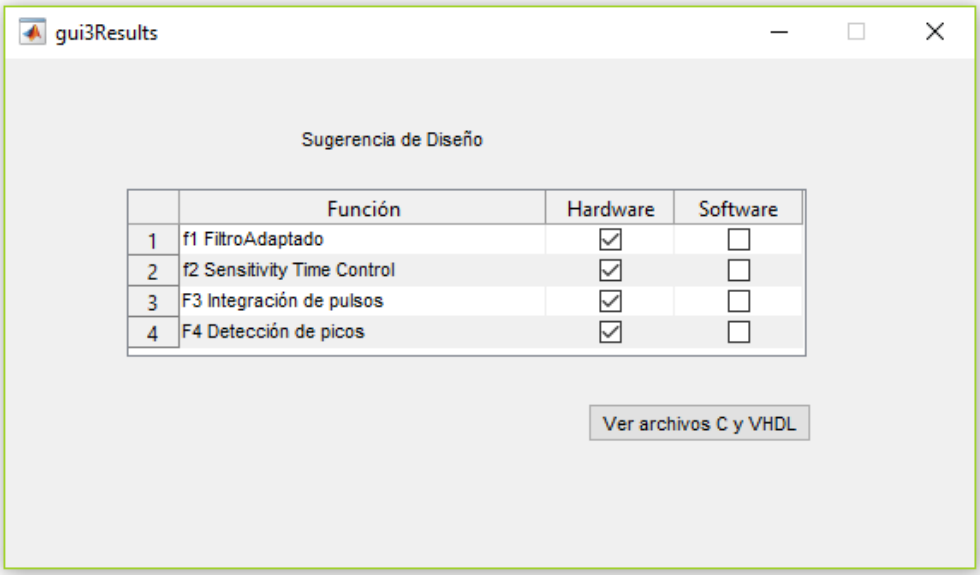


Figura 40. Resultados de la evaluación para relación Velocidad/Área

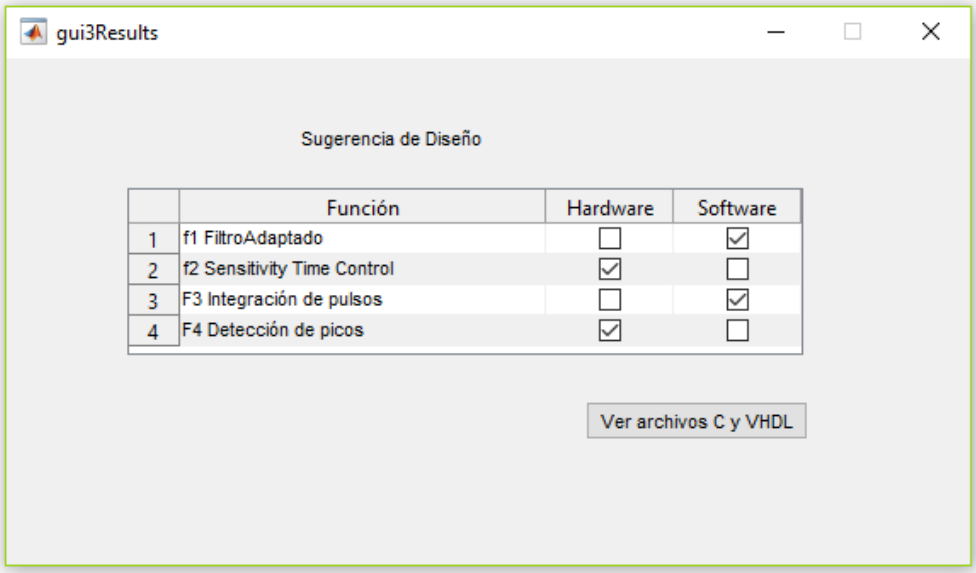


Figura 41. Resultados de la evaluación para selección mediante barra de Velocidad y Área, priorizando la velocidad en un 75%.

