UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA

ESCUELA UNIVERSITARIA DE INGENIENIERÍA TÉCNICA DE TELECOMUNICACIÓN

PROYECTO FIN DE CARRERA

DESARROLO DE UNA HERRAMIENTA WEB PARA LA SELECCIÓN DE BOBINAS INTEGRADAS

Titulación: Sistemas de Telecomunicación
Autor: D. José Cristo González González
Tutores: Dr. D. Javier del Pino Suárez
Dr. D. Luis Miguel Hernández Acosta

Fecha: Julio 2009

UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA

ESCUELA UNIVERSITARIA DE INGENIENIERÍA TÉCNICA DE TELECOMUNICACIÓN

PROYECTO FIN DE CARRERA

DESARROLO DE UNA HERRAMIENTA WEB PARA LA SELECCIÓN DE BOBINAS INTEGRADAS

Presidente: Secretario: Vocal:

Tutores: Autor:

Nota:

Titulación: **Sistemas de Telecomunicación**
Autor: **D. José Cristo González González**
Tutores: **Dr. D. Javier del Pino Suárez**
Dr. D. Luis Miguel Hernández Acosta
Fecha: **Julio 2009**

Índice

Capítulo 1

INTRODUCIÓN

1.1 Introducción

En esta última década la tecnología necesaria para el desarrollo de las comunicaciones inalámbricas ha sufrido un auge considerable, observándose esto en las redes inalámbricas de área local (WLAN: W*ireless Local Area Networks*) , (WIFI: Wireless Fidelity) , o terminales de telefonía móvil, que además de las funciones básicas incorporan reproducción de música MP3, reproducción multimedia MP4, correo electrónico, agenda electrónica PDA, fotografía digital, navegación por Internet y hasta TV digital. La demanda de estos dispositivos es muy alta, hasta tal punto que en países como Luxemburgo, Italia, Hong-Kong o Taiwán ya hay más de un teléfono móvil en uso por habitante [1].

El mercado exige que los terminales de acceso a estas redes inalámbricas sean pequeños, baratos y de bajo consumo de potencia, y que se puedan producir de forma masiva, lo cual puede conseguirse aumentando el nivel de integración de la circuitería que forma parte de estos dispositivos, lo cual conduce al uso de tecnologías estándar de silicio: CMOS, BiCMOS y SiGe. El funcionamiento de los circuitos MOS a alta frecuencia (>1 GHz) ha mejorado tanto que el transceptor de radiofrecuencia (RF) completo, incluyendo el cabezal de RF y los circuitos para el procesamiento en banda-base, ya se puede integrar en un mismo chip.

Con el transcurso de los años, los Circuitos Integrados están constantemente migrando a tamaños más pequeños con mejores características, permitiendo que mayor cantidad de circuitos sean empaquetados en cada chip. Al mismo tiempo que el tamaño se comprime prácticamente todo se mejora (el coste y el consumo de energía disminuyen y la velocidad aumenta). Aunque estas ganancias son aparentemente para el

usuario final, existe una feroz competencia entre los fabricantes para utilizar geometrías cada vez más delgadas.

Teniendo en cuenta lo anteriormente comentado en este proyecto se realizará utilizando las herramientas y tecnologías a las que tiene acceso el Instituto Universitario de Microelectrónica Aplicada (IUMA) de la ULPGC, para el desarrollo de una herramienta Web para la simulación del comportamiento de bobinas integradas y su correcta selección según las necesidades en los diferentes supuestos. Se inscribe, además, en una de las líneas de investigación que tiene en marcha la División de Tecnología Microelectrónica de IUMA.

1.2 Inductores integrados

Un inductor integrado se diseña comúnmente colocando una pista de metal enrollada de forma espiral sobre un sustrato determinado. Uno de los dos extremos de la espiral, que serán los terminales del inductor, queda en el interior de la misma, por lo que necesitaremos al menos dos niveles de metal para poder tener acceso a dicho terminal. A este fragmento de metal que permite sacar el puerto del interior del inductor se le denomina *underpass* o *cross-under*. Los parámetros geométricos que definen un inductor son el número de lados, el radio externo (r_{EXT}), el ancho de la pista (w), la separación entre las pistas (*s*), y el número de vueltas (*n*). En la Figura 1.1 se muestra el *layout* de una bobina espiral cuadrada simple, en donde se señalan estos parámetros y se puede apreciar la disposición del *underpass.*

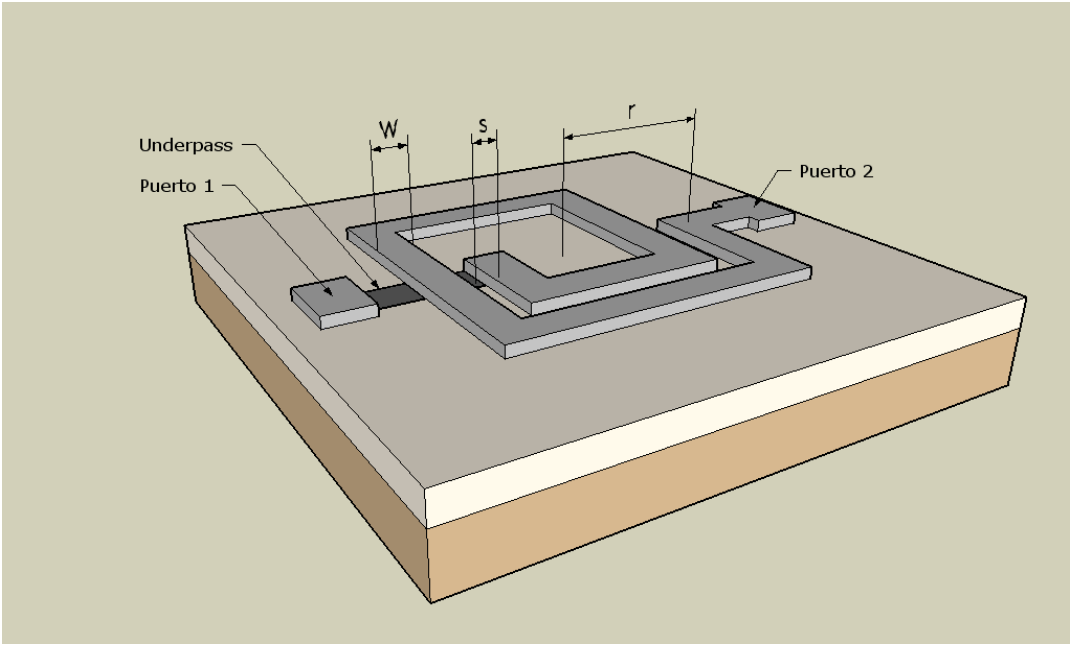

Figura 1.1 *Layout* de una bobina espiral cuadrada simple.

El inductor, también llamado bobina, es un componente pasivo que almacena energía en forma de campo magnético. Las dos características que definen cómo desempeña esta función un inductor determinado son la inductancia y el factor de calidad. La inductancia, L, da una medida de la cantidad de energía que puede almacenar el campo magnético creado en la bobina. El factor de calidad, Q, representa la relación que existe entre la energía almacenada por el dispositivo y la energía disipada. Es decir, representa una figura de mérito relacionada con la eficiencia en el almacenamiento de energía:

$$Q = 2 \cdot \pi \cdot \frac{E_{almacenada}}{E_{disipada}} \tag{1.1}$$

El factor de calidad puede ser calculado de diferentes maneras dependiendo del uso que se le vaya a dar al inductor, aunque todas están relacionadas entre sí [2]. La definición más comúnmente empleada es la que viene dada por la relación entre la parte imaginaria y la real del parámetro Y_{11}, que es la admitancia que se ve desde el puerto 1 cuando el puerto 2 está conectado a tierra [3].

$$Q = -\frac{Im(Y_{11})}{Re(Y_{11})} \tag{1.2}$$

Conseguir un inductor integrado con un factor de calidad alto depende principalmente de la tecnología que se utilice para su integración. Si se trata de un sustrato de resistividad alta, como por ejemplo los cerámicos [4], plásticos [5], de cristal [6] o de GaAs [7] se pueden conseguir bobinas de alta calidad. Otra forma de hacerlo es utilizar algún proceso de fabricación no estándar, como los que se emplean en los inductores toroidales [8] [9] [10] [11] y solenoidales [12] [13] [14]. En estos casos generalmente se añaden núcleos magnéticos sobre los que se hacen pasar metalizaciones de cobre o aluminio. Aunque los resultados son muy buenos, las técnicas para conseguir este tipo de inductores son muy complejas, y resultan demasiado caras para hacer viable su comercialización.

Por desgracia el silicio es muy conductivo y se generan muchas pérdidas en el sustrato que, sumadas a las que se producen en los metales, hacen que la calidad de los inductores se deteriore notablemente. En el siguiente apartado se profundiza en los fenómenos físicos que dan origen a estas pérdidas y en su influencia sobre las prestaciones del inductor.

1.2.1 Pérdidas en inductores integrados sobre silicio

Lo primero que hay que estudiar para identificar las pérdidas que se producen en un inductor integrado sobre silicio, son los fenómenos físicos que se producen en él. La Figura 1.2 muestra los campos eléctricos y magnéticos que se generan en el inductor al aplicar una tensión variable en los extremos de la espiral:

- *B(t)* es el campo magnético del inductor, que está originado por la corriente alterna que circula por la espiral y es el que genera el comportamiento inductivo del dispositivo. Además este campo atraviesa tanto el sustrato como las pistas de la espira, induciendo en ambos las llamadas corrientes de torbellino que estudiaremos a continuación.
- $E_1(t)$ es el campo eléctrico en las pistas de la espiral. Produce la corriente de conducción, que genera pérdidas óhmicas debido a la resistividad del metal.
- $E_2(t)$ es el campo eléctrico entre las pistas de la espiral. Se genera por la diferencia de tensión entre los conductores de distintas vueltas, y da lugar a una capacidad parásita entre ellos debido al óxido de silicio que los rodea, que actúa como dieléctrico.
- $E_3(t)$ es el campo eléctrico entre la espiral y el sustrato, generado por la diferencia de tensión entre ambos. Produce el acoplamiento capacitivo entre la espiral y el sustrato además de pérdidas óhmicas en este último.
- $E_4(t)$ es el campo eléctrico entre la espiral y el *underpass*. Genera una capacidad parásita asociada en paralelo a la bobina.

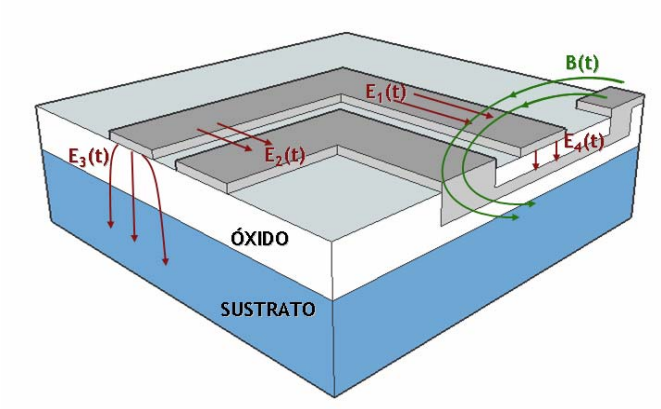

Figura 1.2 Campos que se generan en un inductor integrado.

Estos campos son el origen de las importantes pérdidas que se dan en un inductor y que degradan el factor de calidad. Se pueden agrupar en dos tipos: las que se generan en las pistas de metal y las que se generan en el sustrato.

1.2.1.1 Pérdidas en las pistas

Cuando por un conductor circula una corriente a frecuencias bajas, la distribución de esta corriente por la superficie del conductor es uniforme. El valor de la resistencia que presenta el conductor al paso de esta corriente se puede calcular fácilmente por medio de la expresión (1.3).

$$R = \rho \cdot \frac{l}{A} \tag{1.3}$$

donde ρ es la resistividad del conductor, y l y A su longitud y área respectivamente.

Sin embargo, a medida que la frecuencia aumenta se generan nuevos efectos más complejos que redistribuyen la corriente a lo largo del conductor.

Por un lado, el campo magnético inducido en el propio conductor genera una fuerza electromotriz que actúa sobre la corriente del conductor modificando la dirección de movimiento de la carga. Esto hace que la corriente circule por una sección cercana a los bordes del conductor, tal y como muestra la Figura 1.3 para el caso de un conductor cilíndrico. Al disminuir la superficie por la que circula el mismo flujo de intensidad la resistencia del conductor aumenta. A este efecto se le conoce como efecto pelicular (*skin effect*).

Figura 1.3 Efecto pelicular en un conductor de sección circular.

El parámetro que evalúa esta redistribución de corriente es la profundidad pelicular δ, que se define como el espesor equivalente de un conductor hueco que tiene la misma resistencia a una frecuencia determinada. Cuantificar δ de forma analítica no es tarea fácil, y en la literatura sólo se encuentran expresiones fiables para conductores de geometría circular como el de la Figura 1.3. En este caso la profundidad pelicular viene dada por (1.4).

$$\delta = \sqrt{\frac{2}{\mu \cdot \sigma \cdot \omega}} \tag{1.4}$$

donde μ es la permeabilidad magnética del material, σ la conductividad y ω la frecuencia angular.

Cuando más adelante se hable del modelo paramétrico, veremos que no se conoce una expresión similar para conductores planos como en el caso de los inductores integrados. Por tanto, se recurre a expresiones semi-analíticas o empíricas que generalmente utilizan parámetros de ajuste.

Por otra parte el campo magnético variable ($B(t)_{principal}$), que fluye en la bobina tal y como muestra la Figura 1.4 (a), puede atravesar sus vueltas interiores. De acuerdo con la ley de Faraday-Lenz, esto crea un nuevo campo eléctrico que a su vez origina bucles de corriente conocidos como corrientes de torbellino (*eddy currents*) (ver Figura 1.4 (b)). Estas nuevas corrientes a su vez generan un nuevo campo magnético ($B(t)_{torbellino}$) en la misma dirección que $B(t)_{principal}$ pero en sentido opuesto, tal y como muestra la Figura 1.4 (c). La magnitud del campo eléctrico inducido, y por tanto la de $I_{torbellino}$, es proporcional a la derivada del campo magnético principal con respecto al tiempo, por lo que este efecto será más significativo a altas frecuencias.

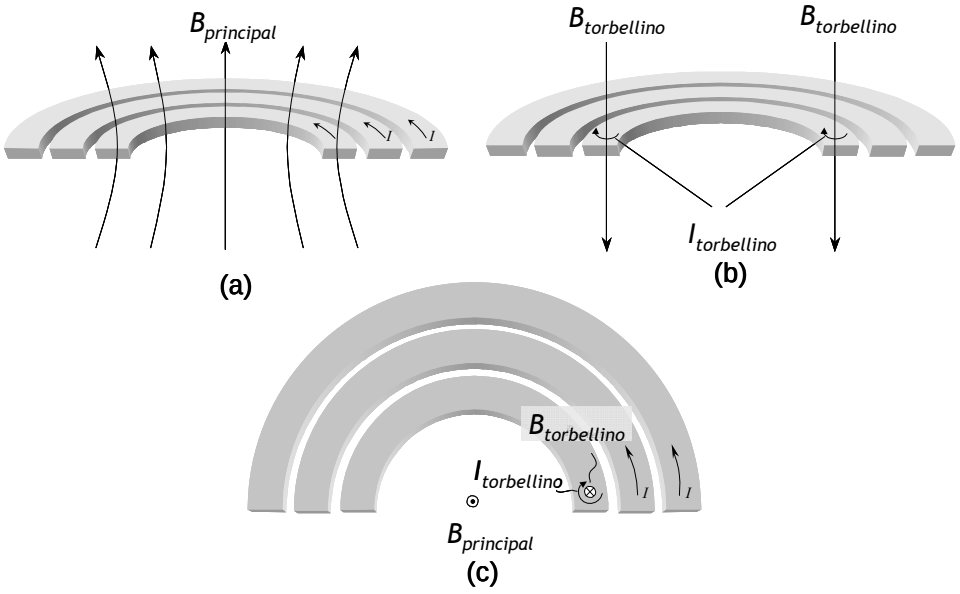

Figura 1.4 Generación de corrientes de torbellino en las pistas interiores de un inductor plano.

El efecto de las corrientes de torbellino, también denominado efecto de proximidad, adquiere más importancia cuando las vueltas de metal de un inductor dejan un agujero central pequeño. En estos casos gran parte del campo magnético de mayor magnitud (el máximo se da en el centro del inductor) atraviesa estas vueltas interiores, y las corrientes de torbellino son lo suficientemente importantes como para redistribuir el

flujo de corriente. Así, en el lado interior de las vueltas centrales, la corriente principal y las de torbellino fluyen en el mismo sentido, con lo que la densidad de corriente es mayor. Por el contrario, en el lado exterior, ambas corrientes van en sentido contrario y por ello la densidad de corriente resultante es menor. Como resultado de este proceso, la corriente en las vueltas centrales se concentra en el lado interior del conductor produciendo un aumento de la resistencia serie asociada a dichas vueltas.

Craninckx analiza en [14] la aportación de cada vuelta de un inductor integrado a la resistencia total del mismo. En un principio se podría pensar que las vueltas exteriores son las que más resistencia tienen por ser más largas. Sin embargo, el aumento de la resistencia debido al efecto de las corrientes de torbellino a frecuencias altas hace que gran parte de las pérdidas resistivas totales del inductor se deba a las vueltas interiores.

1.2.1.2 Pérdidas en el sustrato

Las pérdidas que se generan en el sustrato se originan por dos campos diferentes.

Por un lado, en el sustrato aparecen nuevas corrientes generadas de la misma manera que las corrientes de torbellino en las pistas interiores del inductor, como ilustra la Figura 1.5. En ella podemos ver un corte transversal del inductor, incluyendo la parte de óxido y el propio sustrato. Tal y como fluye el campo magnético *B(t)* en el inductor, induce por la ley de Faraday-Lenz un campo eléctrico que fluye en una especie de bobina imaginaria en el interior del sustrato. Este campo eléctrico a su vez genera una corriente de torbellino I_{subs} que se opone a la corriente del inductor (*I*).

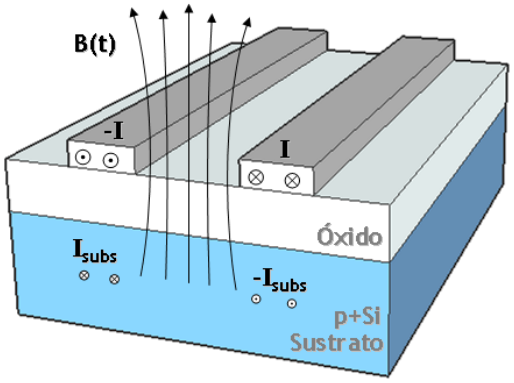

Figura 1.5 Generación de corrientes de torbellino en el sustrato.

Si el sustrato presenta una resistividad alta (>10 Ω·cm), el campo eléctrico inducido en el sustrato y la corriente que éste genera serán pequeños [15]. Por lo tanto se podrán despreciar, y el factor de calidad del inductor vendrá dado principalmente por las

pérdidas en las pistas de metal. Si por el contrario el sustrato es muy conductivo, las corrientes de torbellino inducidas en el sustrato influirán en gran medida en el factor de calidad del inductor. Además de aumentar la resistencia, estas corrientes disminuirán la inductancia total del inductor. Como se explicó en el apartado anterior, ambos efectos serán más importantes conforme aumente la frecuencia.

Por otro lado aparecen las pérdidas creadas debido a los efectos inducidos eléctricamente en el sustrato. Cuando se utiliza una tecnología convencional, en la capa de óxido que se encuentra entre la espiral y el sustrato aparece una capacidad parásita. A esta hay que añadirle la que se crea en el propio sustrato. Junto con la inductancia que genera la espiral, estas capacidades originan una frecuencia de resonancia por encima de la cual el inductor deja de funcionar como tal para pasar a ser un condensador.

El campo eléctrico en el sustrato resistivo genera también pérdidas óhmnicas que deterioran el factor de calidad. Más adelante veremos las soluciones a las que recurren los diseñadores para minimizar estas pérdidas.

1.2.2 Métodos para la mejora de la calidad

En 1990 Nguyen y Meyer propusieron el primer inductor integrado sobre silicio [15], con un factor de calidad inferior a 5. Las tecnologías empleadas ya habían evolucionado lo suficiente como para incluir varios metales y capas de óxido gruesas para aislar el sustrato de la espiral metálica. Esto, junto con el hecho de que se comenzaba a trabajar a frecuencias más altas, hizo que se pudieran conseguir bobinas con factores de calidad aceptables.

Desde entonces numerosos grupos se han dedicado al estudio de los inductores integrados sobre silicio. Uno de los temas en los que más hincapié se ha hecho es la búsqueda de nuevas técnicas de diseño para minimizar las pérdidas que, como acabamos de ver, degradan el factor de calidad.

Las soluciones que se proponen para reducir las pérdidas resistivas en la espiral metálica están relacionadas en su mayoría con modificaciones del *layout* del inductor:

- Utilizar inductores multinivel, con varios metales en paralelo, de forma que disminuye la resistencia total del conductor [3] [16]. Esto, sin embargo, lleva a utilizar metales cercanos al sustrato y por tanto aumenta las capacidades parásitas disminuyendo la frecuencia máxima de trabajo. Basándose en esta idea, las tecnologías que se emplean actualmente

incluyen en el nivel más alejado del sustrato metales más gruesos pensados para el diseño de inductores de calidad.

- Para evitar la aparición de las corrientes de torbellino en las pistas interiores, algunos autores proponen modificar las vueltas internas del inductor, por ejemplo haciéndolas más estrechas que el resto [14] [17]. Pero esto incrementa la resistencia en continua del inductor, con el consiguiente deterioro de Q.
- La solución más empleada y que nosotros utilizaremos, pasa por eliminar estas vueltas interiores de las bobinas, ya que apenas contribuyen al aumento de la inductancia total y disminuyen notablemente la calidad del inductor [14].

Para minimizar las pérdidas que se generan en el sustrato muchos autores han presentado técnicas que modifican o incluso eliminan el silicio que queda por debajo del inductor. Otros se decantan por técnicas más convencionales y muy extendidas en la actualidad, como es el uso de apantallamiento en el sustrato:

- Se pueden reducir las pérdidas del sustrato de manera local colocando un pozo n polarizado bajo el inductor [18], introduciendo una capa gruesa de óxido enterrada en el sustrato [19], bombardeándolo con protones [20], o formando un silicio poroso [21] [22], por nombrar algunos ejemplos.
- Un gran número de publicaciones se basan en el vaciado selectivo del sustrato bajo el inductor, de forma que prácticamente se eliminan las pérdidas generadas en él [23] [24] [25] [26] [27] [28] [29] [7] [30]. Sin embargo, aunque la calidad del inductor mejora mucho, el uso de estas técnicas es inviable desde el punto de vista de su comercialización debido al elevado coste que conlleva.
- Muchos diseñadores utilizan el apantallamiento del sustrato (*Patterned Ground Shield*, PGS) para eliminar las pérdidas óhmnicas debidas al campo eléctrico [31] [32] [33] [34] [35]. Consiste en insertar un plano conectado a tierra entre la espira y el sustrato. Puede ser de metal o bien de algún polisilicio más resistivo que contenga la tecnología con la que se trabaje [32]. Aunque el factor de calidad puede aumentar hasta en un 30%, esta técnica conlleva un aumento considerable de la capacidad parásita a tierra, y por tanto una disminución considerable de la frecuencia de resonancia del inductor [36].

Puesto que cada vez se trabaja a frecuencias más altas, los esfuerzos de muchos investigadores han ido encaminados hacia el diseño de nuevas estructuras que presenten frecuencias de resonancia mayores. Así aparecen, por ejemplo, los inductores apilados [37] [38] [39] [40] o los tridimensionales [41], que pretenden maximizar la inductancia y al mismo tiempo reducir el área ocupada.

1.2.3 Estudio de la geometría del inductor

Una vez que se han explicado los efectos parásitos que aparecen en un inductor, se puede analizar detalladamente cómo influye la geometría sobre su funcionamiento. La Tabla 1.1 recoge las tendencias del factor de calidad máximo (Q_{MAX}), el valor de la inductancia (L), y la frecuencia de resonancia (f_{RES}) cuando se varía uno de los parámetros geométricos (ver Figura 1.1) manteniendo el resto constante.