Matlab permite la generación automática de código, con lo que el siguiente paso sería tomar estos archivos generados por el sistema y realizar la verificación del código para implementar sobre las plataformas reales.

6. ANÁLISIS DE RESULTADOS

6.1. Ventajas de la solución

Tomando como base los resultados y aprendizajes a lo largo del proyecto, así como la realimentación de los usuarios, se plantean las principales ventajas que la solución presenta:

- La principal ventaja identificada en el desarrollo del proyecto, es la de proponer un acercamiento metodológico al codiseño hardware software, enfocado en el desarrollo de sistemas de procesamiento radar, puesto que es un tema que no había sido previamente explorado. Este primer acercamiento sirve como base para iniciar a trabajar, recopilar experiencias y así orientar trabajos futuros. De la misma forma, permite acercar a la industria, las nuevas tendencias en el desarrollo de sistemas embebidos.

- La metodología COMRADES permite aportar mecanismos de decisión para la segmentación de un diseño de procesamiento radar, en un sistema embebido. Adicionalmente, permite al diseñador experimentar sobre el impacto que pueden tener en tiempo de ejecución y recursos, diferentes opciones de segmentación.

- La herramienta desarrollada para la metodología, permite estimar el tiempo de ejecución y recursos necesarios para las funciones de procesamiento radar. Esto permite al diseñador dimensionar correctamente la arquitectura de la solución.

- Para el desarrollo de la herramienta, se identifica como acierto el uso de un entorno con lenguaje de programación que es de uso común en ingeniería electrónica como Matlab, por lo que, para los usuarios evaluados, no implicó tiempo de aprendizaje de un nuevo lenguaje de programación, ni un cambio de paradigma en la forma de modelar el procesamiento, como sí puede suceder con otras herramientas como SysML, SystemC, BCL.

- La herramienta COMRADES, se presenta como una herramienta abierta, es decir, permite la modificación del código fuente desde la misma interfaz de implementación. Esto permite a los usuarios, generar fácilmente ajustes y cambios en los parámetros de simulación de acuerdo a lo que requiera el desarrollo específico.

6.2. Limitaciones de la solución

El proyecto COMRADES, ha dejado aprendizajes y ha permitido identificar retos. A continuación, se listan las limitaciones presentes en el proyecto:

- A pesar de que el flujo de la metodología COMRADES puede aplicarse de forma general para el desarrollo de sistemas de procesamiento radar, para la herramienta fue necesario limitar la implementación, restringida a cierto tipo de sistemas radar, enfocándose principalmente en radares monoestáticos de vigilancia. Para desarrollo de procesamiento de sistemas radar de seguimiento, radares de imagen tipo SAR (*Synthetic Aperture Radar*), o radares multiestáticos, se requerirían mayores modificaciones en la herramienta. De igual manera, en la herramienta se implementó sólo un conjunto de

funciones de procesamiento, del total de funciones identificadas, con el principal objetivo de servir como base para la construcción de la metodología. Las que se identificaron que también son de importancia a implementar son: algoritmos para tratamiento del clutter, CFAR, MTD, *scan conversion*, procesamiento Doppler, asociación de *blobs* y *plots*, y, generación de *tracks*

- La herramienta COMRADES realiza una estimación en tiempo de ejecución de las funciones en plataformas hardware y plataformas software. Sin embargo, es importante tener en cuenta que la comunicación de datos entre los dispositivos hardware y los dispositivos software, implica un tiempo. Este tiempo, se ha modelado en los estudios previos como costo de comunicación entre las tareas [55]. Este tiempo no se contempla dentro del cálculo de métricas de la herramienta COMRADES.

- Los métodos usados para la segmentación de forma automática, suponen que el procesador deberá ejecutar todas las tareas de forma secuencial, por lo tanto la herramienta es válida para arquitecturas de sistemas basados en un único núcleo de procesamiento software y recursos disponibles para hardware reconfigurable.

- En la implementación de la solución se identifica la variabilidad en los resultados de las métricas, dependiendo el estilo de codificación de las funciones de procesamiento radar en el lenguaje de alto nivel (Matlab). Existen estilos de codificación de funciones que favorecen más la implementación del código en C, mientras que otros favorecen más la implementación en VHDL. Esto plantea un dilema sobre los métodos de diseño en alto nivel, ya que este debería hacerse de forma independiente a la plataforma de implementación, sin embargo, al final, el alcance de dicha abstracción estará dado por la capacidad que tenga la herramienta, de tomar ese modelo en alto nivel y convertirlo a lenguaje embebido para software o para hardware. Un ejemplo de esto está relacionado con los prototipos de funciones que se crean en Matlab, si la función recibe un arreglo de datos como parámetro de entrada, al momento de realizar la codificación automática a VHDL, Matlab interpreta que estos datos se reciben de forma simultánea como un puerto paralelo, ocupando una gran cantidad de recursos en términos de puertos entrada/salida, que en son un recurso escaso en las FPGA.

- Matlab se caracteriza por tener una amplia gama de *toolbox* y funciones. Sin embargo, del total de funciones que implementa Matlab, tan sólo un conjunto reducido de funciones permiten realizar la migración a VHDL [56]. Esto plantea una limitación, puesto que el código en alto nivel, debe sólo usar este subconjunto de funciones con el propósito de que sea sintetizable, esto sin importar si la función es más afín a una implementación en software. Esta limitación, tiene dos impactos principales: por una parte, puede implicar emplear más tiempo del que normalmente requeriría un diseñador en la implementación de una función en Matlab, y por otra parte, si finalmente la función es afín a implementación en software, esta función podría tener un rendimiento desmejorado en comparación a una que no haya sido escrita con esta restricción.

- La herramienta se implementó sobre la versión académica de Matlab, sin embargo, al momento de generar soluciones para la industria, sería necesario incurrir en los costos de licenciamiento. Existen alternativas libres como Octave, el cual maneja un lenguaje casi idéntico que Matlab, pero no incorpora los mismos *toolbox* ni tiene la posibilidad de generación automática de código en C o VHDL.

6.3. ¿Cómo se compara con otras soluciones?

A pesar de que no se encontraron estudios previos en propuestas metodológicas de codiseño, enfocadas al desarrollo de sistemas de procesamiento radar, existen trabajos previos que se han usado como base e inspiración para el desarrollo del proyecto COMRADES.

Un trabajo que se ha tomado como referencia, ha sido el propuesto por Chang, Daniel [32], en el cual plantea una metodología de codiseño para hardware, software y firmware aplicado al procesamiento digital de señales, que inicia con los requisitos y especificaciones, luego plantea definir unas restricciones para luego pasar a una arquitectura. Una vez definida la arquitectura, se crea un árbol del posible mapeo que podría existir entre las tareas o funciones con las diferentes modalidades {hardware, software, firmware}. La metodología plantea la medición de un costo en: dinero, tiempo de ejecución, potencia, espacio requerido, complejidad de diseño, entre otros. Con este valor de costo se puede plantear una tabla que permita tomar una decisión favorable en costo, la cual no necesariamente se considera la óptima, debido que el problema de segmentación es un problema combinatorio complejo [55]. La solución que plantea la metodología COMRADES, se ha basado en algunos fundamentos de este trabajo de referencia, sin embargo, al estar enfocada exclusivamente en el desarrollo de sistemas de procesamiento radar, ha permitido explorar algoritmos específicos y definir reglas de segmentación a la medida para este tipo de sistemas, además aporta una herramienta que permite materializar el esquema propuesto por la metodología.