Tabla 1.1 Influencia de la geometría del inductor sobre sus prestaciones

	Q_{MAX}	L	f_{RES}
radio externo (r_{EXT})	↓	↑	↓
anchura de pista (w)	↑	↓	↓
espaciado entre pistas (s)	--	↓	↑
número de vueltas (n)	↓	↑	↓

Al aumentar el radio externo del inductor, aumenta el área ocupada, y por tanto hay más pérdidas resistivas y disminuye el factor de calidad máximo. Lo mismo ocurre con la frecuencia de resonancia, que baja debido a que la capacidad parásita es proporcional al área ocupada. La inductancia aumenta ya que el hueco interior del inductor se hace mayor.

Si se mantienen todos los parámetros intactos y sólo se aumenta el ancho de la pista del inductor, la resistencia asociada a las pistas se reduce, y en consecuencia el factor de calidad máximo aumenta. La inductancia sin embargo disminuye porque el agujero central del inductor se cierra. Lo mismo ocurre con la frecuencia de resonancia, puesto

que las capacidades parásitas se hacen mayores al aumentar el área de metal en la bobina.

El aumento del espaciado entre pistas no produce variaciones significativas en el factor de calidad máximo del inductor [42] [43]. Sí influye en su inductancia, que disminuye al ser más débil la inductancia mutua entre pistas y disminuir el área encerrada. La frecuencia de resonancia aumenta ligeramente ya que los acoplamientos capacitivos entre pistas disminuyen si se aumenta la distancia entre ellas.

Cuando se añaden vueltas en un inductor se añade más metal, lo que implica que hay más resistencia y disminuye el factor de calidad. Además la capacidad parásita aumenta y por tanto la frecuencia de resonancia también disminuye. La inductancia sin embargo se incrementa porque con más metal se genera más acoplamiento inductivo.

Con respecto al número de lados, aunque no se ha incluido en la Tabla 1.1, estudios anteriores sobre inductores integrados demuestran que la forma óptima de los mismos, desde el punto de vista del factor de calidad, es la geometría circular [44]. Sin embargo, las tecnologías generalmente no permiten inductores circulares, por lo que se recurre a geometrías octogonales, en las que el factor de calidad es similar al anterior.

Al margen de los métodos para la mejora de la calidad que vimos en el apartado anterior, muchos autores tratan de optimizar Q eligiendo las dimensiones del inductor más apropiadas para cada aplicación. Esta selección se basa en la experiencia previa del diseñador, en los resultados de la simulación de la bobina utilizando simuladores electromagnéticos, y/o en el uso de modelos paramétricos que predigan el funcionamiento del inductor. Todos estos aspectos, igualmente importantes para el diseño de inductores integrados, se van a estudiar a lo largo de este trabajo.

1.2.4 Modelo equivalente de un inductor. Modelo clásico

Un inductor integrado puede ser representado mediante un circuito formado por una serie de componentes electrónicos que simulen los fenómenos electromagnéticos que se producen en este elemento pasivo. De esta forma, aparecen elementos que tratan de definir el comportamiento y los efectos producidos por las pistas metálicas, de éstas entre sí y con el sustrato semiconductor y de este último material.

En la Figura 1.6 se puede observar el modelo clásico de dos puertos o modelo de banda ancha en π. Este modelo trata de dar respuesta a los fenómenos físicos vistos anteriormente en todo el rango de la frecuencia.

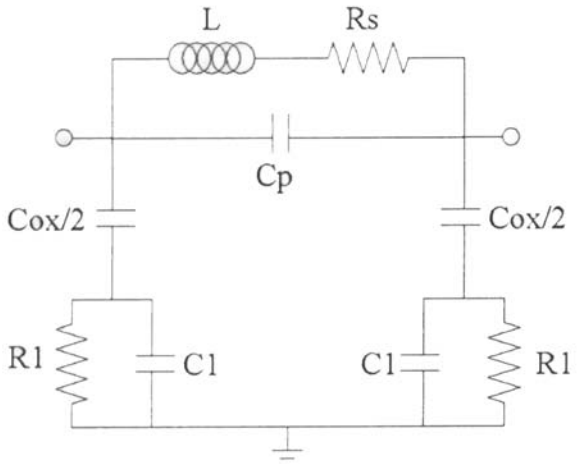

Figura 1.6 Modelo clásico de dos puertos de un inductor integrado.

La inductancia se modela a través del elemento L_S. Las pérdidas óhmicas generadas en las pistas debido a la resistividad de los inductores se modelan mediante una resistencia, R_S, en serie con el elemento anterior. En paralelo a esta rama aparece el condensador C_P, el cual modela el efecto capacitivo que aparece entre las pistas de la espira (capacidad lateral) y entre éstas y el *crossunder*.

Los elementos concentrados descritos anteriormente describen los efectos de las propias pistas que conforman el inductor. Sin embargo, también aparecen una serie de efectos debido a la presencia del sustrato. De esta forma, C_{ox1} y C_{ox2} modelan las capacidades del óxido que existe entre la espiral y el sustrato semiconductor. Por otro lado, los condensadores C_{sub1} y C_{sub2} reflejan las capacidades del propio sustrato y las resistencias R_{sub1} y R_{sub2} las pérdidas óhmicas del mismo.

El circuito anteriormente descrito no es simétrico ya que no lo es el *layout* de la inductancia integrada (ver Figura 1.1). La presencia de la pista de *underpass* cerca de uno de los puertos de la inductancia provoca una variación en el acoplamiento capacitivo con el sustrato entre ambos extremos. Consecuentemente, se debe tener unos valores ligeramente distintos entre los de un extremo (C_{ox1}, C_{sub1} y R_{sub1}) y otro (C_{ox2}, C_{sub2} y R_{sub2}), aunque normalmente las variaciones son mínimas ya que el área ocupada por esta pista auxiliar es reducida en comparación con el metal de la espiral.

1.3 Objetivos

La integración actual de las tecnologías es tal que permite el diseño mixto analógico/digital en el mismo chip, y va siendo cada vez más necesario integrar inductancias en los sistemas, como puede ser el caso particular de los de comunicaciones. En este contexto se plantea el objetivo de este proyecto: generar el

layout de la bobina integrada mediante el software *Cadence* para cada caso particular, con la posibilidad de cambiar la forma de la espira. Como antecedentes en este trabajo nos encontramos con el software I-MODEL [46], el cual fue desarrollado en Matlab y está basado en el modelo paramétrico elaborado a partir de valores empíricos. Este software permite que el usuario pueda conocer el valor de los elementos del circuito equivalente de un inductor conociendo sus parámetros geométricos, en los cuales se basará la creación de los nombrados layouts. Por otro lado, el I-MODEL puede obtener la geometría de la bobina que presenta mayor factor de calidad a la frecuencia de operación y con la inductancia deseada. Además de los parámetros de esta bobina, el programa genera un fichero de texto con las características de todas las bobinas que satisfacen los requisitos impuestos.

Otra tarea a abordar en el trabajo consiste en la elaboración de una herramienta de trazado de los *layouts* de las bobinas que esté integrada con la herramienta anterior. El objeto final es que las dos herramientas conformen un único programa que sea accesible mediante una página Web y que facilite de esta manera la presentación de los *layouts* de las bobinas óptimas en cada aplicación. Por tanto, se ha trabajado el software en el lenguaje Java para implementar el mismo.

1.4 Estructura de la memoria

En este apartado se explicará muy brevemente en que consiste cada uno de los capítulos que conforman este proyecto de fin de carrera.

- Capítulo 1. Introducción. Se expondrán los principios básicos de los inductores integrados, así como los objetivos que se desean alcanzar en este proyecto de fin de carrera y la exposición de la organización de la memoria.

- Capitulo 2. Inductancias integradas. En este capitulo se especificarán las características determinadas para la tecnología utilizada en cuestión S35D4 de la empresa fundidora Austria Micro Systems (AMS) para obtener los cálculos de las bobinas de los diversos supuestos satisfactoriamente.

- Capitulo 3. Entornos. La herramienta Web está desarrollada en el lenguaje de programación “Java” mediante el entorno NetBeans IDE y tras sus especificaciones y ejecución del programa se obtendrá como resultado un fichero “.cif” que a partir del entorno *Cadence* se obtendrá el *layout* especifico.

En este capítulo se presentan los conceptos generales básicos de los entornos mencionados.

- Capitulo 4. I-MODEL. La herramienta desarrollada en [46] llamada I-MODEL es la base para el desarrollo de la herramienta Web diseñada en este proyecto, observándose en este capítulo su funcionamiento y sus aplicaciones.

- Capitulo 5. Desarrollo de la herramienta Web. Dicho capitulo reúne todos los datos teóricos anteriormente explicados para la obtención de la herramienta Web encargada de la obtención de la bobina de máximo factor de calidad, el cálculo de los parámetros del modelo equivalente en PI de una bobina determinada por su geometría y por la frecuencia en la que esta trabaja, la representación grafica de un barrido en frecuencia tanto en inductancia como en factor de calidad y, finalmente, la obtención de los ficheros".cif" para el correcto trazado de las bobinas en *Cadence*.

- Capitulo 6. Conclusiones. En este apartado final se recogen las conclusiones obtenidas a lo largo de la memoria y se presenta una serie de líneas de trabajo abiertas en la continuación de este proyecto de fin de carrera.

Capítulo 2

INDUCTANCIAS INTEGRADAS

2.1 Descripción de la tecnología

La herramienta Web esta basada en la tecnología de 0.35 µm SiGe-BiCMOS (S35D4) de *Austria Micro Systems* (AMS) [47], ya que ofrece buenas prestaciones con un coste no muy elevado.

En la Figura 2.1 podemos ver un esquema del corte transversal del sustrato correspondiente a esta tecnología. Puesto que estamos tratando el diseño de bobinas nos interesan las capas metálicas. Como podemos ver, contamos con cuatro niveles de metal, de los cuales los tres inferiores tienen un grosor y una conductividad similares, y el cuarto es más grueso y más conductivo que el resto. Esto, junto con el hecho de que es el nivel superior y por tanto genera una capacidad parásita con el sustrato menor, hace que este metal sea el más adecuado para el diseño de bobinas.

La propia fundidora proporciona un *kit* de diseño que contiene, entre otras cosas, librerías de pasivos. En el caso de los inductores, sólo unos cuantos ofrecen buenas prestaciones a frecuencias por encima de los 5 GHz. Además todos ellos son cuadrados, que como fue mencionado en el capitulo anterior no es la geometría que optimiza el funcionamiento del inductor [48]. Así, cuando el diseñador necesita una bobina para un circuito determinado, puede que no encuentre ninguna apropiada entre las que ofrece el *kit*, bien porque no tienen el valor de la inductancia o factor de calidad que requiere, o bien porque este no está centrado en el rango de frecuencias que interesa.

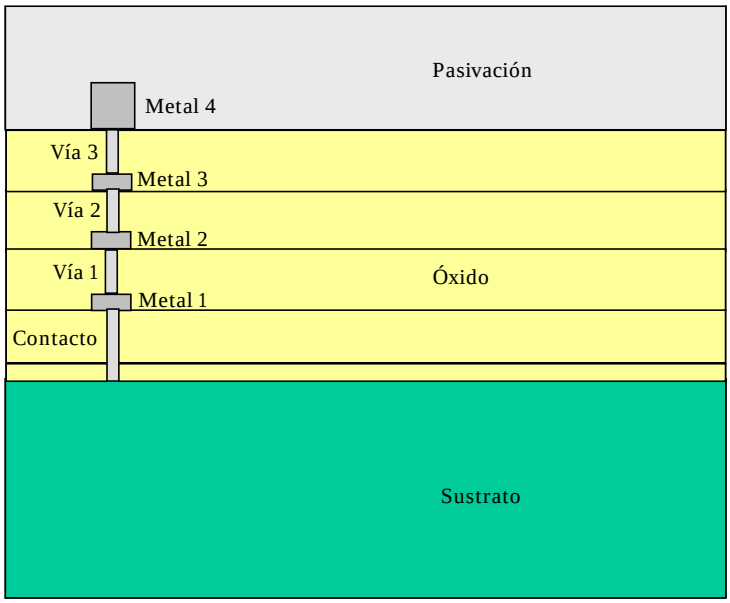

Figura2.1 Esquema del corte transversal de la tecnología AMS 0.35 µm.

Un ejemplo de esto puede verse en la Figura 2.2. En este caso necesitábamos un inductor de 2 nH para trabajar a una frecuencia de 5 GHz. Como podemos ver, el inductor diseñado por el IUMA mejora claramente las prestaciones del más apropiado de entre los que ofrece el *kit* de diseño.

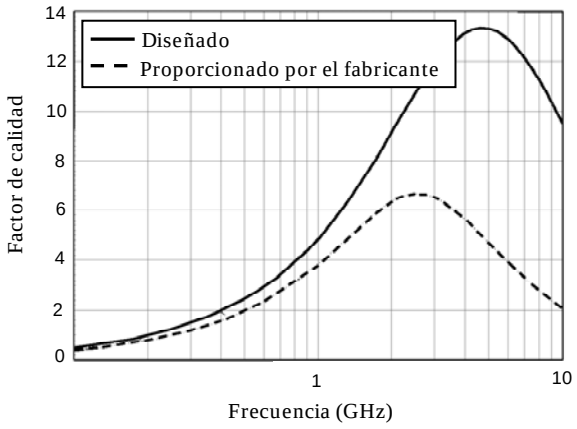

Figura2.2 Comparación del Q de dos inductores de 2 nH.

2.2 Modelo físico de un inductor integrado

Una inductancia integrada esta formada por una pista de metal que toma la forma de una espira. La espira se fabrica a partir de las capas de metal que permite la tecnología, por lo que toda ella estará rodeada de oxido de silicio. La forma de una inductancia integrada se muestra en la Figura 2.3.

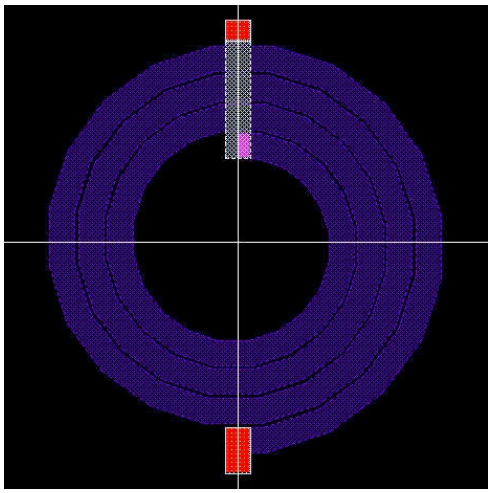

Figura 2.3. Inductancia integrada de 20 lados por vuelta.

Tal y como vimos en el capitulo anterior, el circuito equivalente de dos puertos, que considera la mayoría de los efectos, que aparecen en esta estructura en el que se muestra en la Figura 2.4 [49].

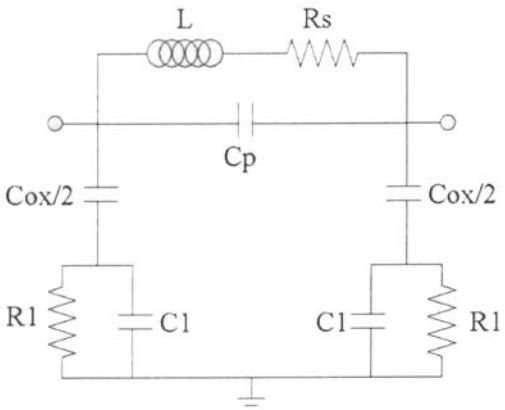

Figura 2.4 Modelo de dos puertos de un inductor en espiral integrado.

Los parámetros más importantes pueden modelarse mediante el siguiente juego de ecuaciones [50]. Los parámetros geométricos son los señalados en la figura 2.5.

- Inductancia:

$$L \approx \frac{K_e \cdot \mu_o \cdot n^2 \cdot a^2}{22 \cdot r - 14 \cdot a} \qquad (2.1)$$

Donde K_e es una constante empírica para bobinas espirales que depende de la tecnología y varía entre 28 y 32 aproximadamente; μ_o = $4 \cdot \pi \cdot 10^{-7}$ H/m es la permeabilidad del vacío; *n* es el número de vueltas de la espiral; *a* es el radio medio de la espiral y *r* es el radio máximo de la espiral.

- Resistencia serie: A bajas frecuencias es la resistencia serie de la pista de metal.

$$R_S \approx R_{DC} = \frac{l}{w \cdot \sigma \cdot t} \qquad (2.2)$$

Donde *l* es la longitud total del bobinado; *w* el ancho de la pista de metal, σ es la conductividad del metal y *t* es el espesor de la pista de metal. Este parámetro es uno de los mas importantes y para altas frecuencias habrá que añadir a la ecuación presentada nuevos efectos como las corrientes torbellino, así como la distribución no uniforme de la corriente dentro de la pista de metal, el llamado efecto pelicular, de estos ya hablaremos posteriormente.

- C_p es la capacidad entre pistas y representa la capacidad de acoplamiento entre conductores:

$$C_P = n_u \cdot w^2 \cdot \frac{\varepsilon}{t_{ox\_M3M4}} \qquad (2.3)$$

Donde n_u es el número de intersecciones entre las pistas de la espiral y el *underpass,* ε es la permitividad del material, y t_{ox_M3M4} es el grosor del óxido que hay entre el *crossunder* y la espiral.

- C_{OX} es la capacidad entre la espira y el sustrato:

$$C_{OX} = w \cdot l \cdot \frac{\varepsilon_{OX}}{t_{OX}}$$

(2.4)

Donde t_{OX} es el espesor del oxido.

Esta capacidad influye mucho en el funcionamiento global de la bobina, ya que es la de más alto valor de las tres capacidades que forman el modelo equivalente del inductor. El desplazamiento de la curva del factor de calidad en frecuencia depende en gran medida de C_{OX}. Para el puerto correspondiente al *underpass* se puede refinar la expresión (2.4) asociándole en paralelo la capacidad que añade dicha tira de metal:

$$C_{OX\_UND} = w \cdot l_{UND} \cdot \frac{\varepsilon}{t_{ox\_UND}} \tag{2.5}$$

Donde l_{UND} es la longitud del *underpass* y t_{ox_UND} la distancia entre él y el sustrato.

- Perdidas del sustrato:

$$R_{SUB} \approx \frac{2}{l \cdot w \cdot G_{SUB0}} \tag{2.6}$$

Donde;

$$G_{SUB0} \approx \frac{1}{\rho_{SUB} \cdot t_{SUB}} \tag{2.7}$$

Donde ρ_{SUB} y t_{SUB} son la resistividad y profundidad del sustrato respectivamente

- Capacidad del sustrato:

$$C_{SUB} \approx \frac{l \cdot w \cdot C_{SUB0}}{2} \tag{2.8}$$

Donde:

$$C_{SUB0} \approx \frac{\varepsilon_{SUB}}{t_{SUB}} \tag{2.9}$$

Donde ε_{SUB} y t_{SUB} son la permitividad eléctrica y profundidad del sustrato respectivamente.

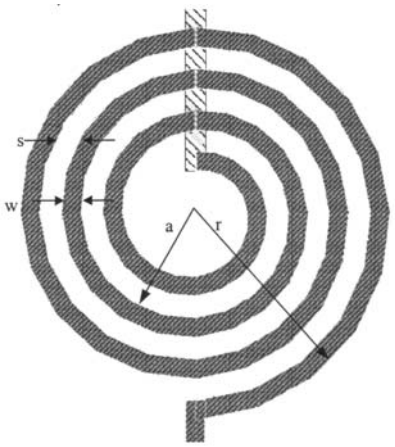

Figura 2.5 Parámetros geométricos de un inductor en espiral.

2.3 Factor de calidad

El factor de calidad *Q* nos da una medida precisa de la importancia de los elementos parásitos de una inductancia integrada. Gracias a este parámetro es posible comparar su funcionamiento con otras. En este apartado definimos el factor de calidad así como su calculo y se definen también parámetros tan importantes como la frecuencia de resonancia.

Para una inductancia integrada, la energía almacenada en el campo magnético es la que nos interesa. Cualquier energía almacenada en el campo eléctrico de la espira es debida a las inevitables capacidades parásitas y resistencias y es lo que se intenta evitar. Entonces *Q* es proporcional a la energía magnética neta almacenada, que es igual a la diferencia entre los picos de energía magnética y eléctrica. Una inductancia entra en resonancia cuando los picos de energía magnética y eléctrica son iguales. Por lo tanto el factor de calidad se hace cero a la frecuencia de resonancia [51].

El factor de calidad puede calcularse fácilmente desde el modelo simplificado de un puerto presentado en la Figura 2.6 Este modelo se ha obtenido colocando a tierra uno de los puertos [49].

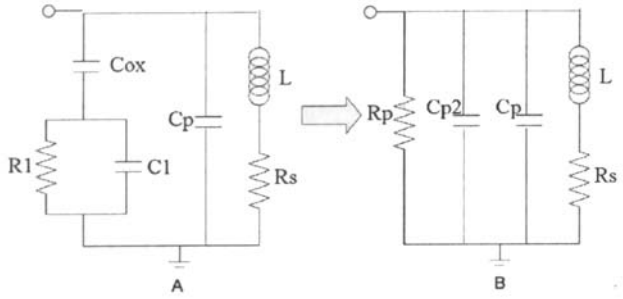

Figura 2.6 Modelo eléctrico de una espira de un puerto.

En la Figura 2.4, la impedancia formada por C_{OX}, R_1 y C_1 es sustituida por R_P y C_{P2} que son dependientes de la frecuencia. Así se facilita el análisis del efecto de R_P en la calidad de la bobina Q y también la extracción de las capacidades parásitas entre capas de metal.

Las expresiones de R_P y C_{P2} son:

$$R_P = \frac{1}{\omega^2 \cdot C_{OX}^2 \cdot R_1} + \frac{R_1 \cdot (C_{OX} + C_1)^2}{C_{OX}^2} \qquad (2.10)$$

$$C_{P2} = C_{OX} \cdot \frac{1 + \omega^2 \cdot (C_{OX} + C_1) \cdot C_1 \cdot R_1^2}{1 + \omega^2 \cdot (C_{OX} + C_1)^2 \cdot R_1^2} \qquad (2.11)$$

Donde $\omega = 2 \cdot \pi \cdot f$ es la frecuencia angular. El factor de calidad obtenido de este modelo de un puerto, aunque muy general, es:

$$Q = \frac{Energia\ almacenada}{Energia\ \text{perdida}} \qquad (2.12)$$

Si $\omega \cdot L / R_S$ representa la carga almacenada por un campo magnético y las perdidas en el metal, y consideramos f_{sub} como el factor de degradación del sustrato que representa las perdidas en el sustrato y f_{res} que seria el factor de degradación asociado a la frecuencia de resonancia que representa las perdidas debidas al acoplamiento

capacitivo producido por C_P, describe la reducción del factor de calidad debido al aumento del campo eléctrico con la frecuencia y la anulación de la Q a la frecuencia de resonancia, la ecuación del factor de calidad quedaría de la siguiente forma:

$$Q = \frac{w \cdot L}{R_S} \cdot f_{sub} \cdot f_{res} \qquad (2.13)$$

Las expresiones para los factores de degradación f_{sub} y f_{res} son:

$$f_{sub} = \frac{R_P}{R_P + \left(\left(\frac{\omega \cdot L}{R_S}\right)^2 + 1\right) \cdot R_S} \qquad (2.14)$$

$$f_{res} = \left(1 - \frac{R_S^2 \cdot (C_{P2} + C_P)}{L} - \omega^2 \cdot L \cdot (C_{P2} + C_P)\right) \qquad (2.15)$$

Ambas expresiones f_{sub} y f_{res} son normalmente mucho mas pequeñas que la unidad.

La siguiente figura representa el comportamiento de la Q y los factores de degradación con respecto a la frecuencia para una inductancia integrada estándar.

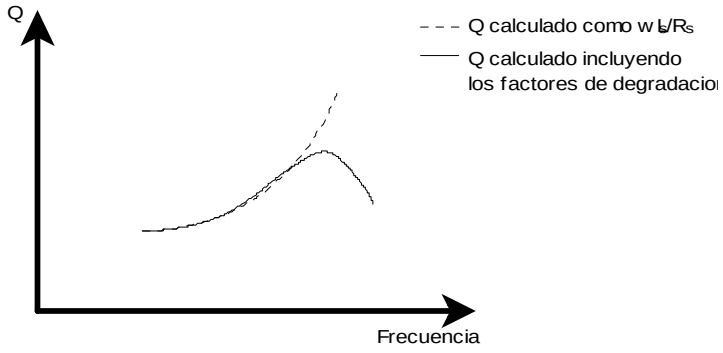

Figura 2.7. Típica variación del factor *Q* de una espira respecto a la frecuencia.

A bajas frecuencias, Q esta bien caracterizada por $\omega \cdot L_S / R_S$. Porque los dos factores de degradación tienen un factor cercano a la unidad. A medida que la

frecuencia aumenta, estos factores disminuyen su valor. La reducción del factor de calidad Q a altas frecuencias es el resultado de una combinación del factor de perdidas del sustrato y el factor de resonancia.

Debido a que a bajas frecuencias los factores de degradación toman valores en torno a la unidad, se puede hacer la aproximación de que a frecuencias menores a 1/5 de la frecuencia de resonancia *($f < 1/5\, f_{res}$)*, la Q esta bien caracterizada por $\omega \cdot L_S / R_S$.

2.4 Frecuencia de resonancia

La frecuencia de resonancia es la frecuencia a la cual el termino de impedancia del inductor integrado presenta una parte imaginaria negativa, aquí el comportamiento inductivo de la bobina se convierte en capacitivo. Nunca debe llegarse a esta frecuencia en una operación normal.

Mientras que un inductor ideal tiene una inductancia constante para todo rango de frecuencias, un inductor integrado real tiene un valor de inductancia muy parecido a la función con respecto a la frecuencia de la siguiente figura [52].

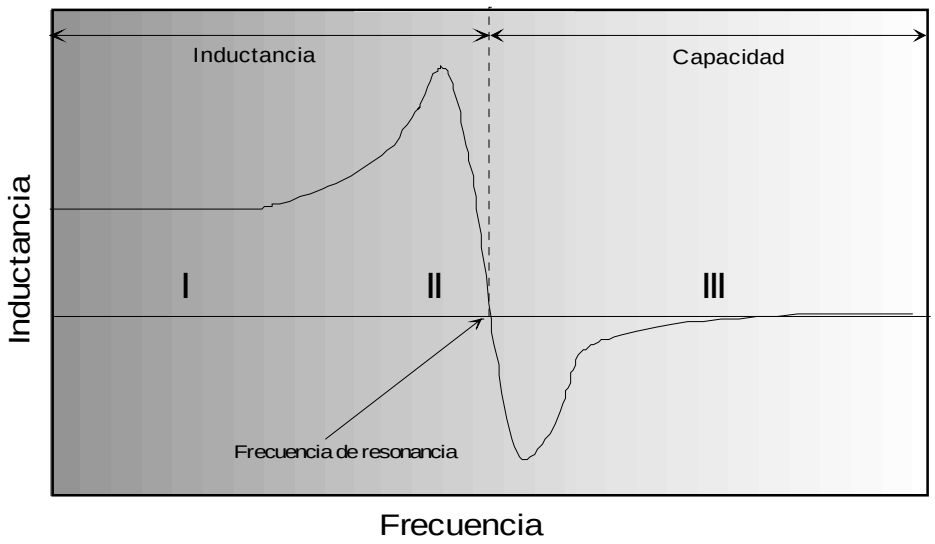

Figura 2.8 Distintas regiones de operación de un inductor integrado típico sobre un sustrato de baja resistividad.

En términos de valores de inductancia, la figura anterior muestra tres regiones distintas de operación que la mayoría de inductores integrados sobre sustrato de silicio de baja resistividad representan normalmente.

La región I comprende la banda de uso de un inductor integrado. Dentro de esta, el valor de la inductancia se mantiene relativamente constante y el elemento pasivo puede ser usado normalmente.

La región II es la zona de transición en la cual el valor de la inductancia se vuelve negativo con un paso por cero, esta es la primera frecuencia de resonancia del inductor. Por encima de este punto de frecuencia critico, el elemento pasivo trabaja como una capacidad, esta zona de trabajo debe ser evitada.

En la región III, el elemento integrado tiene un comportamiento capacitivo y al final de todo el factor de calidad se vuelve cero, haciendo inviable su uso para esas frecuencias.

Igualando la ecuación anterior de f_{res} [52] a cero obtenemos el valor de la frecuencia de resonancia de la inductancia.

$$f_0 = \frac{1}{2 \cdot \pi} \cdot \sqrt{\frac{1}{L \cdot \left(C_{P2} + C_P\right)} - \frac{R_S^2}{L^2}} \qquad (2.16)$$

Esta es la relación entre la frecuencia para una máxima Q y f_0 . Si f_0 aumenta, la frecuencia para el máximo factor de calidad también aumenta.

2.5 Efecto peculiar y corrientes torbellino

A bajas frecuencias, la resistencia del metal puede ser calculada fácilmente como el producto de la capa resistiva y el número de cuadros de la pista de metal. Sin embargo cuando la frecuencia se incrementa, el efecto pelicular y las corrientes torbellino inducidas, pueden causar una gran desviación respecto a las predicciones obtenidas por el modelo de ecuación anteriormente presentado (2.2). R_S debe estimarse

de otra forma para conseguir un correcto modelado del factor de calidad. De todas formas como se dice en [49], el efecto pelicular puede ser despreciado si el grosor del metal es suficientemente fino, la resistencia serie se incrementa bruscamente un 8% cuando el grosor del metal es de 0.5 μm, un 20% cuando es 0.75μm y un 40% para 1μm, todas medidas a 5GHz de frecuencia.

Siguiendo con el calculo de la nueva R_S para altas frecuencias y según [49], comprobamos que $E_1(t)$ produce una densidad de corriente de conducción dada por $J=\sigma \cdot E_1(t)$. Esta distribución de densidad de corriente puede ser resuelta aplicando las ecuaciones de Maxwell para una longitud de vaciado de conductor (sección rectangular de área $w \cdot t$). El problema es bidimensional y la solución se representa expresada en su fasor según la siguiente expresión:

$$J(x,y) = J_x(x) \cdot J_y(y) \tag{2.17}$$

Donde:

$$J_\alpha = a_\alpha \cdot e^{\gamma \cdot \alpha} + b_\alpha \cdot e^{-\gamma \cdot \alpha} \tag{2.18}$$

γ viene dado por:

$$\gamma = \frac{1+j}{\delta} \tag{2.19}$$

Donde δ es la profundidad pelicular, definida como:

$$\delta = \frac{2}{\sqrt{\mu \cdot \sigma \cdot \omega}} \tag{2.20}$$

En la ecuación (2.17) podemos observar como la corriente es empujada a los bordes del conductor cuando la frecuencia aumenta, el llamado efecto pelicular.

Ahora la resistencia serie puede calcularse como:

$$R_S = \mathrm{Re}\left\{\frac{\int_0^l \frac{J}{\sigma}\cdot dz}{\int_S J\cdot ds}\right\} \qquad (2.21)$$

El numerador dentro de los corchetes corresponde a la caída de tensión a través del conductor, el campo $\frac{J}{\sigma}$ debe ser evaluado en la superficie, l es la longitud del vaciado del metal, z es la coordenada longitudinal y S es el trozo de área seccionada. Resolviendo la ecuación anterior llegamos a una nueva expresión para R_S.

$$\frac{R_S}{l} = \frac{1}{2\cdot\sigma\cdot\delta^2}\cdot\frac{\left(sh\frac{w}{\delta}\cdot\sin\frac{t}{\delta}+sh\frac{t}{\delta}\cdot\sin\frac{w}{\delta}\right)}{\left(ch\frac{t}{\delta}-\cos\frac{t}{\delta}\right)\cdot\left(ch\frac{w}{\delta}-\cos\frac{w}{\delta}\right)} \qquad (3.21)$$

Se puede apreciar claramente en esta ecuación que nos aproximamos a la ecuación (2.2) si $\frac{t}{\delta} << 1$, esta es la denominada condición frágil del efecto pelicular.

Aparte del efecto pelicular, el campo magnético producido en la espira induce corrientes torbellino en el sustrato y en el conductor, especialmente en las vueltas internas. Estas corrientes inducidas son pequeños rizos de corriente que fluyen oponiéndose al campo magnético original, haciendo que la corriente no se distribuya uniformemente por la capa de metal y sumándose su efecto al pelicular.

Después de todo este estudio teórico desarrollado cabe decir como conclusión que estas ecuaciones no son del todo correctas y que hoy día todavía no existe un modelo matemático que nos describa con exactitud el comportamiento de una inductancia integrada. Lo que se hace actualmente es diseñar previamente la bobina, medirla y a partir de ahí ajustar los parámetros necesarios para su posterior fabricación masiva.

Con el programa que voy a presentar en este proyecto se facilitará en gran manera el diseño de estas bobinas, proceso en el que se tardaban horas, y una vez establecido un análisis matemático correcto, nuestro programa es el primer paso para ser capaz de dibujar la inductancia más cercana a unos valores dados por el diseñador sin tener que preocuparse de medidas físicas, las cuales serán calculadas directamente para alcanzar el mejor factor de calidad para un valor de inductancia dado.

Capítulo 3

ENTORNOS

3.1 Cadence

Cadence es un software de diseño electrónico utilizado mundialmente que desempeña un papel esencial en la creación de circuitos integrados. Este software está a disposición del Instituto Universitario de Microelectrónica Aplicada (IUMA) de la Universidad de Las Palmas de Gran Canaria y las aplicaciones necesarias se desarrollan en estaciones de trabajo *Solaris* de *Sun Microsistems*.

3.1.1 El arranque del programa CADENCE

Trabajamos con la tecnología 0.35μm BI-CMOS de cuatro metales de la empresa fundidora Austria Micro Systems (AMS)**,** aunque el programa no este orientado a ninguna tecnología en concreto, pudiendo ser usada, siguiendo diversas consideraciones por cualquier tecnología. Para el arranque teclearemos en la consola, para nuestra tecnología;

ams_cds –mode fb

Si se desea usar una estación remota debemos teclear por pantalla anteriormente;

ssh time

setenv DISPLAY lcwsx:0

Siendo x el número de la estación donde nos encontramos.

3.1.2 La ventana de comandos y el library manager

Una vez dentro del programa podemos distinguir dos ventanas, la principal, (figura 3.1, **Command Interpreter Window (CIW)**) y la denominada **library Manager** (figura 3.2) donde tenemos la información sobre las librerías disponibles.

Figura 3.1. Ventana principal.

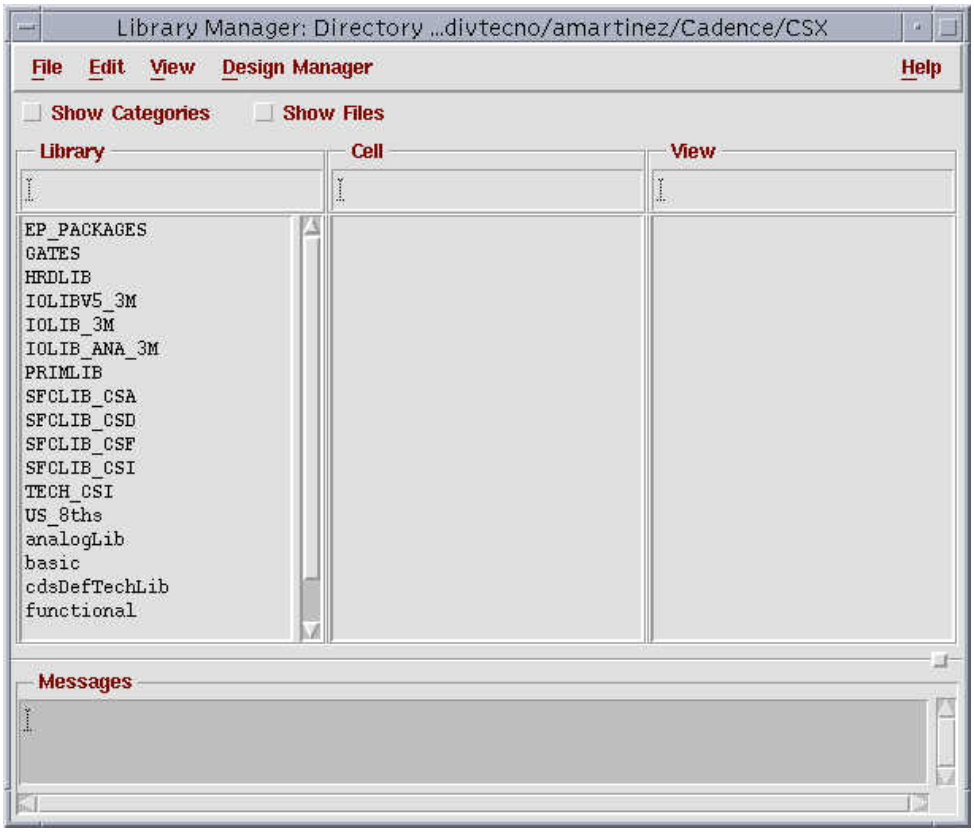

Figura 3.2. Ventana de las librerías *Cadence*.

3.1.3 Creación de una librería

Comencemos por el principio, hemos de crear en primer lugar una librería donde guardaremos el diseño. Esto podrá hacerse de dos maneras, si la librería no existe, la crearemos en el **Library Manager File → New → Library** abriéndose una ventana donde escribiremos el nombre de la nueva librería, en nuestro caso bobinas (figura3.3) y asociándolo a un fichero tecnológico en nuestro caso S35D4 de la empresa fundidora Austria Micro Systems (AMS) (figura 3.4 y figura 3.5).

Figura 3.3.

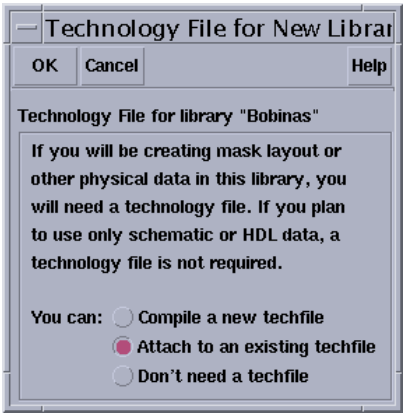

Figura 3.4.

Figura3.5.

Si por el contrario ya tenemos creada la librería, solo habrá que añadirla, en el menú **tools** de la ventana principal elegiremos **library path editor** (figura 3.6) y dentro de esta en **edit** añadimos el nombre de la librería Bobinas, tener en cuenta que aunque no tengamos una librería creada de antemano al utilizar este sistema nos la creará, pero al no tenerla asociada a ninguna tecnología, nos dará errores posteriormente.

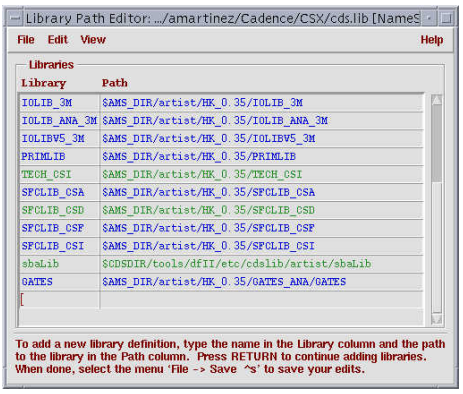

Figura 3.6

Salvando la nueva configuración veremos que se nos a creado una nueva librería en el **Library Manager**.

3.1.4 Creación de una celda

Eligiendo Bobinas y luego en el menú **File** → **New** → **Cell View** abrimos una nueva ventana donde crearemos una celda llamada "generica" con **tool** → **Virtuoso** (figura 3.7).

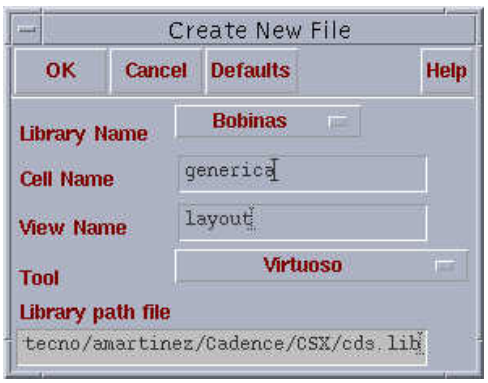

Figura 3.7.

Abriéndose una nueva ventana para el dibujo del layout y creándose en el **Library Manager** el nombre de la celda y la vista, genérica y layout respectivamente (figura 3.8).

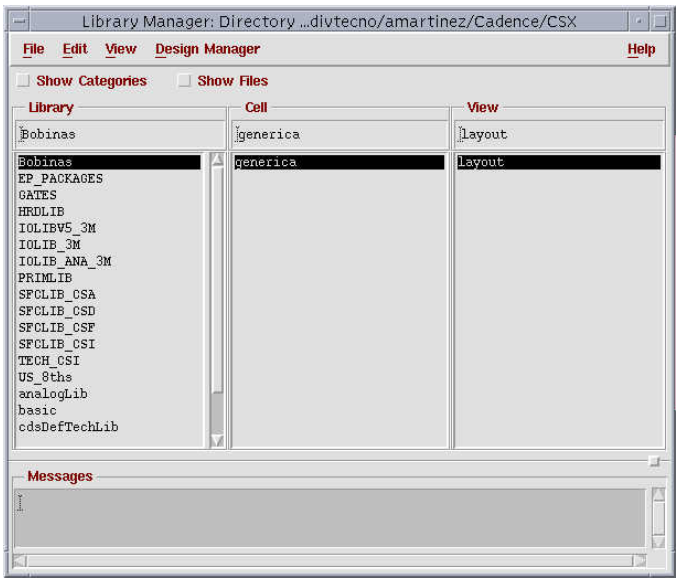

Figura 3.8.

El editor de layouts Virtuoso es una herramienta que usamos para preparar el diseño de circuitos integrados a medida. El editor del layout es el editor base en el juego de herramientas Virtuoso.

El editor de layouts es una herramienta de diseño gráfico que nos permite realizar las siguientes tareas:

- Crear y editar polígonos, pistas, rectángulos, círculos, elipses, pines y contactos en vistas layout (*layout cellviews*).

- Colocar celdas en el interior de otras celdas para la creación de diseños jerárquicos.

- Conectar un pín o un grupo de pines a una red interna o externa.

- Automatizar cada etapa de la tarea del diseño utilizando el *Virtuoso layout accelerator* (Virtuoso XL).

- Crear celdas parametrizables, denominadas Pcells conteniendo datos que podrán ser modificados rápidamente o a los que se podrán acceder a través de comandos SKILL.

3.1.5 Formato CIF

El formato CIF es uno de los diversos formatos de imagen digital. Este estandariza la resolución, tanto vertical como horizontal de los píxeles de las imágenes de imagen digital.

En el formato CIF las líneas se terminan con coma. Los comentarios en el CIF se encierran entre paréntesis. Los comentarios son ignorados en el formato CIF.

(This is an example comment);

Todas las líneas de formato CIF deben concluirse con `;'.

El primer o segundo carácter no espacio en blanco de una línea (es decir, después de `; ') se utilizan como comando.

El DS (definir símbolo) La directiva comienza con un símbolo definido.

DS symnum A B;

Los símbolos no tienen nombres, pero se hace referencia por un número. El símbolo asignado a un número (un entero) sigue tras DS. Los otros dos parámetros son para la ampliación. Cada uno coordina en la escala del símbolo A B. Se hace uso de dos enteros en lugar de un único valor de punto flotante:

DS 0 1 1;

La definición de un símbolo se termina con una línea DF,

DF;

En el símbolo definido hay una capa de directivas seguida de las especificaciones geométricas y una subcelda de llamadas. Una capa directiva se compone de una línea con la forma

L layername;

Donde “layername” es un nombre para una capa. Este nombre es alfanumérico y es de cuatro caracteres de longitud o menos. Toda la geometría que sigue la capa de declaración será asignada a esa capa hasta la próxima capa de declaración.

Hay tres tipos de objetos geométricos en el formato CIF: cajas, polígonos, y cables. Las cajas tienen la forma

B anchura x altura y [RX RY];

Donde los dos primeros parámetros son la anchura y la altura de la caja, y los dos siguientes parámetros son las coordenadas del punto medio de la caja. Los dos últimos parámetros son opcionales e indican una rotación. Estos dos números definen un vector con respecto al origen y el ángulo por el que la caja debe ser rotada.

Los polígonos se especifican con P seguida de pares de coordenadas xy. La primera y la última de las coordenadas deben ser la misma, con indicación de cierre del polígono.

P x0 y0 x1 y1 ... xN yN x0 y0;

El polígono se hace mediante los atributos de la capa sobre la que el polígono está definido. Debe haber al menos cuatro pares de coordenadas definidas para un polígono.

Los cables son de ancho fijo. Un alambre se especifica con W seguida de la anchura, que es seguida por pares de coordenadas xy que representan la ruta de acceso.

W ancho x0 y0 x1 y1 ... xN yN;

En el modelo eléctrico la línea básica es un alambre de ancho cero. En el modo físico los cables se definen con ancho finito como una necesidad física, las coordenadas forman los vértices de la ruta. Un cable técnicamente puede constar de un solo vértice como una caja con la anchura del alambre.

El símbolo de llamada a la subcelda se indica con una C, seguido por un número y a su vez seguido por una transformación del despliegue de condiciones. La transformación de los componentes que representan la traducción, rotación y reflexión. La traducción es indicado por T seguida de los valores:

T x y

La rotación es especificada por R, seguida de dos números que representan un vector con respecto al origen y del ángulo de rotación.

R rx ry

El reflejo sobre el eje y se especifica con

MX

y sobre el eje "x" con

MI

La transformación es una especificación de la concatenación de estas directivas, que se evalúan en secuencia para obtener la coordinación de la cartografía de las coordenadas de la celda, en el símbolo de instancia de la celda, coordenadas de la matriz de la instancia. La sintaxis general es

C symnum transformar;

donde un ejemplo de ello sería

C 0 R 1 0 T -1000 0;

La línea de ejecución se termina con un final de directiva:

END

Esta línea no tiene que ser finalizada con un punto y coma.

La base del sistema de coordenadas especificado para el formato CIF esta formada por 100 unidades por micras.

3.1.6 Importar un archivo CIF

En primer lugar arrancamos *Cadence*. Para ello seguimos los pasos anteriormente explicados. A continuación nos vamos a la ventana principal "**Icfb**" , donde hacemos clic en la pestaña "**File**".

Figura 3.9.

Tras hacer clic en esta opción se despliega un menú de opciones. Hacemos clic sobre "**Import**" y a su vez en "**CIF**" en el menú que se despliega a continuación.

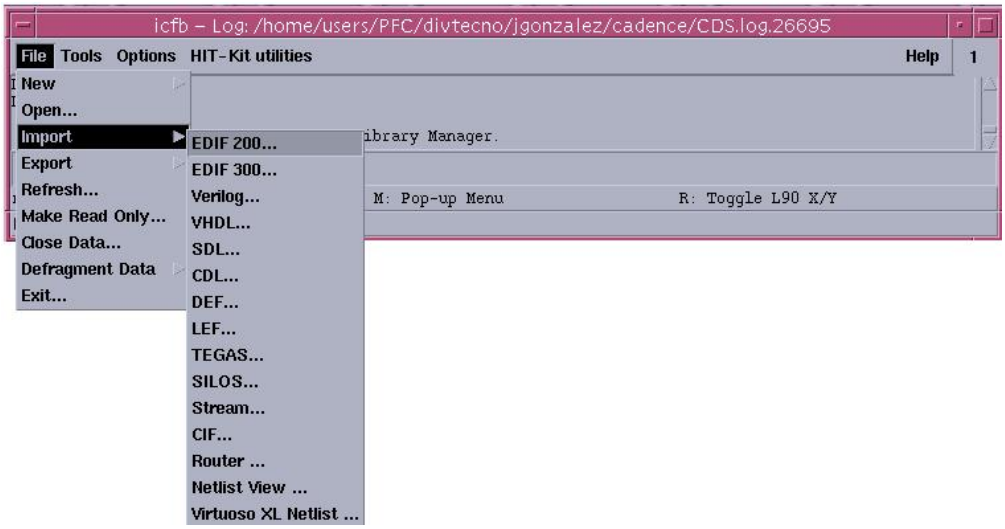

Figura 3.10.

Finalmente se desplegará una ventana donde debemos introducir el nombre del archivo CIF y la librería en la que se encuentra, correspondiendo a las casillas "**Imput File**" y "**Library Name**". Este archivo CIF debe ser correctamente creado con anterioridad.