Otro trabajo de interés ha sido el propuesto por Mischkalla y Mueller [57], en el que realizaron una aproximación a la creación de sistemas reales de procesamiento digital tomando la base de las metodologías de codiseño. Para esto utilizaron un modelado en SysML como entrada inicial de desarrollo del sistema. En la Figura 42 se muestra la comparación de las diferentes herramientas utilizadas por el trabajo de Mischkalla y Mueller, con la herramienta COMRADES. Del trabajo de Mischkalla y Mueller, se destaca el uso de herramientas estándar de modelado como SysML, sin embargo el estudio encontró problemas con el uso de varias herramientas, por lo que fue difícil identificar los errores presentados en las últimas etapas, para hacer una trazabilidad hacia las etapas iniciales de modelado [57]. Por lo tanto, una de las ventajas que aporta la metodología COMRADES es la de tratar de disminuir el número de herramientas de software a utilizar, además de apostar por entornos de desarrollo como Matlab que tienen son ampliamente usados y se prevé un continuo desarrollo a futuro. Esto adicionalmente, enfatiza la importancia de las herramientas computacionales en las tareas de codiseño.

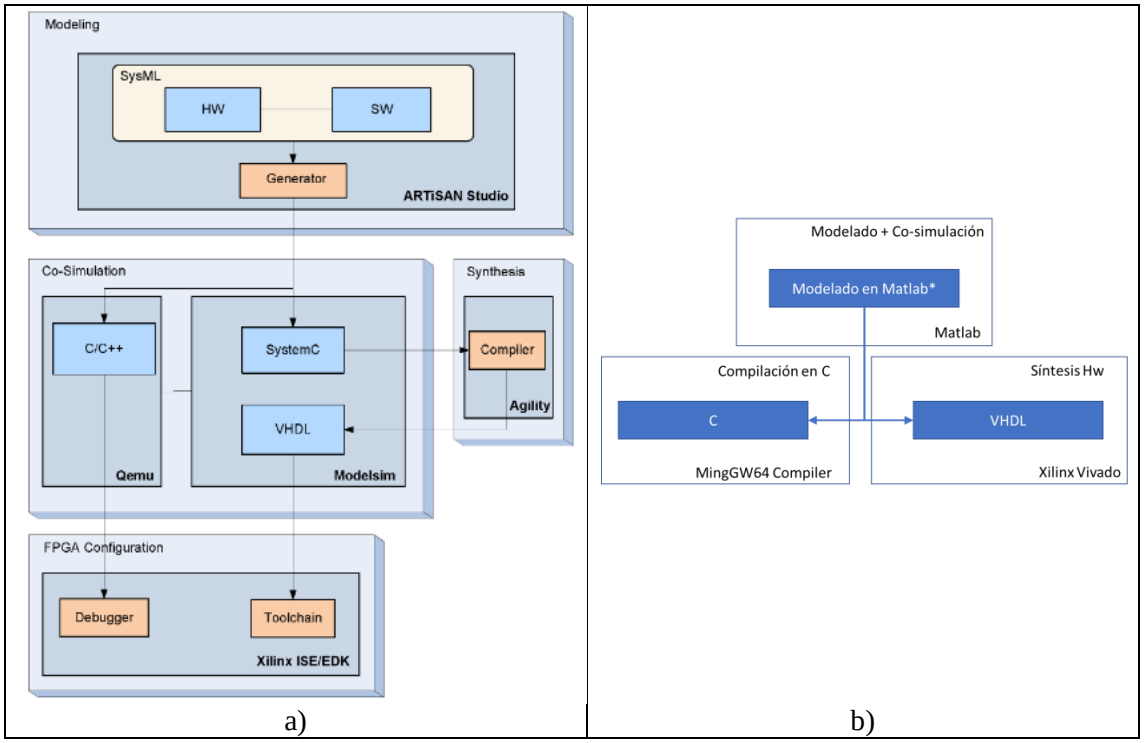


Figura 42. Herramientas usadas en los diferentes trabajos de codiseño. a) El trabajo de Mischkalla y Mueller [57]. b) Herramienta COMRADES.