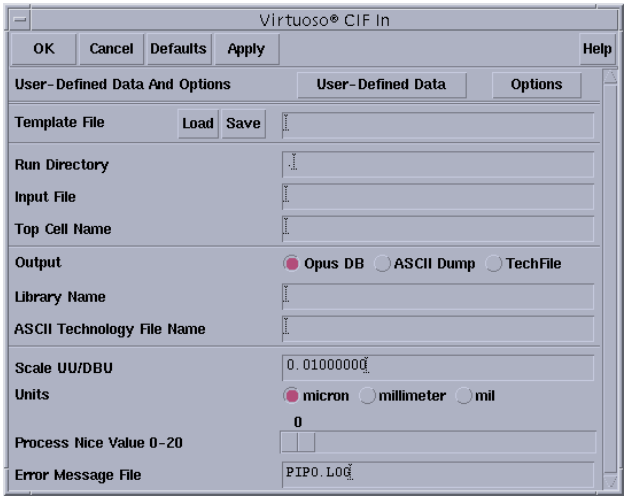

Figura 3.11.

Finalizado este proceso se podrá cargar el layout de la bobina importada desde la librería especificada (Figura 3.8).

3.2 NetBeans IDE

NetBeans IDE es un entorno de desarrollo visual de código abierto para aplicaciones programadas mediante Java, uno de los lenguajes de programación más usado en la actualidad. Su aprendizaje se ha convertido en fundamental para quienes están interesados en el desarrollo de aplicaciones multiplataforma. NetBeans nos permite escribir, compilar, depurar y ejecutar programas en Java. Está escrito en Java pero su utilización es posible con cualquier otro lenguaje de programación.

No importa que el entorno donde se instale sea Linux, Mac o Windows, pues el funcionamiento del programa java creado será el mismo.

Mediante NetBeans es posible diseñar aplicaciones con solo arrastrar y soltar objetos sobre la interfaz de un formulario. Si se está familiarizado con entornos como .NET, aquí también resultan fácilmente accesibles los componentes similares como son los JLabels, JButtons o JTextFields.

Con NetBeans IDE no solo es posible elaborar potentes aplicaciones para el entorno de Escritorio, también para la Web y para dispositivos portátiles, como móviles o Pocket PC, sin que sea necesario cambiar la forma de programar.

La programación mediante NetBeans se realiza a través de componentes software modulares, también llamados módulos. NetBeans pone a disposición de los usuarios decenas de módulos a través de su página web, que pueden ser integrados en él para conseguir mejores aplicaciones. Un módulo es un archivo Java que contiene clases java escritas para interactuar con las APIs de NetBeans y un archivo especial (manifest file) que lo identifica como módulo. Las aplicaciones construidas a partir de módulos pueden ser extendidas agregándole nuevos módulos. Debido a que los módulos pueden ser desarrollados independientemente, las aplicaciones basadas en la plataforma NetBeans pueden ser extendidas fácilmente por otros desarrolladores de software.

3.2.1 Creación de un proyecto en Netbeans

Para la creación de un proyecto en NetBeans en el IDE se selecciona la opción **File > New Project**, como se ilustra a continuación en la Figura 3.12.

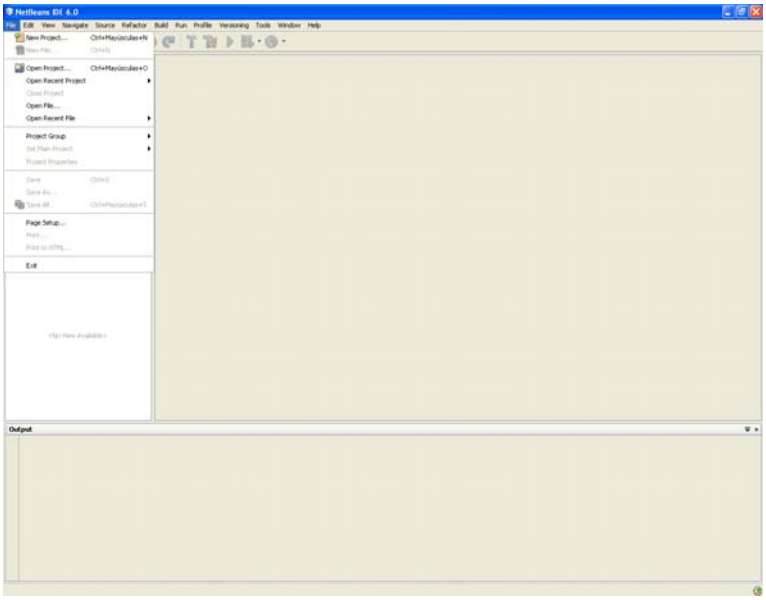

Figura 3.12.

Al seleccionar **New Project** se despliega la siguiente ventana.

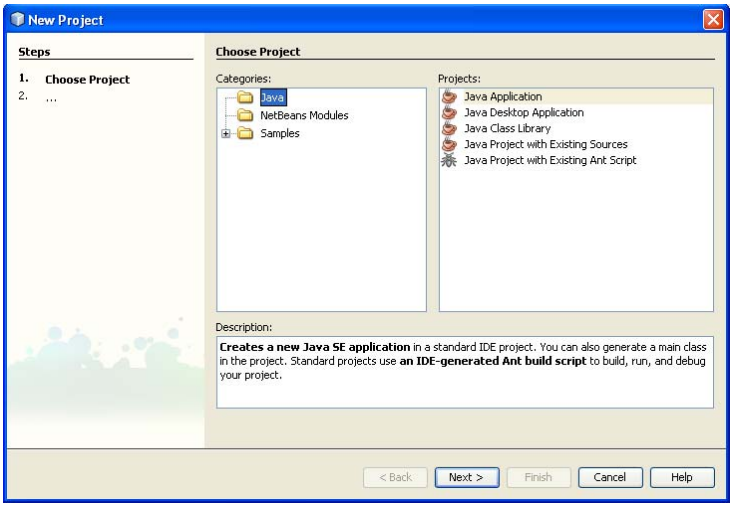

Figura 3.13.

En este momento se debe seleccionar que tipo de proyecto que se desea generar. En nuestro caso se selecciona **General** > **Java Aplication** (Figura 3.14.).

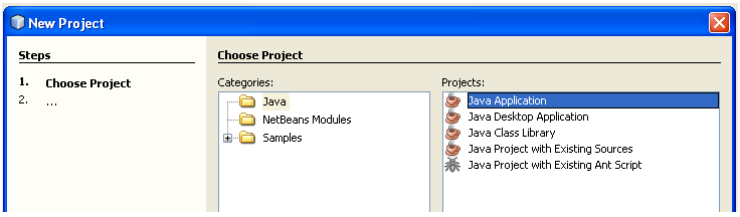

Figura 3.14.

Hacemos clic sobre la opción **Next** y se desplegará la siguiente ventana mostrada en la Figura 3.15.

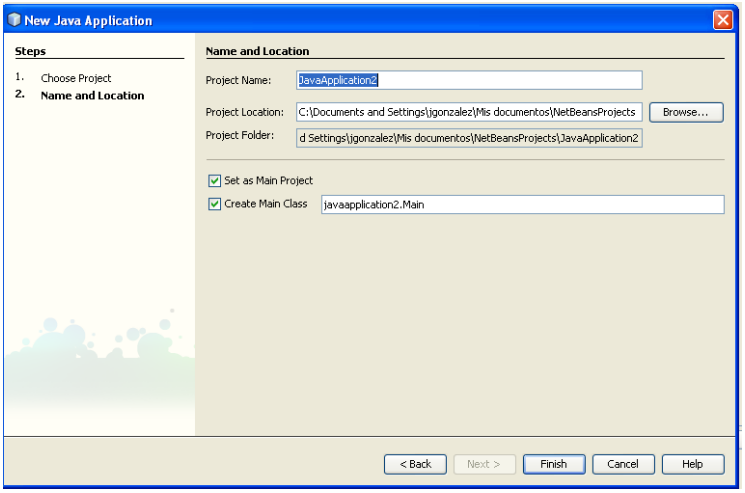

Figura 3.15.

En esta ventana se escribe el nombre del proyecto (en el campo **Project Name:**) y el nombre de la carpeta donde lo guardará (en el campo **Project Location:**). En este ejemplo, el proyecto se denominó JavaApplication2 el cual se albergará en la carpeta C:\Documents and Settings\jgonzalez\Mis documentos\NetBeansProjects. Para esta aplicación las opciones **Set as Main Project** y **Create Main Class** estarán seleccionadas (en los checkbox). En el campo **Create Main Class** aparece el texto javaapplication2; debemos completarlo así: javaapplication2.Main. Esto implicará que la clase que posee el método *main* se denomina Main, esta será la clase principal que se ejecutará en primer lugar.

Seleccionando **Finish** el proyecto habrá sido creado y abierto en el IDE (Figura 3.16.). En este momento pueden observar los siguientes componentes:

- La ventana del proyecto (**Projects window**), la cual contiene un árbol con los componentes del proyecto, incluyendo los archivos fuente y las librerías de las cuales depende el código.
- La ventana donde aparece el código fuente desplegado (**Source Editor window**). En este caso aparece el código de la clase JavaAppication2.java.
- El panel de navegación (**Navigator**), el cual puede usarse para navegar dentro de los elementos de la clase.

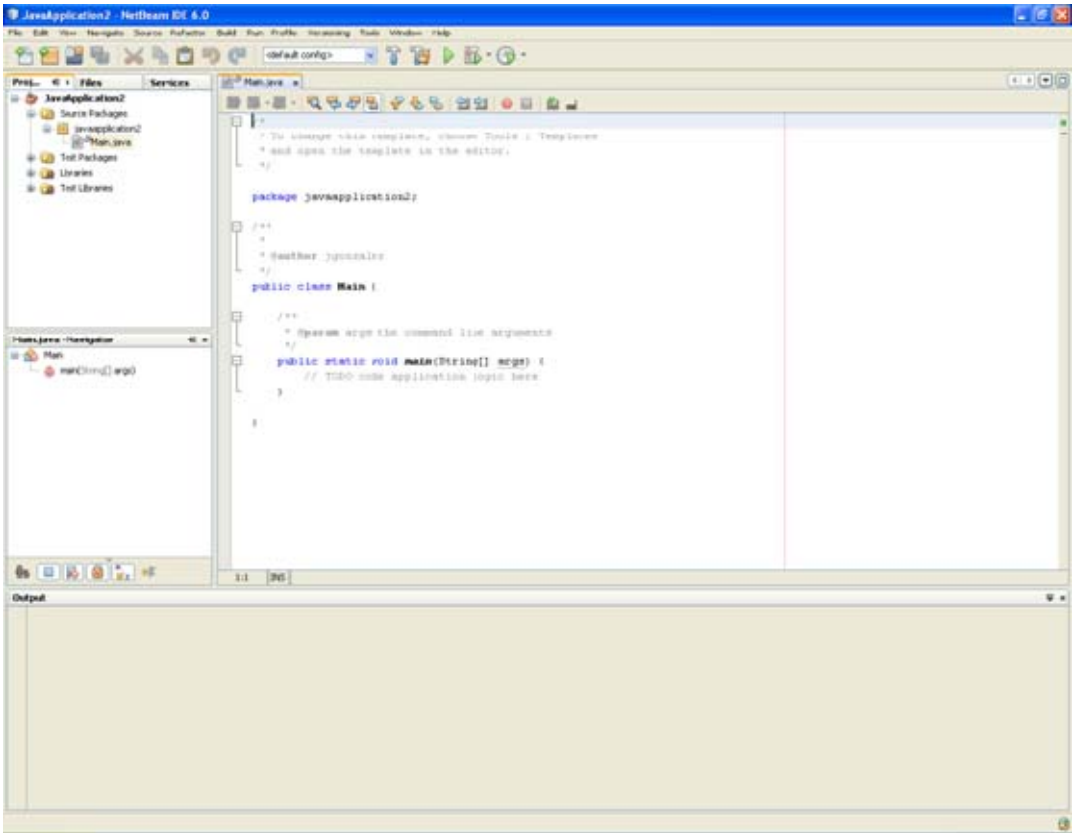

Figura 3.16.

3.2.2 Adición de Código al archivo fuente generado

Como se seleccionó el checkbox **Create Main Class,** esto da lugar a la generación del código esqueleto para la clase JavaApplication2. A continuación se completará el código para esta clase añadiendo todas las sentencias, variables y métodos necesarios.

Finalmente solo resta grabar los cambios realizados escogiendo **File** > **Save.**

3.2.3 Compilación el archivo fuente

Escoger **Build** > **Build Main Project** del menú principal del IDE. En la ventana de salida (**Output**) se presentará los mensajes que la compilación del código mande a consola, algo similar a lo que se muestra en la Figura 3.17.

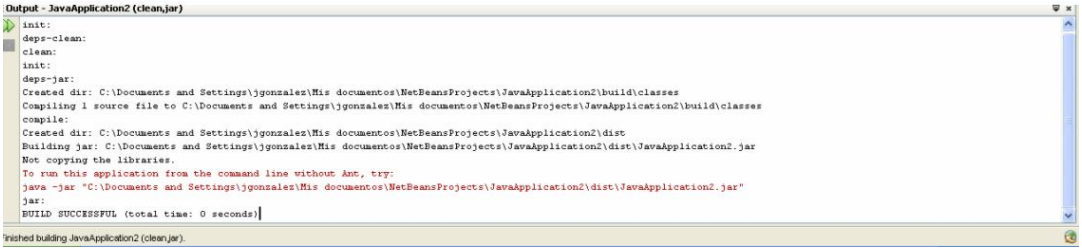

Figura 3.17.

Si no se han cometido errores la última línea desplegada en la ventana **Output** será la sentencia BUILD SUCCESSFUL. Si se han encontrado errores aparecerá en esta línea la sentencia BUILD FAILED. En este último caso se debe proceder a corregir los errores y de nuevo se debe compilar.

Si se selecciona la ventana **Files** se observará la distribución de archivos generados por el IDE, tal y como aparecen en la carpeta donde está guardando el proyecto.

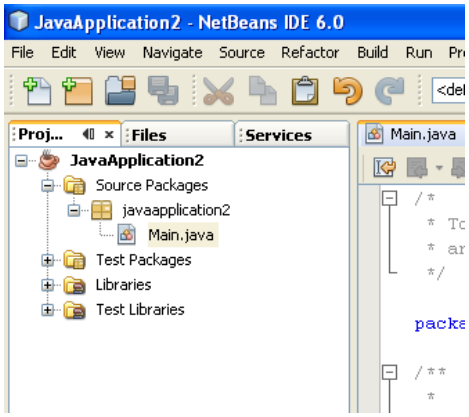

Figura 3.18.

3.2.4 Ejecución del programa

Escoger en el menú principal, **Run** > **Run Main Project**. Las salidas asociadas a la ejecución se desplegará también en la consola (ventana de salida) (Figura3.19.).

Figura 3.19.

Capítulo 4

I-MODEL

1.1 Introducción

A partir de las funciones descritas en el capítulo 2, se desarrolló una herramienta de gran utilidad para el diseñador de circuitos de RF: el software I-MODEL. Este software fue desarrollado en [46]. Estaba implementado en MATLAB y era válido para la tecnología SiGe 0.35 μm de AMS.

1.2 Software I-MODEL

El I-MODEL generado está basado en el modelo paramétrico desarrollado en el capítulo 2. La pantalla principal de la interfaz gráfica se muestra en la Figura 4.1. A través de la misma, el usuario puede conocer el valor de los elementos del circuito equivalente de un inductor conociendo sus parámetros geométricos. Por otro lado, se puede obtener la geometría de la bobina con mayor factor de calidad a la frecuencia de operación y con la inductancia deseada. Además de los parámetros de esta bobina, el programa genera un fichero de texto con las características de todas las bobinas que satisfacen los requisitos impuestos.

Con la opción "Número metales" es posible elegir si el modelo paramétrico debe estar basado en inductores fabricados en un solo metal o con dos de ellos en paralelo.

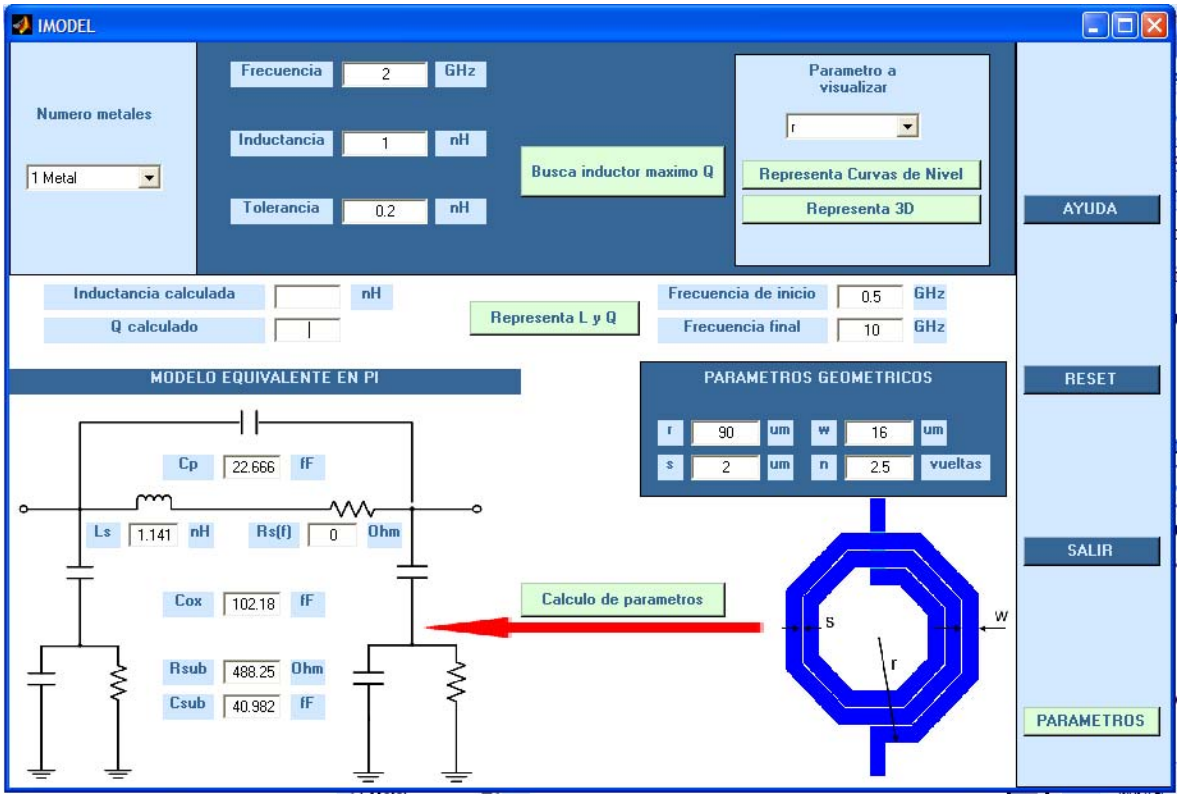

Figura 4.1 Pantalla principal del programa I-MODEL.

Este programa es escalable, y permite definir los parámetros tecnológicos sobre los que se plantea el diseño del inductor. Así, a través del botón "Parámetros" se accede a otra pantalla (ver Figura 4.2) donde es posible modificar los parámetros de la tecnología. Dichos parámetros son:

- Grosor del metal superior de la espira.
- Grosor del metal inferior, que puede actuar sólo como *underpass* o estar en paralelo con el metal superior en toda la espira.
- Distancia entre ambos metales.
- Distancia del metal/es de la espira al sustrato semiconductor.
- Grosor del sustrato semiconductor.
- Resistividad del metal superior de la espira.
- Resistividad del metal inferior de la espira.
- Constante empírica del modelo paramétrico K_e .

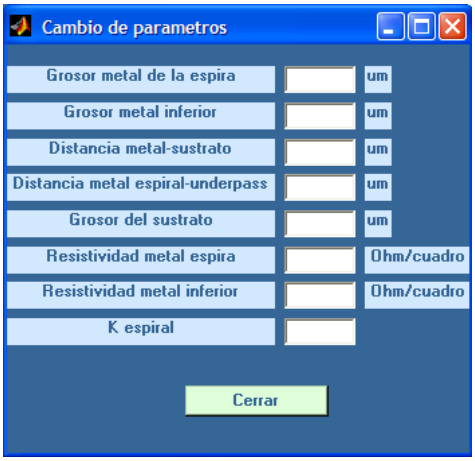

Figura 4.2 Pantalla que permite cambiar los parámetros de la tecnología y del modelo paramétrico.

El I-MODEL permite obtener las curvas de nivel o una visualización en 3D de los parámetros obtenidos a partir de la inductancia y frecuencia introducidos en la interfaz gráfica. Dichos parámetros son:

- Factor de calidad *Q*, en función del número de vueltas y el ancho de pista.
- Inductancia *L*, en función del número de vueltas y el ancho de pista.
- Radio exterior *r*, en función del número de vueltas y el ancho de pista.
- Número de vueltas *n*, en función del radio exterior y el ancho de pista.
- Ancho de pista w, en función del número de vueltas y el radio exterior.

En la Figura 4.3 muestra de los gráficos que proporciona la herramienta para el caso de necesitar un inductor de 3 nH para trabajar a una frecuencia de 3 GHz. De la lista de inductores de 3 nH que el programa genera tras realizar el barrido, se pueden obtener, por ejemplo, las curvas de nivel y la representación tridimensional de cómo varía el factor de calidad en función del número de vueltas y el ancho de pista.

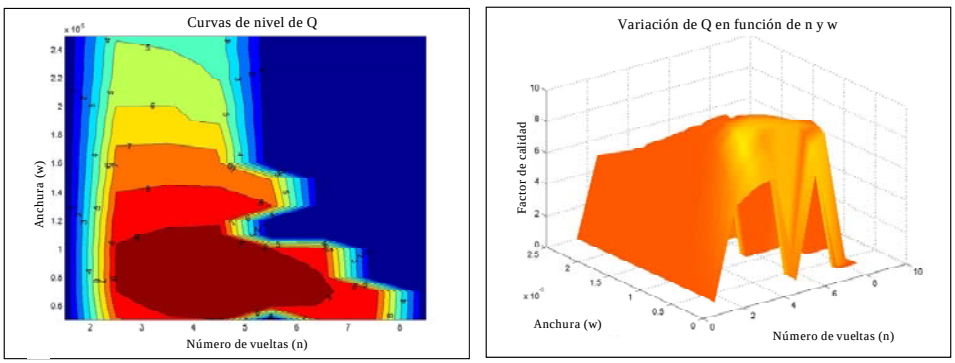

Figura 4.3 Ejemplos de los gráficos 2D y 3D que se pueden obtener con la herramienta I-MODEL.

Se puede obtener también una representación gráfica del factor de calidad y de la inductancia de la bobina cuyos parámetros estén especificados en la interfaz. La representación se hace en función de la frecuencia mínima y máxima dada por el usuario. En dicha gráfica se indica el valor máximo del factor de calidad y la frecuencia a la que se alcanza. Estos valores, junto con los obtenidos a la frecuencia deseada, son mostrados a través del cuadro de comandos de MATLAB. En la Figura 4.4 se muestra el ejemplo para un inductor de r=100 µm, n=2.5 vueltas, w=10 µm y s=2 µm en el barrido de frecuencias comprendido entre 0.5 y 20 GHz.

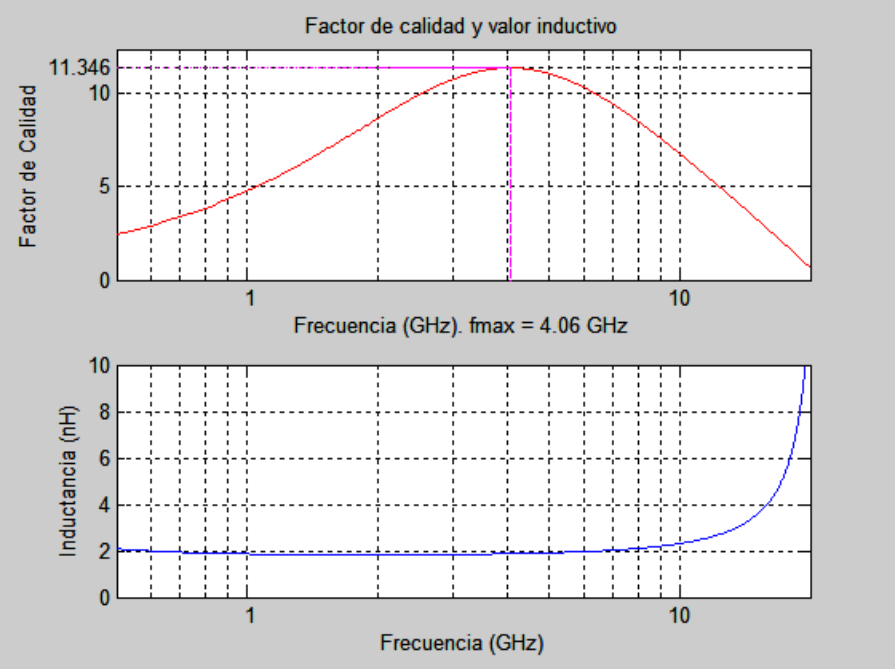

Figura 4.4 Gráfica ejemplo del factor de calidad e inductancia de una bobina para el barrido de frecuencias de 0.5 a 20 GHz.

Por último, utilizando el botón de "Ayuda" de la interfaz se puede acceder a una guía básica sobre el funcionamiento del programa. Esta ayuda ha sido realizada en código html y se integra dentro de la interfaz propia del programa MATLAB. Con esta guía se puede navegar a través de las distintas pantallas conociendo el uso de cada uno de los componentes del programa, así como los posibles mensajes de error que puedan surgir. En la Figura 4.5 se muestra la pantalla principal de esta ayuda.

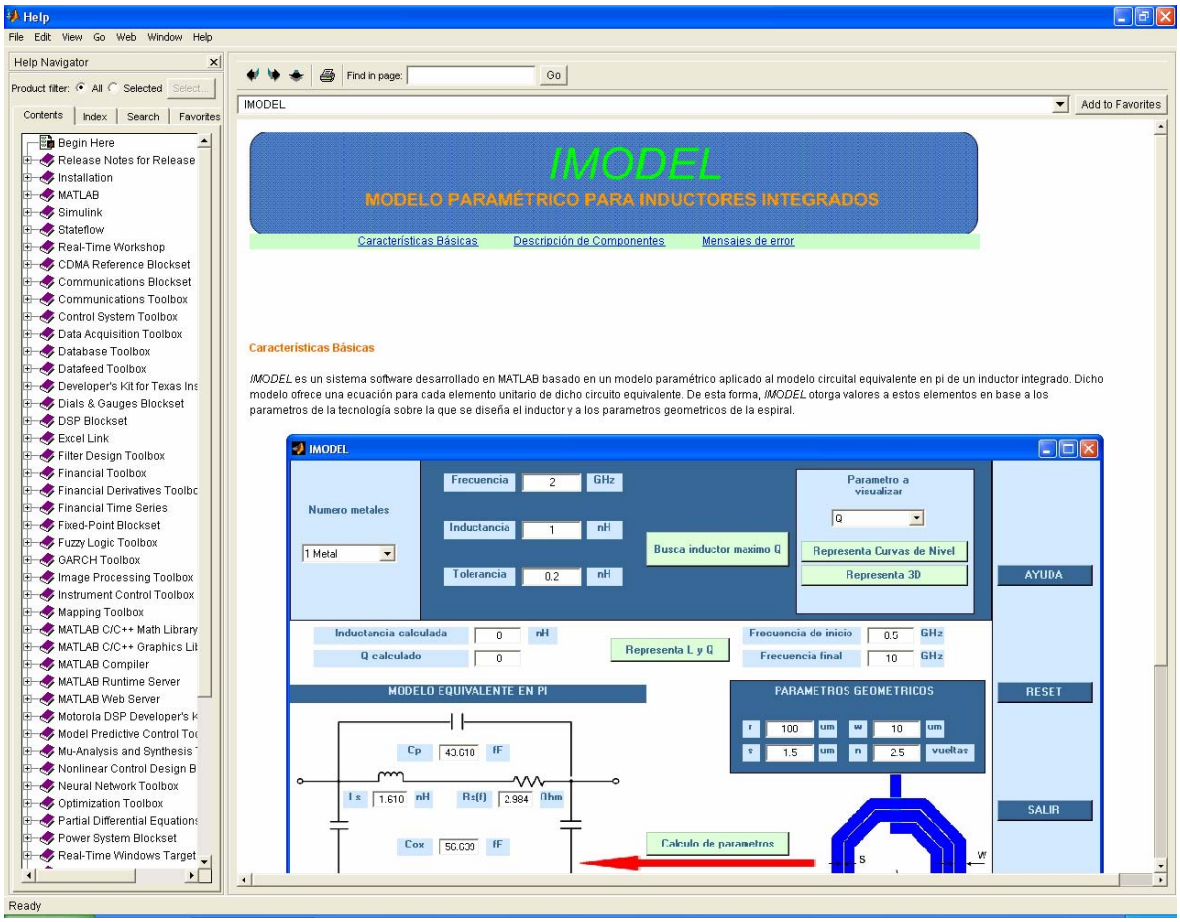

Figura 5.5 Pantalla principal de ayuda del programa I-MODEL.

Capítulo 5

DESARROLLO DE LA HERRAMIENTA SOFTWARE

5.1 Introducción

En este capítulo se muestra cómo se ha realizado la herramienta Software, desglosando por separado los bloques que la conforman, observándose que la finalidad de este proyecto es la generación de *layouts* de bobinas integradas para las necesidades especificas de cada supuesto. Para ello se ha basado en el software I-MODEL que forma parte en [5], que ha sido explicado anteriormente.

Este programa tiene dos modos de funcionamiento. En el primero de ellos el usuario introduce el valor de la inductancia que necesita, la frecuencia a la va a trabajar y la tolerancia que puede presenta esta bobina, el programa devuelve el modelo equivalente y las características geométricas de aquella bobina que obtengan el mejor factor de calidad para dichas condiciones y siempre y cuando trabajemos con bobinas octogonales. Así mismo el programa genera el fichero ".cif" de la bobina elegida para que el usuario la pueda utilizar en *Cadence.*

Por otro lado, el programa tiene otro modo de funcionamiento en el cual el usuario podrá elegir, dentro de unos márgenes que nos dará el tipo de tecnología que esté usando, los parámetros que crea necesarios para su inductancia, como por ejemplo el número de vueltas de la bobina, su radio externo, el ancho de las pistas, etc., así como los tipos de bobina que ponemos a su disposición y que son la bobina cuadrada, octogonal, etc, dependiente del número de lados seleccionado. En este caso el programa devuelve el modelo equivalente de la bobina así como el fichero ".cif" de la misma.

5.2 Diagrama de bloques

En este apartado se explicará una visión global de las funciones principales de la herramienta Software así como los módulos que las componen y la relaciones que se establecen entre estos.

Ya desplegada la ventana principal se observan 4 botones principales que ejecutan las principales funciones de la herramienta Software en cuestión, estas son; *Search Inductor, Calculated Parameters, Generate CIF y Graph L and Q.*

En la figura 5.1 podemos observar el diagrama de bloques del botón “Search Inductor”.

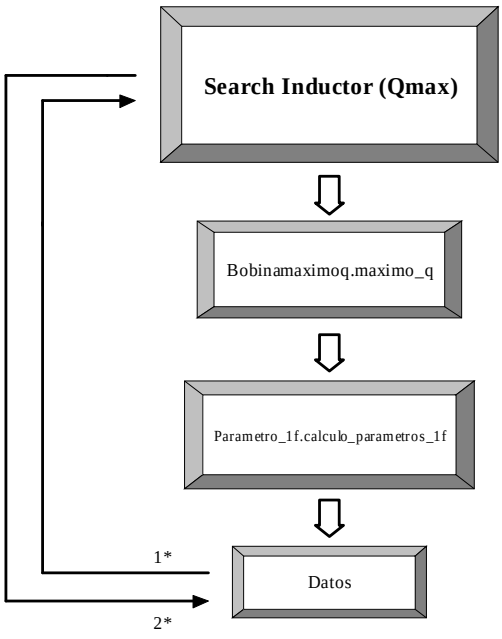

Figura 5.1 Diagrama de bloques “Seach Inductor”.

Partiendo de unos datos iniciales (inductancia “L”, tolerancia “T”, número de metales “Number of Metals” y frecuencia “F”) se obtiene como resultado los parámetros geométricos de la bobina con máximo factor de calidad Q así como el modelo equivalente en PI de esta bobina. Todos estos resultados son almacenados en la clase Datos que se comentará en el próximo apartado y finalmente expuestos en pantalla. Sólo válido para el caso de bobinas octogonales.

Seguimos con el diagrama del botón *Calculated Parameters* (Figura 5.2).

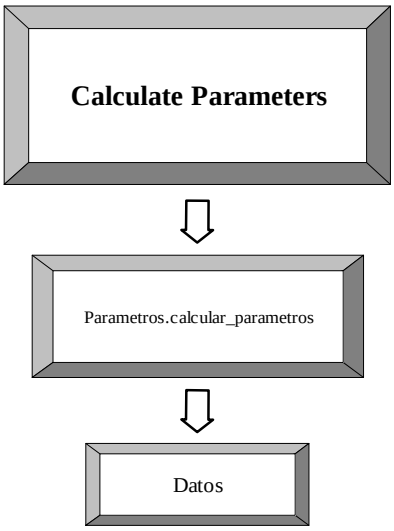

Figura 5.2 Diagrama de bloques "Calculate Parameters".

En este caso; conocida la geometría de la bobina, el número de metales, la frecuencia a la que se desea trabajar, así como un barrido en frecuencia, se obtiene el modelo equivalente en PI para la bobina especificada. Sólo válido para el caso de bobinas octogonales.

El diagrama del botón *Generate CIF* se muestra en la Figura 5.3.

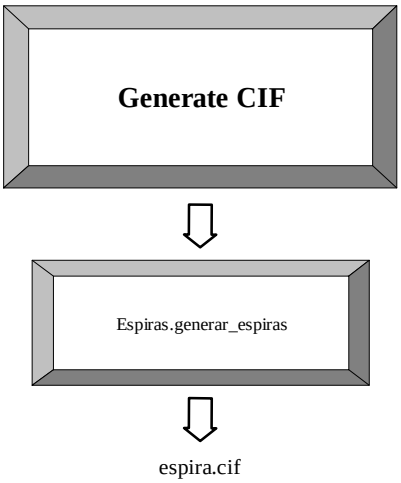

Figura 5.3 Diagrama de bloques "Generate CIF".

Una vez determinados los parámetros geométricos, este botón genera un fichero tipo ".cif" que importado desde *Cadence* dará lugar al layout de la bobina en cuestión.

Finalmente en la Figura 5.4 nos encontramos con el diagrama *Graph L and Q.*

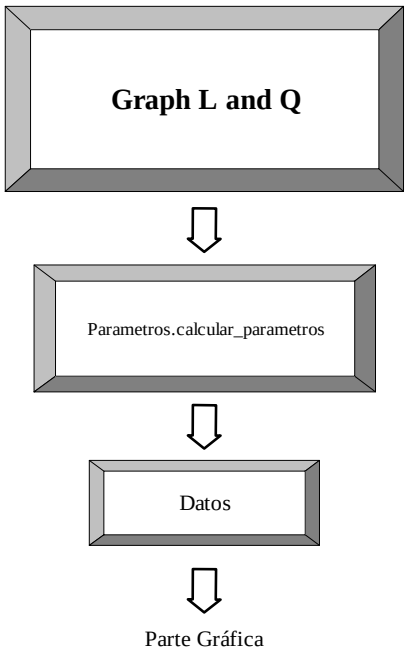

Figura 5.4 Diagrama de bloques "Graph L and Q".

Este botón es el encargado de representar por pantalla la curva de inductancia y factor de calidad para un barrido de frecuencia y una bobina específica según sus parámetros geométricos y el número de metales. Sólo válido para el caso de bobinas octogonales.

5.3 Estructura de las Clases Java de la herramienta

A continuación vamos a conocer todas las clases que han sido desarrolladas para la construcción de la herramienta. Cada una de estas clases es un archivo de texto con extensión *".java"* en el que se ha escrito todo el código utilizando software *Netbeans* IDE, anteriormente comentado en el capítulo 3.

5.3.1 Clase VentanaPrincipal

La clase VentanaPrincipal es una interfaz gráfica y es la ventana principal encargada de llamar al resto de clases pasándole los datos de entrada.

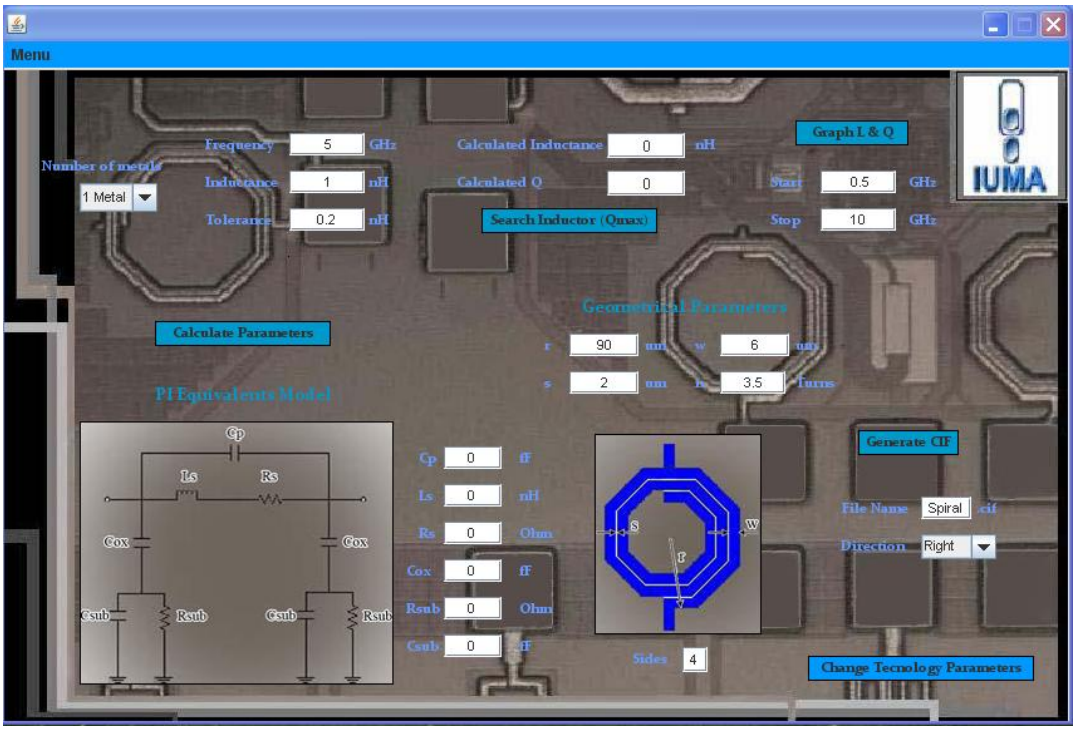

Figura 5.5 Ventana principal de la herramienta.

5.3.2 Clase Tecnologia

Esta clase como la anterior es una interfaz gráfica, donde se despliega una ventana con los parámetros característicos de la tecnología. Modificando estos parámetros, obtenemos los resultados para la tecnología especificada.

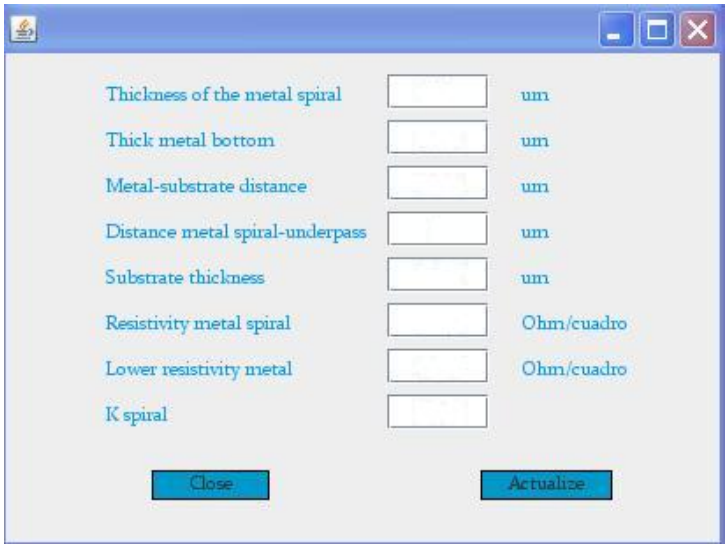

Figura 5.6 Ventana de los parámetros de la tecnología.

5.3.3 Clase BobinaMaximoQ

En esta clase se realiza la búsqueda de la bobina con las características especificadas de inductancia (**L_pedida**), tolerancia (**T**), número de metal (**N_metales**) y frecuencia (**f**) que tenga el mayor factor de calidad (**Q_calc**). Para ello se ha programado un barrido por todos los posibles valores de los parámetros geométricos.

Estos parámetros geométricos son radio exterior de la espira (**r**), separación entre pistas (**s**), ancho de la pista (**w**) y número de vueltas (**n**). Éstos varían entre unos valores predeterminados y, mediante la clase ***Parametros_1f***, que posteriormente se explicará obtenemos las bobinas que cumplen las características especificadas. El funcionamiento de esta clase consiste en rellenar una matriz con los datos de todas las posibles bobinas y, para finalizar, realiza una búsqueda de la bobina con mayor factor de calidad.

Como resultado final tendremos las características geométricas y los componentes del modelo equivalente de la bobina de máximo factor de calidad. Además, como añadido, obtendremos por separado unos vectores con las características independientes de todas las bobinas posibles. Todos estos datos serán guardados en la clase ***Datos*** para su posterior uso en otras clases de la herramienta.

5.3.4 Clase Parametros_1f

En el interior de esta clase se calculan los componentes del modelo equivalente de la bobina; inductancia serie (**Ls**), capacidad paralelo (**Cp**), resistencia serie (**Rs**), capacidad debida al óxido (**Cox**), capacidad debida al sustrato (**Csub**) y la resistencia debida al sustrato (**Rsub**), así como su factor de calidad (**Q**). Todo ello calculado para una frecuencia fija, número de metales y parámetros geométricos determinados.

Todos los resultados se almacenan en la clase ***Datos*** para su utilización en otras clases o para ser presentados en la interfaz gráfica de representación correspondiente.

5.3.5 Clase Parametros

En este apartado se explicará el funcionamiento de la clase ***Parametros***. Ésta parte de los valores: barrido en frecuencia inicializado con una frecuencia de inicio (**minf**) y una frecuencia final (**maxf**), frecuencia a la que se quiere hacer trabajar la bobina (**f**), parámetros geométricos (**r,s,w,n**) anteriormente comentados y número de metales (**N_met**). A partir de estos datos se realiza el cálculo de los componentes del modelo

equivalente de la bobina: impedancia serie (**Ls**), capacidad paralelo (**Cp**), resistencia serie (**Rs**), capacidad debida al oxido (**Cox**), capacidad debida al sustrato (**Csub**) y resistencia debida al sustrato (**Rsub**), así como la inductancia (**L_calc**) y el factor de calidad (**Q_calc**) que presentaría la bobina simulada para la frecuencia específica a la que se quiere hacer trabajar "**f**".

Además esta clase genera dos vectores con los valores de inductancia y factor de calidad (**L,Q**) para el barrido en frecuencia seleccionado. Estos vectores serán de gran importancia en el momento de representar la variación de la inductancia y el factor de calidad en función de la frecuencia en la interfaz gráfica de representación correspondiente.

En el caso que se produzca un barrido erróneo o unos valores geométricos imposibles, la clase devolverá por pantalla un mensaje informando de estos errores en forma de ventana de aviso.

5.3.6 Clase Espiras

Esta es una de las clases principales, ya que en ella se crea una lista de puntos en un fichero ".cif" que, importada desde *Cadente*, nos dibujará los *layouts* de todas las espirales que deseemos. Esta lista de puntos son en realidad las coordenadas cartesianas de los extremos de las pistas cuya unión nos dará como representación una espiral.

Para el cálculo de las coordenadas de los puntos de la espiral debemos tener en cuenta que el ángulo que nos crea el segundo punto con respecto a los ejes, es $2 \cdot \pi / n^o$ *de lados de la espiral.* El primer punto lo conocemos ya que sabemos el valor del radio externo de la espiral, el ancho de la pista, el número de vueltas de la espira y la distancia entre pistas:

Punto inicio = radio externo - ancho de pista / 2 – vueltas * (ancho de pista + espacio entre pistas)

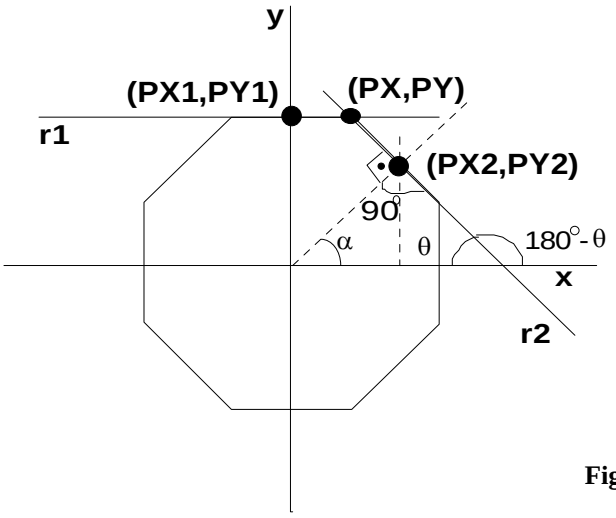

Figura 5.7

En la figura anterior podemos observar como hallaremos los distintos puntos, siguiendo el patrón que a continuación se explica. La coordenada a hallar es *(PX,PY)*, que resulta de la intersección de las dos rectas r1 y r2.

$$r1 \rightarrow y = PY1 \tag{5.1}$$

$$r2 \rightarrow y = a + \mathrm{tg}(\pi - \theta)x \tag{5.2}$$

Donde:

$$\theta = \pi - \frac{\pi}{2} - \alpha \tag{5.3}$$

Sustituyendo en *r2* los puntos *(PX2,PY2)* hallamos *a*:

$$PY2 = a + \mathrm{tg}\left(\pi - \frac{\pi}{2} + \alpha\right)PX2 \tag{5.4}$$

$$a = PY2 - \mathrm{tg}\left(\frac{\pi}{2} + \alpha\right)PX2 \tag{5.5}$$

Entonces *r2* es:

$$y = PY2 - PX2\,\mathrm{tg}\left(\frac{\pi}{2}+\alpha\right) + \mathrm{tg}\left(\frac{\pi}{2}+\alpha\right)x \qquad (5.6)$$

$$y = PY2 - \mathrm{tg}\left(\frac{\pi}{2}+\alpha\right)\cdot\left[PX2 - x\right] \qquad (5.7)$$

Los puntos de corte entre las dos rectas, como ya hemos dicho, serán *(PX,PY)*. Obtenemos directamente *PY*, sustituyendo en la ecuación 5.1:

$$PY = PY1 \qquad (5.8)$$

Lo mismo hacemos para hallar *PX*, sustituyendo en la ecuación 5.7:

$$PY1 = PY2 - \mathrm{tg}\left(\frac{\pi}{2}+\alpha\right)\cdot\left[PX2 - PX\right] \qquad (5.9)$$

$$PX = \frac{PY1 - PY2 - \mathrm{tg}\left(\frac{\pi}{2}+\alpha\right)\cdot PX2}{\mathrm{tg}\left(\frac{\pi}{2}+\alpha\right)} \qquad (5.10)$$

Pero nos quedan por hallar los siguientes puntos en cuyas ecuaciones tendremos que tener en cuenta estos primeros. Tomemos por ejemplo ahora una espiral de doce lados.

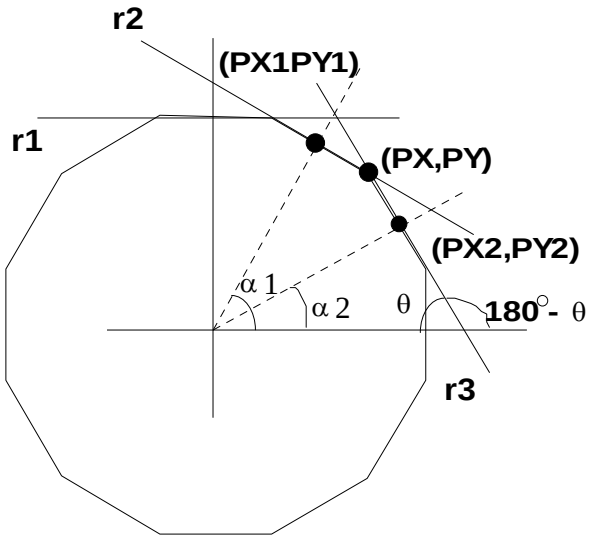

Figura 5.8

Inicialicemos variables respecto a las ecuaciones anteriores:

$PX1=PX2$

$PY1=PY2$

$\alpha = \alpha 1$

$\theta 1 = (\pi / 2 - \alpha 1)$

La ecuación de la recta 2 queda de la siguiente forma:

$$y = PY1 - PX1 \cdot \mathrm{tg}\,\theta 1 + x \cdot \mathrm{tg}\,\theta 1 \qquad (5.11)$$

La ecuación de la recta 3:

$$y = PY2 - PX2 \cdot \mathrm{tg}\,\theta 2 + x \cdot \mathrm{tg}\,\theta 2 \qquad (5.12)$$

Resolviendo el sistema de ecuaciones hallamos los puntos de corte y por tanto las nuevas coordenadas *(PX,PY)*:

$$\left.\begin{aligned} y &= PY1 - \mathrm{tg}\,\theta 1 \cdot [PX1 - x] \\ y &= PY2 - \mathrm{tg}\,\theta 2 \cdot [PX2 - x] \end{aligned}\right\} \qquad (5.13)$$

$$x = PX = \frac{PY2 - PY1 - PX2 \cdot \mathrm{tg}\,\theta 2 + PX1 \cdot \mathrm{tg}\,\theta 1}{\mathrm{tg}\,\theta 1 - \mathrm{tg}\,\theta 2} \qquad (5.14)$$

PY la hallaríamos sustituyendo el valor de esta ecuación en cualquiera de las dos ecuaciones 5.13.