7. CONCLUSIONES Y TRABAJOS FUTUROS

El proyecto COMRADES propone una metodología de codiseño hardware software, como solución al desarrollo de sistemas de procesamiento de señal radar, temática que no había sido explorada anteriormente. Se identifican las principales técnicas de procesamiento de señal radar, las principales herramientas para codiseño hardware software, y se plantea una herramienta propia con el propósito de validar la metodología. Se encuentra que el hecho de conocer de antemano las técnicas de procesamiento, así como tomar medición de métricas, nos aportará una ventaja sobre el tipo de asignación hardware o software que se le dará a las funciones o tareas determinadas en un sistema embebido.

La herramienta seleccionada para la implementación de la metodología, Matlab, plantea ventajas y limitaciones. Se presenta como ventaja el hecho de ser una herramienta de uso común en ingeniería, por lo que no implica el aprendizaje de un nuevo lenguaje o método de modelado, que se requeriría para el manejo de otras herramientas, como es el caso de modelado en SysML o el uso de la librería SystemC que implican un cambio de paradigma. Sin embargo, también se encuentran retos y limitaciones en el modelado de alto nivel. Por una parte, solo un subconjunto de las funciones de Matlab permiten la síntesis en hardware, por lo que para describir un sistema en alto nivel que luego pueda ser asignado a hardware, sólo debería usarse este conjunto de funciones. Puesto que la filosofía de codiseño está en poder escribir un código de alto nivel que luego pueda ser asignado a hardware o software, se podría perder rendimiento si se considera que el código podría ser finalmente asignado a software.

El estilo de codificación, como la forma de declarar los parámetros de entrada de una función, la escritura de las estructuras de control en bucle, entre otros, genera una variabilidad de las métricas de desempeño. Se identifica que hay estilos de codificación en código de alto nivel, que favorecen más una implementación en hardware y hay estilos que favorecen más la implementación en software. Esto plantea un reto al momento de describir un sistema en código de alto nivel.

Al comparar la solución COMRADES con trabajos previos, los cuales, a pesar de tener un alcance diferente, se toman como referencia, se identifica como fortaleza la aplicación de la metodología de codiseño para la solución de una temática particular, lo que permite delimitar la teoría de codiseño y permitir aplicarla de forma práctica. Adicionalmente, se observa que el hecho de usar una herramienta integrada como Matlab, con funcionalidad de generación automática de código y cosimulación, se presenta como una ventaja, ya que permite en una misma herramienta descubrir y realizar trazabilidad de errores, y permitir una modificación más ágil del desarrollo.

7.1. Trabajos futuros

Evaluar mecanismos de segmentación para diferentes arquitecturas de sistemas embebidos, como sistemas con múltiples procesadores y recursos de hardware. Esto aumenta la complejidad en la búsqueda de una solución, pero puede proveer un panorama más fiel a la realidad tomando varios núcleos de procesamiento, y dividendo las tareas para acelerar el tiempo de ejecución.

Proponer un modelo para agregar a la metodología en el que se contemplen los tiempos de comunicación entre procesos, así como reglas que permitan identificar afinidad entre tareas dado que alguna de ellas ya ha sido asignada, por ejemplo: puesto que la tarea A fue asignada a hardware, es preferible que la tarea B también se haga sobre hardware.

Agregar soporte a la herramienta para diferentes tipos de radares, como radares imagen de apertura sintética (SAR), que implican una gran carga de procesamiento, y que manejan técnicas diferentes a los radares de vigilancia en los que se ha enfocado la herramienta COMRADES. Adicionalmente agregar soporte para parámetros como *clutter*, modelos de atenuación atmosférica y movimiento de blancos.

8. BIBLIOGRAFÍA

[1] Petru Eles y Zebo Peng, «Hardware Software Codesign of Embedded Systems. Embedded Tutorial.», 2007. [En línea]. Disponible en: http://www.modprod.liu.se/workshop_2007/1.46549/tutorial-peng.pdf. [Accedido: 21-feb-2016].

[2] «An A+ for C++ - On Target: Embedded Systems». [En línea]. Disponible en: http://blog.vdcresearch.com/embedded_sw/2012/09/an-a-for-c.html. [Accedido: 16-feb-2016].