Como lo que estamos dibujando es una espira, cada punto deberá tener un incremento, que será función del ancho de la pista, del espacio entre ambas y de los lados de la espiral.

5.3.7 Clase Datos

Aquí se almacenan todos los datos de salida de todas las clases con la ventaja de tener acceso directo a esta información desde otras clases sin ningún tipo de problema, ya que es una clase con ámbito global a toda la jerarquía de clases de la herramienta. Debido a que en Java las clases sólo pueden devolver una única variable de salida se optó por integrar en la herramienta esta clase para satisfacer las necesidades de esta.

5.3.8 Clase ParametrosGlobales

La función de la clase ***ParametrosGlobales*** es la de almacenar los parámetros globales de la tecnología ***"S35D4 de la de la fundidora AMS"*** en cuestión, pudiéndose variar éstos para cualquier tecnología a partir de la ventana ***Tecnologia*** anteriormente explicada.

La razón para crear de esta clase es la de disponer de una herramienta versátil; capaz de trabajar con cualquier tecnología conocidos sus parámetros globales.

5.3.9 Clase Complex

Ésta es una clase de apoyo, y está diseñada para el manejo de número complejos. Con ella se ha facilitado los cálculos matemáticos en las clases ***Parametros*** y ***Parametros_1f*** para la suma, resta, multiplicación y división de números complejos, así como manejar conjunta o independientemente la parte real o imaginaria del número complejo en cuestión.

5.3.10 Clase GraficoPuntoCursorZoom

Esta clase es necesaria para la representación gráfica de curvas. Esta clase genera los lienzos, las rejillas y el cursor para desplazarnos por la gráfica. Para un correcto funcionamiento se predeterminan los valores máximos y mínimos tanto en el eje "x" como en el "y". Finalmente, se le pasan los valores a representar mediante dos vectores (**visorX, visorY**), obteniéndose como resultado la gráfica deseada.

5.3.11 Clase BotoneriaZoom

BotoneriaZoom es la clase destinada a la creación de todos los botones encargados de variar el zoom de la interfaz gráfica de representación, dando la opción de acercarnos o alejarnos en las curvas donde representamos la inductancia frente a la frecuencia y en las curvas de factor de calidad frente a frecuencia, pudiéndose así observar con más claridad los detalles de máximos, mínimos e incluso valores en puntos determinados.

5.3.12 Conversión de una aplicación Java en Applet

Una vez finalizado el desarrollo de las clases Java que conforman la herramienta, se debe modificar el código para poder usarla también como aplicación Web, para ello se transforma en un Applet, es un componente de una aplicación que se ejecuta en el contexto de otro programa. Un Java applet es un código Java que carece de un método main, por eso se utiliza principalmente para el trabajo de páginas web, ya que es un pequeño programa que es utilizado en una página HTML y representado por una pequeña pantalla gráfica dentro de ésta.

A continuación se especificaran las variaciones en el código y donde deben hacerse para la correcta obtención del Applet:

1.- Agrega esta línea a los import:

```
import javax.swing.JApplet;
```

2.- En vez que extienda de JFrame has que extienda de JApplet

```
public class Interface extends JApplet implements ActionListener
```

3.- Cambia el nombre del constructor (en este caso Interface) por init, el constructor quedara:

```
//crea la interfaz grafica de usuario GUI

 public void init()
```

4.- Elimina los metodos setTitle, setResizable y setDefaultCloseOperation, ya que no existen en JApplet porque no se necesitan.

5.- Elimina el método principal (main(String args[]) y todo lo que este entre sus {}, es innecesario.

Al compilarlo se creara un .class, pero no se podrá ejecutar con el jcreator, estos se visualizan a través de páginas Web, así que se debe crearse una página Web sencilla que lo mande llamar.

5.4 Ejemplos

Finalmente, una vez obtenida la herramienta Web y comprobado su correcto funcionamiento se ha desarrollado una serie de ejemplos que nos servirán para ilustrar el funcionamiento de la herramienta así como las diferentes opciones que nos permiten. Como resultado de estos ejemplos se obtendrán los diferentes *layouts* para cada supuesto, los cuales han sido dibujados por *Cadence* y con anterioridad creados los ficheros de los *layouts* ".cif" por la herramienta Web creada.

Ejemplo 1

En este primer ejemplo se ha fijado la frecuencia de trabajo a la que la bobina va a trabajar, la inductancia que esta debe presentar y la tolerancia máxima que puede tener. En este caso particular damos los siguientes valores:

F *= 5 GHz*
L *= 1 nH*
T *= 0.2 nH*

Como resultado se observa una bobina con un factor de calidad (***Q*** = 13.348) y las siguientes características geométricas:

***r** = 70 µm*

***s** = 2 µm*

***w** = 9 µm*

***n** = 2.5 turs (vueltas)*

Para finalizar la creación del fichero .cif determinaremos las características del layout a dibujar así como el nombre con el que será guardado. En este caso determinaremos que el número de lados sea cuatro (***Sides*** = 8), que la bobina se dibuje de dentro a fuera hacia la derecha (eligiendo en el campo ***Direction*** la opción ***Right***) y que el nombre de guardado sea "***Spiral1.cif***". Dandole al boton "***Generate CIF"*** se genera el fichero ".cif", el cual importado correctamente en *Cadence* da como resultado el *layout* de este supuesto. (Figura 5.9).

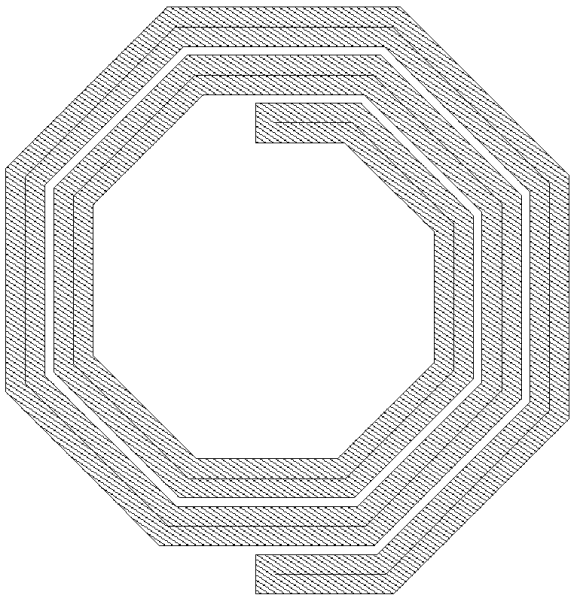

Figura 5.9 Spiral1.

Ejemplo 2

En este segundo ejemplo se ha hecho lo mismo que en el caso anterior, para las siguientes especificaciones; frecuencia de trabajo, inductancia y tolerancia.

***F** = 10 GHz*

***L** = 3 nH*

***T** = 0.1 nH*

Se obtiene como resultado una bobina con un factor de calidad (***Q*** = 8.558) y las siguientes características geométricas:

r = *75 µm*
s = *2 µm*
w = *5 µm*
n = *3.5 turs (vueltas)*

Se determina esta vez que la bobina tenga más lados **(** ***Sides*** = 8), que la bobina se dibuje de dentro a fiera hacia la derecha (eligiendo en el campo ***Direction*** la opción ***Right***) y que el nombre de guardado sea "***Spiral2.cif***". Una vez generado e importado el fichero ".cif" en Cadente se obtiene como resultado el siguiente *layout*. (Figura 5.10).

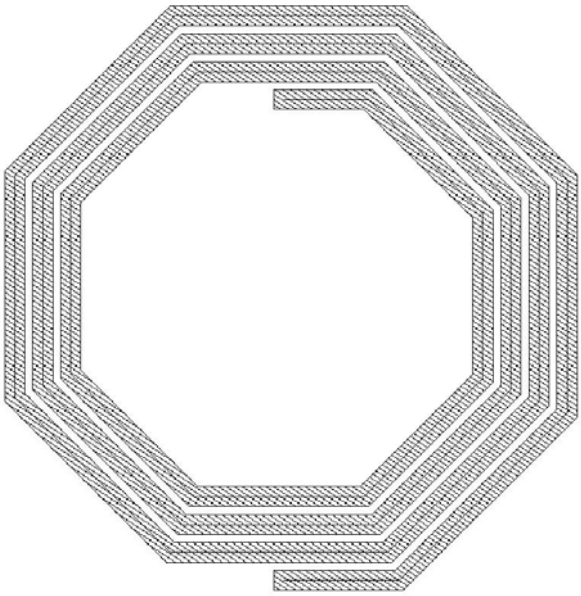

Figura 5.10 Spiral2.

Este ejemplo es para una bobina octogonal por lo cual podemos obtener las gráficas de inductancia y factor de calidad "Q" para un barrido de frecuencia. Para ello hacemos clic en la opción ***"Graph L and Q"*** y determinamos un barrido mediante una frecuencia inicial **"Start"** y una frecuencia final **"Stop"**, este caso se a determinado:

Start = *0.5 GHz*
Stop = *10 GHz*

Obteniéndose como resultado las graficas de inductancia y factor de calidad, Figura 5.11 y Figura 5.12.

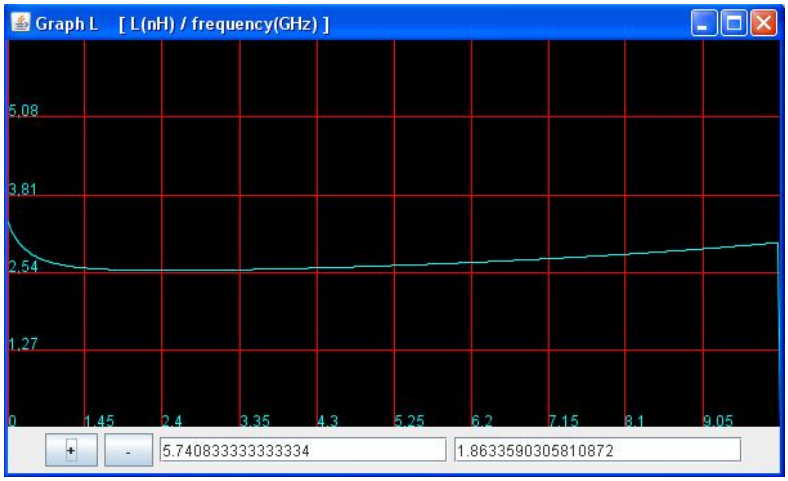

Figura 5.11 Gráfica de la Inductancia.

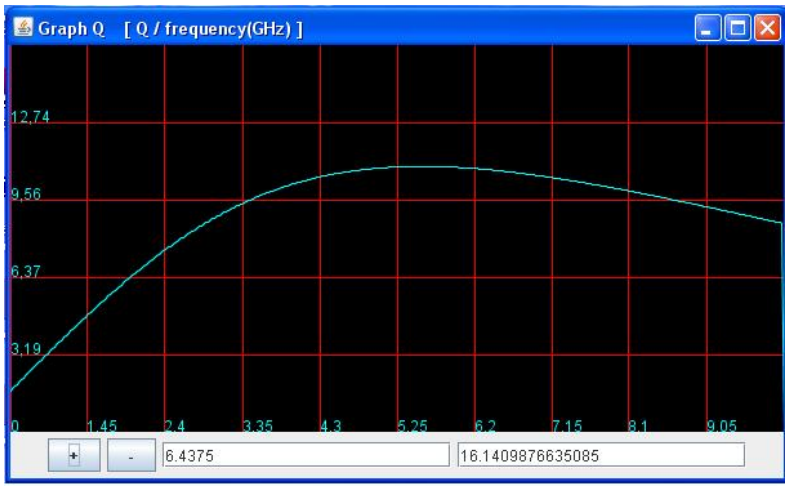

Figura 5.12 Gráfica del Factor de Calidad.

Ejemplo 3

Partiendo de los datos geométricos se genera el fichero ".cif". Para ello introducimos en las casillas correspondientes el radio externo de la espira (***r***) , la separación entre espiras (***s***), el ancho de la espira (***w***) y el número de vueltas de la bobina (***n***). Dándole al botón ***"Generate CIF"*** se genera el fichero ".cif" siempre que las

características geométricas sean posibles, si no, devolverá un mensaje notificando la imposibilidad de crear esta espira.

A continuación se especifican las características geométricas para este tercer caso, observándose que se ha decidido por una bobina octogonal (**Sides** = 8).

***r** = 150 µm*
***s** = 5 µm*
***w** = 20 µm*
***n** = 5 turs (vueltas)*

Una vez generado e importado el fichero ".cif" en Cadence se obtiene como resultado el *layout* para este supuesto (Figura 5.13), el cual se ha guardado como ***"Spiral3.cif"*** y presenta un factor de calidad (***Q** = 1.263*).

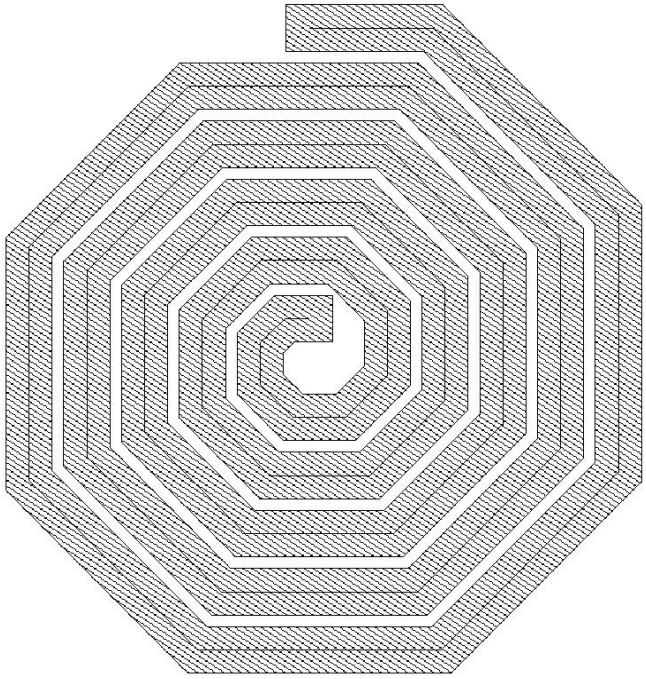

Figura 5.13 Spiral3.

Como se trata de una bobina octogonal podemos generar el modelo equivalente en PI así como el factor de calidad y la inductancia. Esto es posible mediante las opciones ***"Calculate Parameters"*** y ***"Seach Inductor(Qmax)"*** dando como resultado:

C_p *= 70.83 fF*
L_s *= 3.663 nH*
R_s *= 6.07 Ohm*
C_{ox} *= 331.14 fF*
R_{sub}*= 245.732 Ohm*

C_{sub} = *81.43 fF*

L_{cal} = *8.834 nH*

Q = *1.263*

Ejemplo 4

Como cuarto ejemplo se ha considerado dibujar la espiral en la otra dirección de dentro a fuera hacia la izquierda (eligiendo en el campo ***Direction*** la opción ***Left***) , la cual será guardada con el nombre de ***"Spiral4.cif"***. Las especificaciones son las características geométricas:

r = *160 µm*

s = *2 µm*

w = *6 µm*

n = *2.5 turs (vueltas)*

Sides = *6*

Generado e importado el fichero ".cif" en Cadente se obtiene como resultado el *layout* correspondiente. (Figura 5.14).

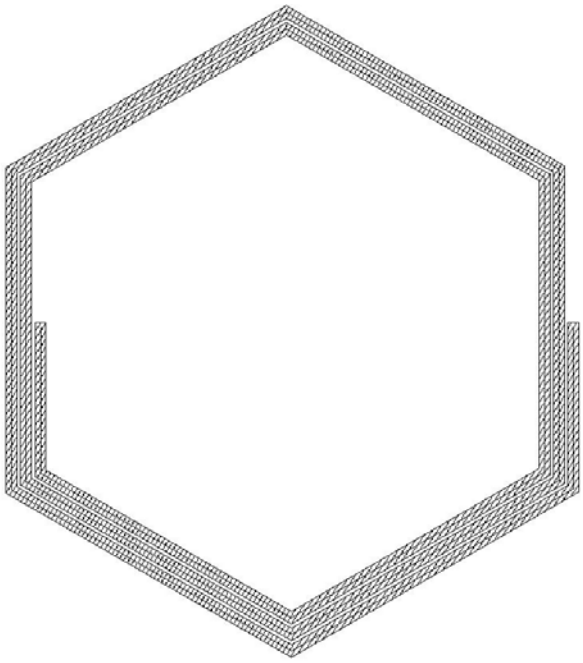

Figura 5.14 Spiral4.

Ejemplo 5

En el siguiente ejemplo se ha especificado una bobina cuadrada pero en primer lugar se ha determinado que se dibuje de dentro a fuera hacia la derecha (eligiendo en el campo ***Direction*** la opción ***Right***) , la cual será guardada con el nombre de ***"Spiral5.cif"***. Las especificaciones son las características geométricas:

r *= 300 µm*
s *= 5 µm*
w *= 20 µm*
n *= 8.5 turs (vueltas)*
Sides *= 4*

Generado e importado el fichero ".cif" en Cadence se obtiene como resultado el *layout* correspondiente. (Figura 5.15).

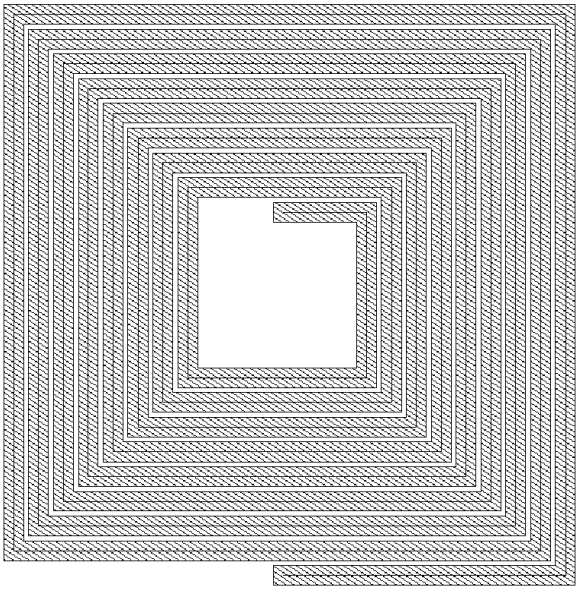

Figura 5.15 Spiral5.

Ejemplo 6

Para este último ejemplo se ha dibujado una bobina de número de lados impar (**Sides =** 11). En primer lugar se ha determinado que se dibuje de dentro a fuera hacia la derecha (eligiendo en el campo ***Direction*** la opción ***Right***) , esta bobina será guardada

con el nombre de ***"Spiral6.cif"***. Las especificaciones son las características geométricas:

r *= 200 µm*
s *= 5 µm*
w *= 15 µm*
n *= 3.5 turs (vueltas)*

El layout generado se muestra en la Figura 5.16.

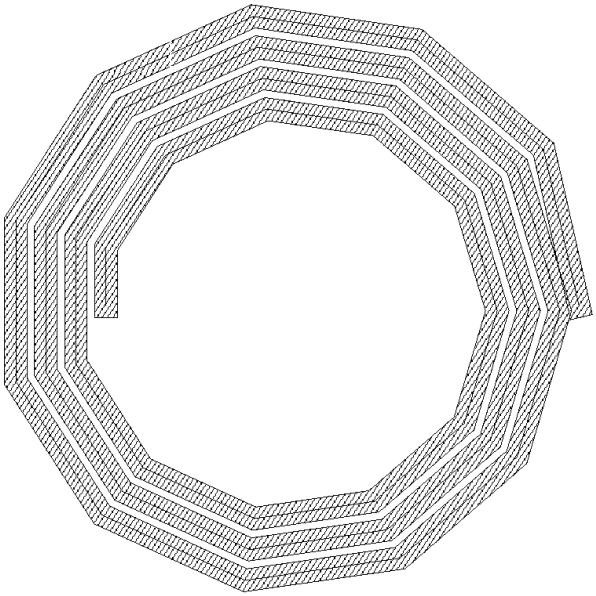

Figura 5.16 Spiral6.

Capítulo 6

Guía de usuario

6.1 Introducción

Esta herramienta es un sistema software desarrollado en Java basado en un modelo paramétrico aplicado al modelo circuital equivalente en pi de un inductor integrado. Dicho modelo ofrece una ecuación para cada elemento unitario de dicho circuito equivalente. De esta forma, esta herramienta otorga valores a estos elementos en base a los parametros de la tecnología sobre la que se diseña el inductor y a los parametros geometricos de la espiral.

6.2 Descripción de componentes

La interfaz de la herramienta está compuesta por diversos elementos editables y botones:

- Number of metals.
- Geometrical Parameters.
- Start y Stop.
- Calculated inductance y calculated Q.
- Calculate parameters.
- Graph L and Q.
- Search inductor (Q_{max}).
- Generate CIF.

- Change Tecnology parameters.
- Menu.

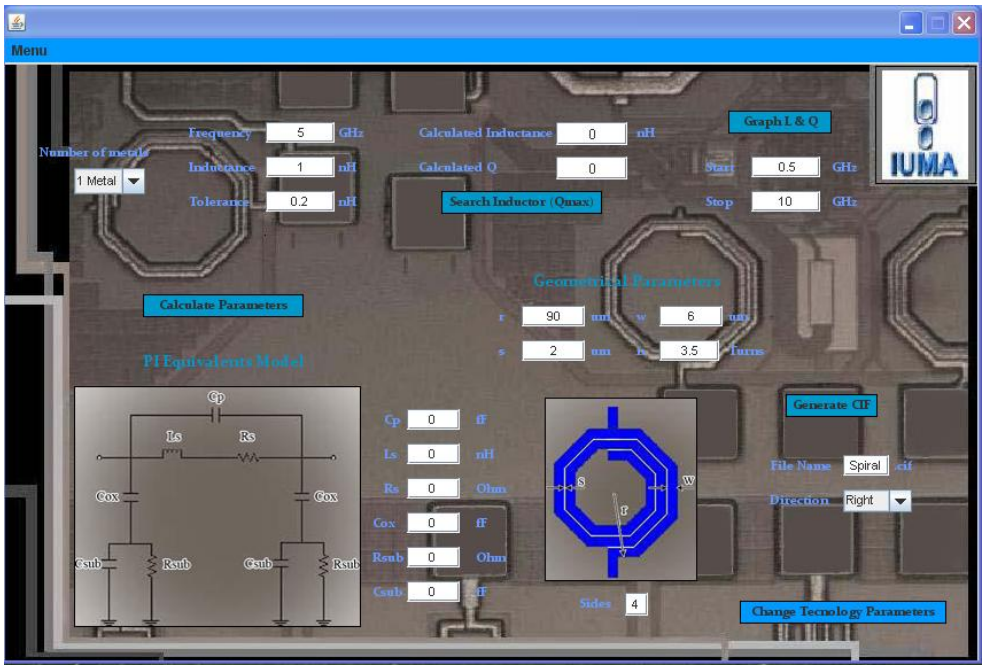

Figura 6.1 Interfaz.

6.2.1 Number of metals

Este es uno de los elementos básicos de elección de la herramienta. A través del mismo se puede elegir si el modelo paramétrico elegido corresponde con:

Inductores cuya espira está formada por un solo metal. El metal inmediatamente inferior a este correspondería con el metal utilizado como underpass. Esta será la opción de 1 Metal.

Inductores formados por dos metales en paralelo. Esta será la opción de 2 Metales.

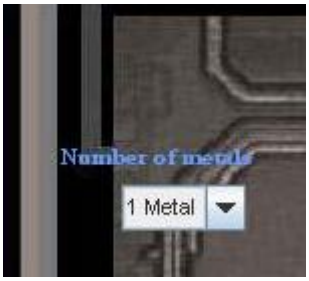

Figura 6.2 Numero de metales usados en el inductor.

6.2.2 Geometrical Parameters

A traves de estos elementos modificables se podrá indicar los valores de:

Radio exterior “r” (en µm).

Anchura de pistas “w” (en µm).

Número de vueltas “n”.

Espaciado de pistas “s” (en µm).

Con estos parámetros ya es posible definir el modelo equivalente y poder representar la inductancia y factor de calidad de la bobina definida.

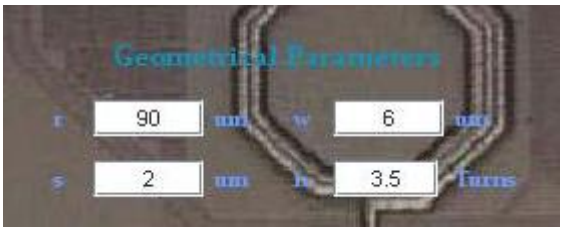

Figura 6.3 Definición de los parámetros geométricos.

6.2.3 Start y Stop

Con estos parametros se indica la frecuencia de comienzo y final necesarios en el calculo de los elementos del modelo equivalente y la representación gráfica de la inductancia y el factor de calidad calculados.

Figura 6.4 Barrido de frecuencias.

6.2.4 Calculated inductance y calculated Q

En estos elementos se indica simplemente la inductancia y el factor de calidad calculadas mediante el modelo parametrico para la frecuencia específica indicada en el paramétro correspondiente.

Figura 6.5 Inductancia y factor de calidad calculados por el modelo paramétrico.

6.2.5 Calculate parameters

Al pulsar el boton de *" Calculate parameters"* se utilizarán los parámetros geométricos indicados en los distintos elementos de entrada de la interfaz para calcular el valor de cada uno de los elementos unitarios que definen el ciruito equivalente del inductor integrado buscado. Una vez calculados, son mostrados en pantalla.

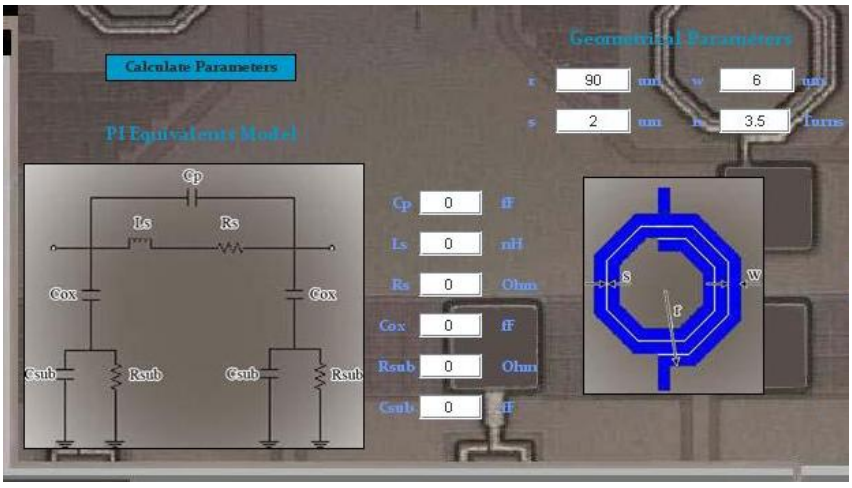

Figura 6.6 Cálculo de los elementos del modelo a partir de los parámetros geométricos.

6.2.6 Graph L and Q

Al hacer click sobre el boton ***"Graph L and Q"*** se muestra en dos gráficas independientes con dichos parámetros calculados a partir de los parámetros geométricos

indicados. Dichas gráficas estarán definidas en el rango impuesto entre la mínima y la máxima frecuencia propuesta.

En las gráficas se puede posicionar el puntero sobre ellas indicando en los casilleros de la parte inferior el punto en el que se encuentra tanto en frecuencia como en inductancia o factor de calidad, según en que gráfica nos encontremos; ademas, tenemos la opción de hacer *zoom* en las gráficas.

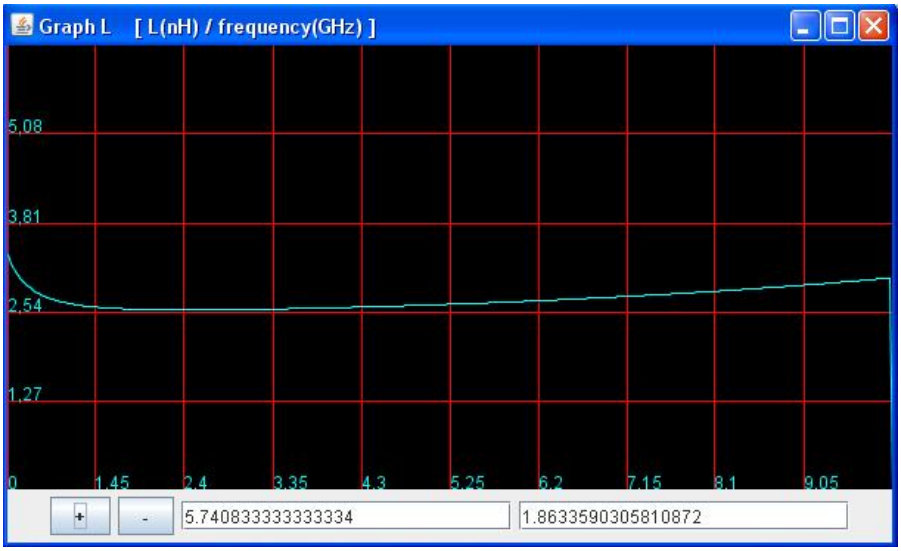

Figura 6.7 Gráfica del valor inductivo.

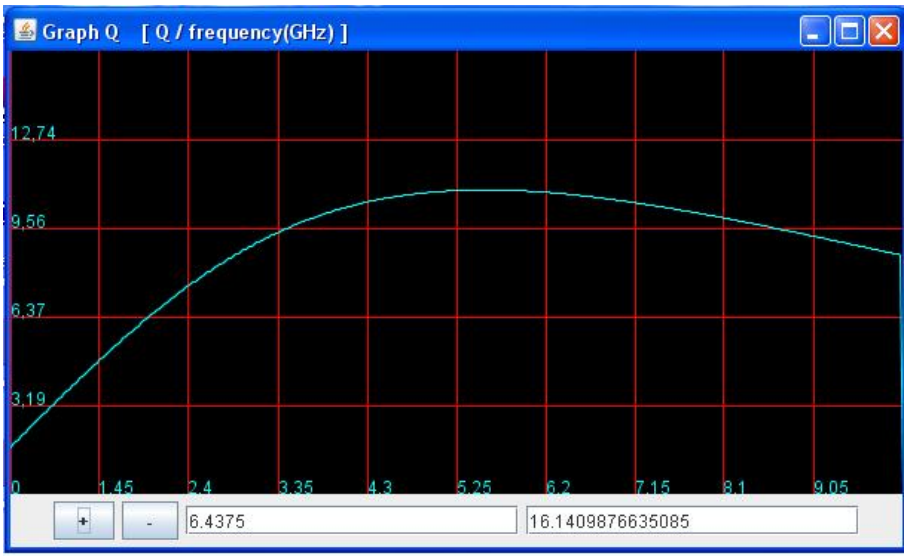

Figura 6.8 Gráfica del factor de calidad.

6.2.7 Search inductor (Q_{max})

Mediante el modelo paramétrico desarrollado, se ha elaborado un algoritmo que busque la bobina con mayor factor de calidad posible para una frecuencia e inductancia dadas. Para ello se debe indicar también la tolerancia permitida en el valor inductivo antes de realizar la búsqueda.

Figura 6.9 Parámetros necesarios para el buscador de máximo Q.

Entonces, el algoritmo realiza un barrido por los distintos parámetros geométricos y busca aquellas bobinas que cumplan los requisitos de frecuencia, inductancia y tolerancia en la misma. De entre todas las combinaciones, se tomará la de mayor factor de calidad y se mostrará en pantalla los parámetros geométricos, los valores de los elementos unitarios, la inductancia y el factor de calidad obtenidos para la frecuencia pedida.

6.2.8 Generate CIF

Al pulsar el boton de ***"Generate CIF"*** se genera un archivo ".cif" con las indicaciones de diseño de una espira para los parámetros geométricos introducidos en la interfaz principal, que se dibujará según la dirección especificada *"Direction"*. Este archivo será guardado con el nombre especificado en *"File Name"*.

Figura 6.10 Especificaciones del archivo ".cif".

6.2.9 Change Tecnology parameters

Al pusar el botón de ***"Change Tecnology parameters"*** aparece una nueva pantalla donde el usuario puede editar los distintos parámetros que conciernen a la tecnología a utilizar en cada caso.

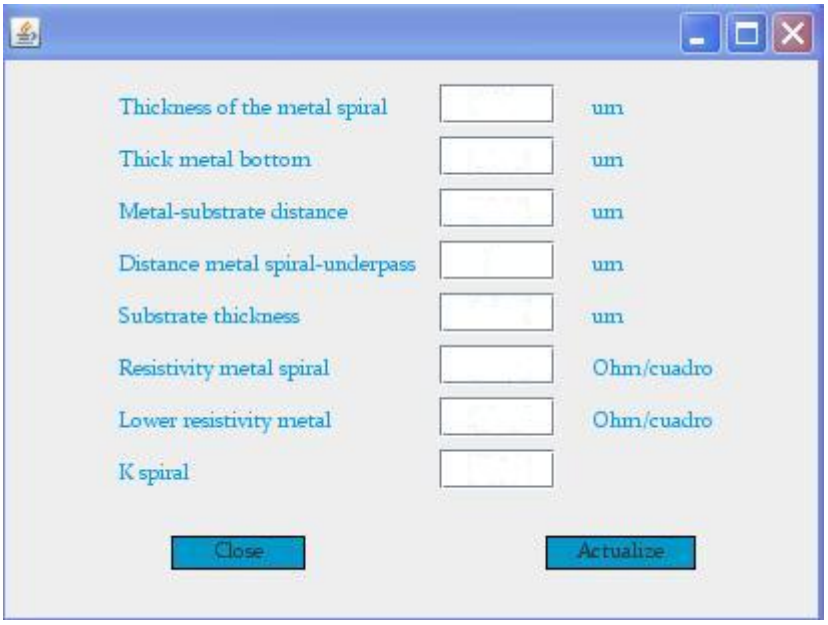

Figura 6.11 Pantalla para el cambio de parámetros.

Dichos parámetros hacen referencia ha grosores de las capas metálicas, distancias relativas y resistividades. Además, se puede modificar el valor de la constante utilizada en el modelo paramétrico con el fin de ver su efecto en el cálculo de los parámetros de la bobina.

6.2.10 Mensajes de error

A continuación se muestran los posibles mensajes de error que pueden aparecer durante la ejecución cuando la herramienta no es utilizada de manera correcta o no se puede llevar a cabo alguna de las tareas solicitadas.

Este mensaje aparece en pantalla cuando los valore de mínima y máxima frecuencia en el barrido son incoherentes.

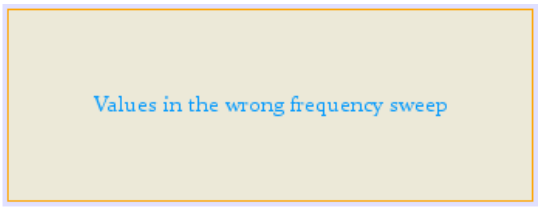

Figura 6.12

Este mensaje aparece cuando la frecuencia indicada en la casilla "Frequency" está fuera del rango impuesto por el barrido de frecuencia. En este caso, no se puede mostrar el factor de calidad, la inductancia y la resistencia serie para tal frecuencia.

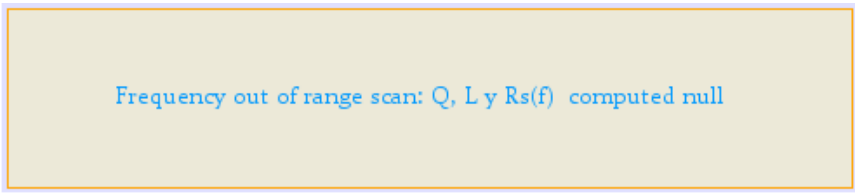

Figura 6.13

Este mensaje aparece cuando los valores geométricos indicados en la interfaz no son coherentes para generar un inductor integrado.

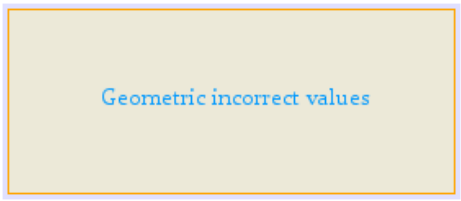

Figura 6.14

Este mensaje aparece en el caso que el buscador de bobinas con máximo factor de calidad no haya encontrado ningún inductor que cumpla los requisistos impuestos por el usuario.

Figura 6.15

Capítulo 7

Conclusiones y líneas futuras

7.1 Introducción

Concluida la realización de este proyecto de fin de carrera, se expone las conclusiones obtenidas en el presente capítulo. A continuación se presenta también las posibles líneas de investigación futuras aplicables a este proyecto.

7.2 Conclusiones

Finalizado el capítulo 1 quedan claro los conceptos básicos de los inductores y el desarrollo de la memoria, prosiguiendo con el siguiente capítulo, en el que se desarrolla las características de la tecnología utilizada.

En el tercer capítulo de esta memoria se ha profundizado el los entornos empleados para el desarrollo de este proyecto, estos son el NetBeans IDE y el *Cadence*. Tras el estudio de estos entornos se pudo diseñar la herramienta Web escrita en el lenguaje Java, que tras su ejecución obtiene un fichero ".cif" que se puede importar a *Cadence* para obtener finalmente el layout correspondiente a la bobina especificada.

A continuación en el capítulo 4 se expone la herramienta desarrollada en [46] llamada I-MODEL, así como su funcionamiento y sus aplicaciones. Este software servirá como base a nuestra herramienta Web.

En el capítulo 5, se presentó y desarrolló la herramienta Web creada en este proyecto. Dicho capitulo reúne todos los datos teóricos anteriormente explicados para la obtención de la herramienta Web encargada de la obtención de la bobina de máximo factor de calidad, el cálculo de los parámetros del modelo equivalente en PI de una bobina determinada por su geometría y por la frecuencia en la que esta trabaja, la representación grafica de un barrido en frecuencia tanto en inductancia como en factor

de calidad para bobinas octogonales y, finalmente, la obtención de los ficheros".cif" para el correcto dibujado de las bobinas en *Cadence*.

Por otro lado, con esta herramienta Web se ha obtenido mayores ventajas frente al I-MODEL desarrollado en [46], ya que la herramienta Web desarrollada no necesita de la instalación de ningún software para su uso. Esto es debido a que se encuentra en un servidor Web, donde cualquier usuario con acceso a el puede cargarla y utilizarla libremente. Además tiene el generador de archivo ".cif" para la creación de *layouts* de cualquier tipo de bobina novedad destaca frente al I-MODEL.

Como comentario final puntualizar la versatilidad de esta herramienta Web para cualquier tecnología, ya que mediante la ventana "Parameters" se pueden cambiar las características de la tecnología a simular, obteniéndose como resultado una herramienta mucho más potente para la hora de diseñar *layouts*.

7.3 Líneas futuras

Este proyecto de fin de carrera posee un marcado contenido científico-tecnológico, por ello es pertinente indicar posibles líneas de trabajo futuras abiertas a partir del mismo, que son diversas y se presenta las más relevantes:

- Fabricación de distintas bobinas en la tecnología ***S35D4*** con una serie de requisitos geométricos y del modelo paramétrico.
- Realizar un estudio de estructuras alternativas de inductores integrados para la obtención de altos valores inductivos y de factor de calidad. Existen diversos estudios que proponen la utilización de configuraciones distintas en el *layout* de un inductor en tecnologías multicapa para este fin. Así, se puede encontrar en la literatura bobinas balanceadas o apiladas de forma no paralela [40].
- Estudiar otros modelos equivalentes de inductores integrados para su aplicación para cualquier tecnología. En los últimos años se ha publicado modelos alternativos al modelo clásico en π utilizado en este proyecto [3], [47], y que pudieran obtener resultados más precisos en un mayor rango de frecuencias.
- Seguir con la búsqueda de nuevas estructuras inductivas en tecnologías de bajo coste, para mejorar las prestaciones de los inductores a altas frecuencias.

- Estudio en profundidad del modelado del sustrato teniendo en cuenta el efecto que sobre la resistencia y la capacidad del mismo ejerce la distribución de campo eléctrico lateral cuando el área de la espiral se reduce.
- Mejorar la herramienta para que esta generé los *Underpass* de las bobinas integradas. Así como la posibilidad de generar distintos tipo de bobinas, como son las alargadas, apiladas, etc.

Anexo a

Javadoc

Class VentanaPrincipal

```
java.lang.Object
  └ java.awt.Component
      └ java.awt.Container
          └ java.awt.Panel
              └ java.applet.Applet
                  └ javax.swing.JApplet
                      └ VentanaPrincipal
```

All Implemented Interfaces:

java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable, javax.accessibility.Accessible, javax.swing.RootPaneContainer

---

```
public class VentanaPrincipal
extends javax.swing.JApplet
```

See Also:

Serialized Form

Nested Class Summary

Nested classes/interfaces inherited from class javax.swing.JApplet

javax.swing.JApplet.AccessibleJApplet

Nested classes/interfaces inherited from class java.applet.Applet

java.applet.Applet.AccessibleApplet

Nested classes/interfaces inherited from class java.awt.Panel

java.awt.Panel.AccessibleAWTPanel

Nested classes/interfaces inherited from class java.awt.Container

java.awt.Container.AccessibleAWTContainer

Nested classes/interfaces inherited from class java.awt.Component

java.awt.Component.AccessibleAWTComponent, java.awt.Component.BaselineResizeBehavior, java.awt.Component.BltBufferStrategy, java.awt.Component.FlipBufferStrategy

Field Summary

Fields inherited from class javax.swing.JApplet

accessibleContext, rootPane, rootPaneCheckingEnabled

Fields inherited from class java.awt.Component

BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT, TOP_ALIGNMENT

Fields inherited from interface java.awt.image.ImageObserver

ABORT, ALLBITS, ERROR, FRAMEBITS, HEIGHT, PROPERTIES, SOMEBITS, WIDTH

Constructor Summary

VentanaPrincipal()

Method Summary

void	**init**() Creates new form ventana
static void	**main**(java.lang.String[] args)
double	**redondear**(double numero, int decimales)

Methods inherited from class javax.swing.JApplet

addImpl, createRootPane, getAccessibleContext, getContentPane, getGlassPane, getGraphics, getJMenuBar, getLayeredPane, getRootPane, getTransferHandler, isRootPaneCheckingEnabled, paramString, remove, repaint, setContentPane, setGlassPane, setJMenuBar, setLayeredPane, setLayout, setRootPane, setRootPaneCheckingEnabled, setTransferHandler, update

Methods inherited from class java.applet.Applet

destroy, getAppletContext, getAppletInfo, getAudioClip, getAudioClip, getCodeBase, getDocumentBase, getImage, getImage, getLocale, getParameter, getParameterInfo, isActive, newAudioClip, play, play, resize, resize, setStub, showStatus, start, stop

Methods inherited from class java.awt.Panel

addNotify

Methods inherited from class java.awt.Container

add, add, add, add, add, addContainerListener, addPropertyChangeListener, addPropertyChangeListener, applyComponentOrientation, areFocusTraversalKeysSet, countComponents, deliverEvent, doLayout, findComponentAt, findComponentAt, getAlignmentX, getAlignmentY, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents, getComponentZOrder, getContainerListeners, getFocusTraversalKeys, getFocusTraversalPolicy, getInsets, getLayout, getListeners, getMaximumSize, getMinimumSize, getMousePosition, getPreferredSize, insets, invalidate, isAncestorOf, isFocusCycleRoot, isFocusCycleRoot, isFocusTraversalPolicyProvider, isFocusTraversalPolicySet, layout, list, list, locate, minimumSize, paint, paintComponents, preferredSize, print, printComponents, processContainerEvent, processEvent, remove, removeAll, removeContainerListener, removeNotify, setComponentZOrder, setFocusCycleRoot, setFocusTraversalKeys, setFocusTraversalPolicy, setFocusTraversalPolicyProvider, setFont, transferFocusBackward, transferFocusDownCycle, validate, validateTree

Methods inherited from class java.awt.Component

action, add, addComponentListener, addFocusListener, addHierarchyBoundsListener, addHierarchyListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addMouseWheelListener, bounds, checkImage, checkImage, coalesceEvents, contains, contains, createImage, createImage, createVolatileImage, createVolatileImage, disable, disableEvents, dispatchEvent, enable, enable, enableEvents, enableInputMethods, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, getBackground, getBaseline, getBaselineResizeBehavior, getBounds, getBounds, getColorModel, getComponentListeners, getComponentOrientation, getCursor, getDropTarget, getFocusCycleRootAncestor, getFocusListeners, getFocusTraversalKeysEnabled, getFont, getFontMetrics, getForeground, getGraphicsConfiguration, getHeight, getHierarchyBoundsListeners, getHierarchyListeners, getIgnoreRepaint, getInputContext, getInputMethodListeners, getInputMethodRequests, getKeyListeners, getLocation, getLocation, getLocationOnScreen, getMouseListeners, getMouseMotionListeners, getMousePosition, getMouseWheelListeners, getName, getParent, getPeer, getPropertyChangeListeners, getPropertyChangeListeners, getSize, getSize, getToolkit, getTreeLock, getWidth, getX, getY, gotFocus, handleEvent, hasFocus, hide, imageUpdate, inside, isBackgroundSet, isCursorSet, isDisplayable, isDoubleBuffered, isEnabled, isFocusable, isFocusOwner, isFocusTraversable, isFontSet, isForegroundSet, isLightweight, isMaximumSizeSet, isMinimumSizeSet, isOpaque, isPreferredSizeSet, isShowing, isValid, isVisible, keyDown, keyUp, list, list, list, location, lostFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move, nextFocus, paintAll, postEvent, prepareImage, prepareImage, printAll, processComponentEvent, processFocusEvent, processHierarchyBoundsEvent, processHierarchyEvent, processInputMethodEvent, processKeyEvent, processMouseEvent, processMouseMotionEvent, processMouseWheelEvent, remove, removeComponentListener, removeFocusListener, removeHierarchyBoundsListener, removeHierarchyListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removeMouseWheelListener, removePropertyChangeListener, removePropertyChangeListener, repaint, repaint, repaint, requestFocus, requestFocus, requestFocusInWindow, requestFocusInWindow, reshape, setBackground, setBounds, setBounds, setComponentOrientation, setCursor, setDropTarget, setEnabled, setFocusable, setFocusTraversalKeysEnabled, setForeground, setIgnoreRepaint, setLocale, setLocation, setLocation, setMaximumSize, setMinimumSize, setName, setPreferredSize, setSize, setSize, setVisible, show, show, size, toString, transferFocus, transferFocusUpCycle

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

VentanaPrincipal

```
public VentanaPrincipal()
```

Method Detail

init

```
public void init()
```

Creates new form ventana

Overrides:

`init` in class `java.applet.Applet`

---

redondear

```
public double redondear(double numero,
                        int decimales)
```

---

main

```
public static void main(java.lang.String[] args)
```

Parameters:

`args` - the command line arguments

Class Tecnologia

```
java.lang.Object
  └ java.awt.Component
      └ java.awt.Container
          └ java.awt.Panel
              └ java.applet.Applet
                  └ javax.swing.JApplet
                      └ Tecnologia
```

All Implemented Interfaces:

java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable, javax.accessibility.Accessible, javax.swing.RootPaneContainer

---

```
public class Tecnologia
extends javax.swing.JApplet
```

See Also:

Serialized Form

Nested Class Summary

Nested classes/interfaces inherited from class javax.swing.JApplet

javax.swing.JApplet.AccessibleJApplet

Nested classes/interfaces inherited from class java.applet.Applet

java.applet.Applet.AccessibleApplet

Nested classes/interfaces inherited from class java.awt.Panel

java.awt.Panel.AccessibleAWTPanel

Nested classes/interfaces inherited from class java.awt.Container

java.awt.Container.AccessibleAWTContainer

Nested classes/interfaces inherited from class java.awt.Component

java.awt.Component.AccessibleAWTComponent, java.awt.Component.BaselineResizeBehavior, java.awt.Component.BltBufferStrategy, java.awt.Component.FlipBufferStrategy