[3] J. Teich, «Hardware/Software Codesign: The Past, the Present, and Predicting the Future», *Proc. IEEE*, vol. 100, n.º Special Centennial Issue, pp. 1411-1430, may 2012.

[4] C. Li *et al.*, «A Review on Recent Progress of Portable Short-Range Noncontact Microwave Radar Systems», *IEEE Trans. Microw. Theory Tech.*, vol. 65, n.º 5, pp. 1692-1706, may 2017.

[5] E. Hyun *et al.*, «FPGA based signal processing module design and implementation for FMCW vehicle radar systems», en *2011 IEEE CIE International Conference on Radar (Radar)*, 2011, vol. 1, pp. 273-275.

[6] R. K. Gupta, «Hardware-software co-design: Tools for architecting systems-on-a-chip», en *Design Automation Conference, 1997. Proceedings of the ASP-DAC '97 Asia and South Pacific*, 1997, pp. 285-289.

[7] M. I. Skolnik, *Introduction to radar systems*, 2d ed. New York: McGraw-Hill, 1980.

[8] H. Meikle, *Modern Radar Systems*, Second. Arthec House, 2008.

[9] S. Xu, M. Li, y J. Suo, «Clutter processing for digital radars based on FPGA and DSP», en *International Conference on Wireless Communications Signal Processing, 2009. WCSP 2009*, 2009, pp. 1-4.

[10] «IEEE Xplore Abstract - An FPGA-based signal processing system for a 77 GHz MEMS tri-mode automotive radar». [En línea]. Disponible en: http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=5929968&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D5929968. [Accedido: 16-feb-2016].

[11] S. Ayhan *et al.*, «FPGA controlled DDS based frequency sweep generation of high linearity for FMCW radar systems», en *Microwave Conference (GeMiC), 2012 The 7th German*, 2012, pp. 1-4.

[12] Y. Wang, Q. Liu, y A. E. Fathy, «CW and Pulse #x2013;Doppler Radar Processing Based on FPGA for Human Sensing Applications», *IEEE Trans. Geosci. Remote Sens.*, vol. 51, n.º 5, pp. 3097-3107, may 2013.

[13] «IEEE Xplore Abstract - Model based design flow for implementing an anti-collision radar detection system». [En línea]. Disponible en: http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=5399285&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D5399285. [Accedido: 12-ene-2016].

[14] «IEEE Xplore Abstract - Efficient algorithm and hardware/software co-design for chirp echo estimation in ultrasonic NDE appl...» [En línea]. Disponible en: http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=6562309&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D6562309. [Accedido: 12-ene-2016].

[15] R. M. O'Donnell, «Radar Systems Engineering Lecture 4 The Radar Equation».

[16] M. B. Said, Y. H. Kacem, N. B. Amor, M. Kerboeuf, y M. Abid, «Model-Based Design of Real Time Embedded Application Reconfiguration», en *Languages, Design Methods, and Tools for Electronic System Design*, M.-M. Louërat y T. Maehne, Eds. Springer International Publishing, 2015, pp. 245-264.

[17] Y. Watanabe y S. Swan, «Clearing the clutter: Unified modeling and verification methodology for system level hardware design», 2012, pp. 21-23.

[18] Nawar Obeidat, «The Design and Development Process for Hardware/Software Embedded Systems: Example Systems and Tutorials», nov-2014. [En línea]. Disponible en: https://etd.ohiolink.edu/!etd.send_file?accession=ucin1416233177&disposition=inline. [Accedido: 25-nov-2015].
[19] F. Herrera *et al.*, «The COMPLEX methodology for UML/MARTE Modeling and design space exploration of embedded systems», *J. Syst. Archit.*, vol. 60, n.º 1, pp. 55-78, Enero 2014.
[20] M. King, J. Hicks, y J. Ankcorn, «Software-Driven Hardware Development», en *Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, New York, NY, USA, 2015, pp. 13–22.
[21] G. Rong, T. Liu, M. Xie, J. Chen, C. Ma, y D. Shao, «Processes for Embedded Systems Development: Preliminary Results from a Systematic Review», en *Proceedings of the 2014 International Conference on Software and System Process*, New York, NY, USA, 2014, pp. 94–98.
[22] M. Ben Youssef, J.-F. Boland, G. Nicolescu, G. Bois, y J. Hugues, «Bridging the high-level model to execution platform for design space exploration and implementation», presentado en Embedded Real-Time Software and Systems - ERTS^2 2014, 2014.
[23] T. Ball, R. Bodik, S. Krishnamurthi, B. S. Lerner, y G. Morrisett, *1st Summit on Advances in Programming Languages (SNAPL 2015)*. Wadern: Schloss Dagstuhl - Leibniz-Zentrum für Informatik GmbH, 2015.
[24] M. King, N. Dave, y Arvind, «Automatic generation of hardware/software interfaces», pp. 325–336, 2012.
[25] M. D. King, «A methodology for hardware-software codesign», Thesis, Massachusetts Institute of Technology, 2013.
[26] M. Meier, M. Breddemann, y O. Spinczyk, «Interfacing the hardware API with a feature-based operating system family», *J. Syst. Archit.*
[27] M. Meier, M. Breddemann, y O. Spinczyk, «Hardware APIs: A Software-Centric Approach for Automated Derivation of MPSoC Hardware Structures Based on Static Code Analysis», en *Architecture of Computing Systems – ARCS 2014*, E. Maehle, K. Römer, W. Karl, y E. Tovar, Eds. Springer International Publishing, 2014, pp. 111-122.
[28] G. L. B. L. Osamu Saotome, «CoH-Agile: Proposal of methodology for development HW/SW embedded», *Int. J. Enhanc. Re Search Sci. Technol. Eng.*, vol. 4, may 2015.
[29] G. L. Bertoze Lima, G. A. Lopes Ferreira, O. Saotome, A. M. Da Cunha, y L. A. Vieira Dias, «Hardware Development: Agile and Co-Design», en *2015 12th International Conference on Information Technology - New Generations (ITNG)*, 2015, pp. 784-787.
[30] M. Kaisti *et al.*, «Agile methods for embedded systems development - a literature review and a mapping study», *EURASIP J. Embed. Syst.*, vol. 2013, n.º 1, pp. 1-16, nov. 2013.
[31] J. Teich, «Hardware/Software Codesign: The Past, the Present, and Predicting the Future», *Proc. IEEE*, vol. 100, n.º Special Centennial Issue, pp. 1411-1430, may 2012.
[32] D. Y. Chang, «A systematic software, firmware, and hardware codesign methodology for digital signal processing», Thesis, Monterey, California: Naval Postgraduate School, 2014.
[33] H. Sahlbach, D. Thiele, y R. Ernst, «A system-level FPGA design methodology for video applications with weakly-programmable hardware components», *J. Real-Time Image Process.*, pp. 1-19, mar. 2014.
[34] A. Platunov, A. Penskoi, y A. Kluchev, «The architectural specification of embedded systems», en *2014 3rd Mediterranean Conference on Embedded Computing (MECO)*, 2014, pp. 48-51.
[35] O. Feki, T. Grandpierre, M. Akil, N. Masmoudi, y Y. Sorel, «SynDEx-Mix: A hardware/software partitioning CAD tool», en *2014 15th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA)*, 2014, pp. 247-252.
[36] M. A. El-Moursy, A. Sheirah, M. Safar, y A. Salem, «Efficient embedded SoC hardware/software codesign using virtual platform», en *Design Test Symposium (IDT), 2014 9th International*, 2014, pp. 36-38.

[37] «EBSCOhost | 97888220 | Hardware-Software Partitioning Algorithm Based on Binary Search Trees And Genetic Algorithm To Optimize Logic Area For SOPC.» [En línea].. [Accedido: 23-nov-2015].
[38] A. B. Trindade y L. C. Cordeiro, «Applying SMT-based verification to hardware/software partitioning in embedded systems», *Des. Autom. Embed. Syst.*, pp. 1-19, abr. 2015.
[39] A. Gonzalez, R. Angel, y E. Gonzalez, «BDI concurrent architecture orientedto goal managment», en *Computing Colombian Conference (8CCC), 2013 8th*, 2013, pp. 1-6.
[40] «IEEE Xplore Abstract - IBM z13 circuit design and methodology». [En línea]. Disponible en: http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=7175096&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D7175096. [Accedido: 25-nov-2015].
[41] N. Govil y S. R. Chowdhury, «High performance and low cost implementation of Fast Fourier Transform algorithm based on Hardware Software co-design», en *2014 IEEE Region 10 Symposium*, 2014, pp. 403-407.
[42] Y. H. Shiau, C. H. Li, Z. H. Wang, y Y. T. Guo, «Hardware and Software Co-design of the Moving Object Tracking System», en *2014 Tenth International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP)*, 2014, pp. 305-308.
[43] M. Shand, «A case study of hardware software co-design in a consumer ASIC», en *International Conference on Formal Methods and Models for Co-Design*, 2011.
[44] «Welcome to Artisan Studio 7.4 :: Atego». [En línea]. Disponible en: http://www.atego.com/campaigns/artisan-studio-7-4/. [Accedido: 18-may-2017].
[45] «SpaceStudio by Space Codesign Systems Inc. - Accelerate C/C++ application with FPGAs». [En línea]. Disponible en: http://www.spacecodesign.com/. [Accedido: 18-may-2017].
[46] «FPGA Design and Codesign - MATLAB & Simulink Solutions - MATLAB & Simulink». [En línea]. Disponible en: https://www.mathworks.com/solutions/fpga-design.html. [Accedido: 18-may-2017].
[47] «Vivado High-Level Synthesis». [En línea]. Disponible en: https://www.xilinx.com/products/design-tools/vivado/integration/esl-design.html. [Accedido: 18-may-2017].
[48] «Welcome Page | Open Virtual Platforms». [En línea]. Disponible en: http://www.ovpworld.org/. [Accedido: 18-may-2017].
[49] «solidThinking Embed | Visual Environment for Embedded Systems». [En línea]. Disponible en: http://www.solidthinking.com/embed_land.html. [Accedido: 18-may-2017].
[50] «ESL | Electronic System Level Design - Mentor Graphics». [En línea]. Disponible en: https://www.mentor.com/esl/. [Accedido: 18-may-2017].
[51] «Cycle Models | ARM SoC Designer – ARM Developer». [En línea]. Disponible en: https://developer.arm.com/products/system-design/cycle-models/arm-soc-designer. [Accedido: 18-may-2017].
[52] «SystemVue Electronic System-Level (ESL) Design Software | Keysight (formerly Agilent's Electronic Measurement)». [En línea]. Disponible en: http://www.keysight.com/en/pc-1297131/systemvue-electronic-system-level-esl-design-software?nid=-34264.0&cc=CO&lc=eng. [Accedido: 18-may-2017].
[53] T. Milligan, *Modern Antenna Design*, Second Edition. IEEE Press, Wiley Interscience, 2005.
[54] «Designing a Basic Monostatic Pulse Radar - MATLAB & Simulink Example - MathWorks España». [En línea]. Disponible en: https://es.mathworks.com/help/phased/examples/designing-a-basic-monostatic-pulse-radar.html. [Accedido: 22-jun-2017].
[55] W. Jigang, T. Srikanthan, y T. Jiao, «Algorithmic aspects for functional partitioning and scheduling in hardware/software co-design», *Des. Autom. Embed. Syst.*, vol. 12, n.º 4, pp. 345-375, dic. 2008.
[56] Mathworks, *Matlab HDL coder user manual*. 2016.

[57] F. Mischkalla, D. He, y W. Mueller, «Closing the gap between UML-based modeling, simulation and synthesis of combined HW/SW systems», en *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2010*, 2010, pp. 1201–1206.