Field Summary

Fields inherited from class javax.swing.JApplet

accessibleContext, rootPane, rootPaneCheckingEnabled

Fields inherited from class java.awt.Component

BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT, TOP_ALIGNMENT

Fields inherited from interface java.awt.image.ImageObserver

ABORT, ALLBITS, ERROR, FRAMEBITS, HEIGHT, PROPERTIES, SOMEBITS, WIDTH

Constructor Summary

Tecnologia()

Method Summary

void	**init**() Creates new form ventana

Methods inherited from class javax.swing.JApplet

addImpl, createRootPane, getAccessibleContext, getContentPane, getGlassPane, getGraphics, getJMenuBar, getLayeredPane, getRootPane, getTransferHandler, isRootPaneCheckingEnabled, paramString, remove, repaint, setContentPane, setGlassPane, setJMenuBar, setLayeredPane, setLayout, setRootPane, setRootPaneCheckingEnabled, setTransferHandler, update

Methods inherited from class java.applet.Applet

destroy, getAppletContext, getAppletInfo, getAudioClip, getAudioClip, getCodeBase, getDocumentBase, getImage, getImage, getLocale, getParameter, getParameterInfo, isActive, newAudioClip, play, play, resize, resize, setStub, showStatus, start, stop

Methods inherited from class java.awt.Panel

addNotify

Methods inherited from class java.awt.Container

add, add, add, add, add, addContainerListener, addPropertyChangeListener, addPropertyChangeListener, applyComponentOrientation, areFocusTraversalKeysSet, countComponents, deliverEvent, doLayout, findComponentAt, findComponentAt, getAlignmentX, getAlignmentY, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents, getComponentZOrder, getContainerListeners, getFocusTraversalKeys, getFocusTraversalPolicy, getInsets, getLayout, getListeners, getMaximumSize, getMinimumSize, getMousePosition, getPreferredSize, insets, invalidate, isAncestorOf, isFocusCycleRoot, isFocusCycleRoot, isFocusTraversalPolicyProvider, isFocusTraversalPolicySet, layout, list, list, locate, minimumSize, paint, paintComponents, preferredSize, print, printComponents, processContainerEvent, processEvent, remove, removeAll, removeContainerListener, removeNotify, setComponentZOrder, setFocusCycleRoot, setFocusTraversalKeys, setFocusTraversalPolicy, setFocusTraversalPolicyProvider, setFont, transferFocusBackward, transferFocusDownCycle, validate, validateTree

Methods inherited from class java.awt.Component

action, add, addComponentListener, addFocusListener, addHierarchyBoundsListener, addHierarchyListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addMouseWheelListener, bounds, checkImage, checkImage, coalesceEvents, contains, contains, createImage, createImage, createVolatileImage, createVolatileImage, disable, disableEvents, dispatchEvent, enable, enable, enableEvents, enableInputMethods, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, getBackground, getBaseline, getBaselineResizeBehavior, getBounds, getBounds, getColorModel, getComponentListeners, getComponentOrientation, getCursor, getDropTarget, getFocusCycleRootAncestor, getFocusListeners, getFocusTraversalKeysEnabled, getFont, getFontMetrics, getForeground, getGraphicsConfiguration, getHeight, getHierarchyBoundsListeners, getHierarchyListeners, getIgnoreRepaint, getInputContext, getInputMethodListeners, getInputMethodRequests, getKeyListeners, getLocation, getLocation, getLocationOnScreen, getMouseListeners, getMouseMotionListeners, getMousePosition, getMouseWheelListeners, getName, getParent, getPeer, getPropertyChangeListeners, getPropertyChangeListeners, getSize, getSize, getToolkit, getTreeLock, getWidth, getX, getY, gotFocus, handleEvent, hasFocus, hide, imageUpdate, inside, isBackgroundSet, isCursorSet, isDisplayable, isDoubleBuffered, isEnabled, isFocusable, isFocusOwner, isFocusTraversable, isFontSet, isForegroundSet, isLightweight, isMaximumSizeSet, isMinimumSizeSet, isOpaque, isPreferredSizeSet, isShowing, isValid, isVisible, keyDown, keyUp, list, list, list, location, lostFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move, nextFocus, paintAll, postEvent, prepareImage, prepareImage, printAll, processComponentEvent, processFocusEvent, processHierarchyBoundsEvent, processHierarchyEvent, processInputMethodEvent, processKeyEvent, processMouseEvent, processMouseMotionEvent, processMouseWheelEvent, remove, removeComponentListener, removeFocusListener, removeHierarchyBoundsListener, removeHierarchyListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removeMouseWheelListener, removePropertyChangeListener, removePropertyChangeListener, repaint, repaint, repaint, requestFocus, requestFocus, requestFocusInWindow, requestFocusInWindow, reshape, setBackground, setBounds, setBounds, setComponentOrientation, setCursor, setDropTarget, setEnabled, setFocusable, setFocusTraversalKeysEnabled, setForeground, setIgnoreRepaint, setLocale, setLocation, setLocation, setMaximumSize, setMinimumSize, setName, setPreferredSize, setSize, setSize, setVisible, show, show, size, toString, transferFocus, transferFocusUpCycle

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

Tecnologia

```
public Tecnologia()
```

Method Detail

init

```
public void init()
```

Creates new form ventana

Overrides:

`init` in class `java.applet.Applet`

Package Class Use Tree Deprecated Index Help

PREV CLASS NEXT CLASS FRAMES NO FRAMES All Classes

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

Class BobinaMaximoQ

java.lang.Object
└ **BobinaMaximoQ**

public class **BobinaMaximoQ**
extends java.lang.Object

Constructor Summary

BobinaMaximoQ()

Method Summary

int	**calcular_maximo**(int filasmax, double[][] result)
void	**maximo_q**(float L_pedida, float Tolerancia, int N_metales, float f)
double	**redondear**(double numero, int decimales)

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

BobinaMaximoQ

public **BobinaMaximoQ**()

Method Detail

calcular_maximo

public int **calcular_maximo**(int filasmax,
double[][] result)

maximo_q

```
public void maximo_q(float L_pedida,
                     float Tolerancia,
                     int N_metales,
                     float f)
```

redondear

```
public double redondear(double numero,
                        int decimales)
```

Package Class Use Tree Deprecated Index Help

PREV CLASS NEXT CLASS FRAMES NO FRAMES All Classes
SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

Class BotoneriaZoom

```
java.lang.Object
  └ java.awt.Component
      └ java.awt.Container
          └ javax.swing.JComponent
              └ javax.swing.JPanel
                  └ BotoneriaZoom
```

All Implemented Interfaces:

java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable, javax.accessibility.Accessible

---

```
public class BotoneriaZoom
extends javax.swing.JPanel
```

See Also:

Serialized Form

Nested Class Summary

Nested classes/interfaces inherited from class javax.swing.JPanel

javax.swing.JPanel.AccessibleJPanel

Nested classes/interfaces inherited from class javax.swing.JComponent

javax.swing.JComponent.AccessibleJComponent

Nested classes/interfaces inherited from class java.awt.Container

java.awt.Container.AccessibleAWTContainer

Nested classes/interfaces inherited from class java.awt.Component

java.awt.Component.AccessibleAWTComponent, java.awt.Component.BaselineResizeBehavior, java.awt.Component.BltBufferStrategy, java.awt.Component.FlipBufferStrategy

Field Summary

Fields inherited from class javax.swing.JComponent

accessibleContext, listenerList, TOOL_TIP_TEXT_KEY, ui, UNDEFINED_CONDITION, WHEN_ANCESTOR_OF_FOCUSED_COMPONENT, WHEN_FOCUSED, WHEN_IN_FOCUSED_WINDOW

Fields inherited from class java.awt.Component

BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT, TOP_ALIGNMENT

Fields inherited from interface java.awt.image.ImageObserver

ABORT, ALLBITS, ERROR, FRAMEBITS, HEIGHT, PROPERTIES, SOMEBITS, WIDTH

Constructor Summary

BotoneriaZoom(chuidiang.graficos.InterfaceEscalaGrafica escala)
Creates a new instance of BotoneriaZoom

Method Summary

Methods inherited from class javax.swing.JPanel

getAccessibleContext, getUI, getUIClassID, paramString, setUI, updateUI

Methods inherited from class javax.swing.JComponent

addAncestorListener, addNotify, addVetoableChangeListener, computeVisibleRect, contains, createToolTip, disable, enable, firePropertyChange, firePropertyChange, firePropertyChange, fireVetoableChange, getActionForKeyStroke, getActionMap, getAlignmentX, getAlignmentY, getAncestorListeners, getAutoscrolls, getBaseline, getBaselineResizeBehavior, getBorder, getBounds, getClientProperty, getComponentGraphics, getComponentPopupMenu, getConditionForKeyStroke, getDebugGraphicsOptions, getDefaultLocale, getFontMetrics, getGraphics, getHeight, getInheritsPopupMenu, getInputMap, getInputMap, getInputVerifier, getInsets, getInsets, getListeners, getLocation, getMaximumSize, getMinimumSize, getNextFocusableComponent, getPopupLocation, getPreferredSize, getRegisteredKeyStrokes, getRootPane, getSize, getToolTipLocation, getToolTipText, getToolTipText, getTopLevelAncestor, getTransferHandler, getVerifyInputWhenFocusTarget, getVetoableChangeListeners, getVisibleRect, getWidth, getX, getY, grabFocus, isDoubleBuffered, isLightweightComponent, isManagingFocus, isOpaque, isOptimizedDrawingEnabled, isPaintingForPrint, isPaintingTile, isRequestFocusEnabled, isValidateRoot, paint, paintBorder, paintChildren, paintComponent, paintImmediately, paintImmediately, print, printAll, printBorder, printChildren, printComponent, processComponentKeyEvent, processKeyBinding, processKeyEvent, processMouseEvent, processMouseMotionEvent, putClientProperty, registerKeyboardAction, registerKeyboardAction, removeAncestorListener, removeNotify, removeVetoableChangeListener, repaint, repaint, requestDefaultFocus, requestFocus, requestFocus, requestFocusInWindow, requestFocusInWindow, resetKeyboardActions, reshape, revalidate, scrollRectToVisible, setActionMap, setAlignmentX, setAlignmentY, setAutoscrolls, setBackground, setBorder, setComponentPopupMenu, setDebugGraphicsOptions, setDefaultLocale, setDoubleBuffered, setEnabled, setFocusTraversalKeys, setFont, setForeground, setInheritsPopupMenu, setInputMap, setInputVerifier, setMaximumSize, setMinimumSize, setNextFocusableComponent, setOpaque,

setPreferredSize, setRequestFocusEnabled, setToolTipText, setTransferHandler, setUI, setVerifyInputWhenFocusTarget, setVisible, unregisterKeyboardAction, update

Methods inherited from class java.awt.Container

add, add, add, add, add, addContainerListener, addImpl, addPropertyChangeListener, addPropertyChangeListener, applyComponentOrientation, areFocusTraversalKeysSet, countComponents, deliverEvent, doLayout, findComponentAt, findComponentAt, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents, getComponentZOrder, getContainerListeners, getFocusTraversalKeys, getFocusTraversalPolicy, getLayout, getMousePosition, insets, invalidate, isAncestorOf, isFocusCycleRoot, isFocusCycleRoot, isFocusTraversalPolicyProvider, isFocusTraversalPolicySet, layout, list, list, locate, minimumSize, paintComponents, preferredSize, printComponents, processContainerEvent, processEvent, remove, remove, removeAll, removeContainerListener, setComponentZOrder, setFocusCycleRoot, setFocusTraversalPolicy, setFocusTraversalPolicyProvider, setLayout, transferFocusBackward, transferFocusDownCycle, validate, validateTree

Methods inherited from class java.awt.Component

action, add, addComponentListener, addFocusListener, addHierarchyBoundsListener, addHierarchyListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addMouseWheelListener, bounds, checkImage, checkImage, coalesceEvents, contains, createImage, createImage, createVolatileImage, createVolatileImage, disableEvents, dispatchEvent, enable, enableEvents, enableInputMethods, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, getBackground, getBounds, getColorModel, getComponentListeners, getComponentOrientation, getCursor, getDropTarget, getFocusCycleRootAncestor, getFocusListeners, getFocusTraversalKeysEnabled, getFont, getForeground, getGraphicsConfiguration, getHierarchyBoundsListeners, getHierarchyListeners, getIgnoreRepaint, getInputContext, getInputMethodListeners, getInputMethodRequests, getKeyListeners, getLocale, getLocation, getLocationOnScreen, getMouseListeners, getMouseMotionListeners, getMousePosition, getMouseWheelListeners, getName, getParent, getPeer, getPropertyChangeListeners, getPropertyChangeListeners, getSize, getToolkit, getTreeLock, gotFocus, handleEvent, hasFocus, hide, imageUpdate, inside, isBackgroundSet, isCursorSet, isDisplayable, isEnabled, isFocusable, isFocusOwner, isFocusTraversable, isFontSet, isForegroundSet, isLightweight, isMaximumSizeSet, isMinimumSizeSet, isPreferredSizeSet, isShowing, isValid, isVisible, keyDown, keyUp, list, list, list, location, lostFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move, nextFocus, paintAll, postEvent, prepareImage, prepareImage, processComponentEvent, processFocusEvent, processHierarchyBoundsEvent, processHierarchyEvent, processInputMethodEvent, processMouseWheelEvent, remove, removeComponentListener, removeFocusListener, removeHierarchyBoundsListener, removeHierarchyListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removeMouseWheelListener, removePropertyChangeListener, removePropertyChangeListener, repaint, repaint, repaint, resize, resize, setBounds, setBounds, setComponentOrientation, setCursor, setDropTarget, setFocusable, setFocusTraversalKeysEnabled, setIgnoreRepaint, setLocale, setLocation, setLocation, setName, setSize, setSize, show, show, size, toString, transferFocus, transferFocusUpCycle

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

BotoneriaZoom

```
public BotoneriaZoom(chuidiang.graficos.InterfaceEscalaGrafica escala)
```

Creates a new instance of BotoneriaZoom

Package Class Use Tree Deprecated Index Help

PREV CLASS NEXT CLASS FRAMES NO FRAMES All Classes

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

Class Datos

java.lang.Object
  └ **Datos**

public class **Datos**
extends java.lang.Object

Constructor Summary

Datos()

Method Summary

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

Datos

public **Datos**()

Class Espiras

java.lang.Object
  └ **Espiras**

public class **Espiras**
extends java.lang.Object

Constructor Summary

Espiras()

Method Summary

void	**generar_espiras**(int direct, int lados, double r, double s, double w, double n, java.lang.String name)
double	**redondear**(double numero, int decimales)

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

Espiras

public **Espiras**()

Method Detail

generar_espiras

public void **generar_espiras**(int direct,
int lados,
double r,

```
                        double s,
                        double w,
                        double n,
                        java.lang.String name)
```

---

redondear

```
public double redondear(double numero,
                        int decimales)
```

Package Class Use Tree Deprecated Index Help

PREV CLASS NEXT CLASS FRAMES NO FRAMES All Classes

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

Class GraficoPuntosCursorZoom

```
java.lang.Object
  └ java.awt.Component
      └ java.awt.Container
          └ java.awt.Panel
              └ java.applet.Applet
                  └ javax.swing.JApplet
                      └ GraficoPuntosCursorZoom
```

All Implemented Interfaces:

java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable, javax.accessibility.Accessible, javax.swing.RootPaneContainer

---

```
public class GraficoPuntosCursorZoom
extends javax.swing.JApplet
```

Ejemplo de uso de los Objetos gr□ficos Seno y RejillaFija.

See Also:

Serialized Form

Nested Class Summary

Nested classes/interfaces inherited from class javax.swing.JApplet

javax.swing.JApplet.AccessibleJApplet

Nested classes/interfaces inherited from class java.applet.Applet

java.applet.Applet.AccessibleApplet

Nested classes/interfaces inherited from class java.awt.Panel

java.awt.Panel.AccessibleAWTPanel

Nested classes/interfaces inherited from class java.awt.Container

java.awt.Container.AccessibleAWTContainer

Nested classes/interfaces inherited from class java.awt.Component
java.awt.Component.AccessibleAWTComponent, java.awt.Component.BaselineResizeBehavior, java.awt.Component.BltBufferStrategy, java.awt.Component.FlipBufferStrategy

Field Summary

Fields inherited from class javax.swing.JApplet
accessibleContext, rootPane, rootPaneCheckingEnabled

Fields inherited from class java.awt.Component
BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT, TOP_ALIGNMENT

Fields inherited from interface java.awt.image.ImageObserver
ABORT, ALLBITS, ERROR, FRAMEBITS, HEIGHT, PROPERTIES, SOMEBITS, WIDTH

Constructor Summary

GraficoPuntosCursorZoom()

Method Summary

void	init()
static void	main(java.lang.String[] args) Instancia un Lienzo y mete dentro un Seno y una RejillaFija con la escala adecuada.

Methods inherited from class javax.swing.JApplet
addImpl, createRootPane, getAccessibleContext, getContentPane, getGlassPane, getGraphics, getJMenuBar, getLayeredPane, getRootPane, getTransferHandler, isRootPaneCheckingEnabled, paramString, remove, repaint, setContentPane, setGlassPane, setJMenuBar, setLayeredPane, setLayout, setRootPane, setRootPaneCheckingEnabled, setTransferHandler, update

Methods inherited from class java.applet.Applet
destroy, getAppletContext, getAppletInfo, getAudioClip, getAudioClip, getCodeBase, getDocumentBase, getImage, getImage, getLocale, getParameter, getParameterInfo, isActive, newAudioClip, play, play, resize, resize, setStub, showStatus, start, stop

Methods inherited from class java.awt.Panel
addNotify

Methods inherited from class java.awt.Container

add, add, add, add, add, addContainerListener, addPropertyChangeListener, addPropertyChangeListener, applyComponentOrientation, areFocusTraversalKeysSet, countComponents, deliverEvent, doLayout, findComponentAt, findComponentAt, getAlignmentX, getAlignmentY, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents, getComponentZOrder, getContainerListeners, getFocusTraversalKeys, getFocusTraversalPolicy, getInsets, getLayout, getListeners, getMaximumSize, getMinimumSize, getMousePosition, getPreferredSize, insets, invalidate, isAncestorOf, isFocusCycleRoot, isFocusCycleRoot, isFocusTraversalPolicyProvider, isFocusTraversalPolicySet, layout, list, list, locate, minimumSize, paint, paintComponents, preferredSize, print, printComponents, processContainerEvent, processEvent, remove, removeAll, removeContainerListener, removeNotify, setComponentZOrder, setFocusCycleRoot, setFocusTraversalKeys, setFocusTraversalPolicy, setFocusTraversalPolicyProvider, setFont, transferFocusBackward, transferFocusDownCycle, validate, validateTree

Methods inherited from class java.awt.Component

action, add, addComponentListener, addFocusListener, addHierarchyBoundsListener, addHierarchyListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addMouseWheelListener, bounds, checkImage, checkImage, coalesceEvents, contains, contains, createImage, createImage, createVolatileImage, createVolatileImage, disable, disableEvents, dispatchEvent, enable, enable, enableEvents, enableInputMethods, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, getBackground, getBaseline, getBaselineResizeBehavior, getBounds, getBounds, getColorModel, getComponentListeners, getComponentOrientation, getCursor, getDropTarget, getFocusCycleRootAncestor, getFocusListeners, getFocusTraversalKeysEnabled, getFont, getFontMetrics, getForeground, getGraphicsConfiguration, getHeight, getHierarchyBoundsListeners, getHierarchyListeners, getIgnoreRepaint, getInputContext, getInputMethodListeners, getInputMethodRequests, getKeyListeners, getLocation, getLocation, getLocationOnScreen, getMouseListeners, getMouseMotionListeners, getMousePosition, getMouseWheelListeners, getName, getParent, getPeer, getPropertyChangeListeners, getPropertyChangeListeners, getSize, getSize, getToolkit, getTreeLock, getWidth, getX, getY, gotFocus, handleEvent, hasFocus, hide, imageUpdate, inside, isBackgroundSet, isCursorSet, isDisplayable, isDoubleBuffered, isEnabled, isFocusable, isFocusOwner, isFocusTraversable, isFontSet, isForegroundSet, isLightweight, isMaximumSizeSet, isMinimumSizeSet, isOpaque, isPreferredSizeSet, isShowing, isValid, isVisible, keyDown, keyUp, list, list, list, location, lostFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move, nextFocus, paintAll, postEvent, prepareImage, prepareImage, printAll, processComponentEvent, processFocusEvent, processHierarchyBoundsEvent, processHierarchyEvent, processInputMethodEvent, processKeyEvent, processMouseEvent, processMouseMotionEvent, processMouseWheelEvent, remove, removeComponentListener, removeFocusListener, removeHierarchyBoundsListener, removeHierarchyListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removeMouseWheelListener, removePropertyChangeListener, removePropertyChangeListener, repaint, repaint, repaint, requestFocus, requestFocus, requestFocusInWindow, requestFocusInWindow, reshape, setBackground, setBounds, setBounds, setComponentOrientation, setCursor, setDropTarget, setEnabled, setFocusable, setFocusTraversalKeysEnabled, setForeground, setIgnoreRepaint, setLocale, setLocation, setLocation, setMaximumSize, setMinimumSize, setName, setPreferredSize, setSize, setSize, setVisible, show, show, size, toString, transferFocus, transferFocusUpCycle

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

GraficoPuntosCursorZoom

public **GraficoPuntosCursorZoom**()

Method Detail

main

public static void **main**(java.lang.String[] args)

Instancia un Lienzo y mete dentro un Seno y una RejillaFija con la escala adecuada. Pone tambi&#65533;n una botoner&#65533;a con zoom de acercar y alejar

---

init

public void **init**()

Overrides:

`init` in class `java.applet.Applet`

Class Parametros

java.lang.Object
  └ **Parametros**

public class **Parametros**
extends java.lang.Object

Constructor Summary

	Parametros()

Method Summary

void	**calcular_parametros**(float minf, float maxf, float f, double r, double s, double w, float n, int N_metales)
double	**redondear**(double numero, int decimales)

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

Parametros

public **Parametros**()

Method Detail

calcular_parametros

public void **calcular_parametros**(float minf,
float maxf,
float f,

```
                        double r,
                        double s,
                        double w,
                        float n,
                        int N_metales)
```

---

redondear

```
public double redondear(double numero,
                        int decimales)
```

Package Class Use Tree Deprecated Index Help

PREV CLASS NEXT CLASS FRAMES NO FRAMES All Classes

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

Class Parametros_1F

java.lang.Object
  └ **Parametros_1F**

public class **Parametros_1F**
extends java.lang.Object

Constructor Summary

Parametros_1F()

Method Summary

void	calculo_parametros_1f(int N_metales, double r, double s, double w, double n, float f)

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

Parametros_1F

public **Parametros_1F**()

Method Detail

calculo_parametros_1f

```
public void calculo_parametros_1f(int N_metales,
                                  double r,
                                  double s,
                                  double w,
                                  double n,
                                  float f)
```

Class ParametrosGlobales

java.lang.Object
  └ **ParametrosGlobales**

public class **ParametrosGlobales**
extends java.lang.Object

Constructor Summary

ParametrosGlobales()

Method Summary

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

ParametrosGlobales

public **ParametrosGlobales**()

Anexo b

Presupuesto

Introducción

Una vez completado el diseño de la herramienta Web y comprobado su correcto funcionamiento, para concluir con el proyecto, en este anexo se realizará un estudio económico con los costes tanto parciales como totales de este.

Baremos utilizados

El cálculo del presupuesto de este proyecto se ha seguido según la **"Propuesta de baremos orientativos para el cálculo de honorarios"** establecida por el Colegio Oficial de Ingenieros Técnicos de Telecomunicación a partir de 1-01-2006 [55].

Esta propuesta establece que para **"Trabajos tarifados por tiempo empleado"** se aplique:

H = Hn · 65 + He · 78

Siendo:

H = Honorarios a percibir.

Hn = Horas contabilizadas en jornada normal.

He = Horas contabilizadas fuera de la jornada normal de trabajo.

Los honorarios que se obtengan por la aplicación de la clave “H” se reducirán a medida que aumente el número de horas, a cuyo efecto serán multiplicados por los coeficientes reductores con arreglo a lo detallado en la Tabla 1.

HORAS			COEFICIENTE
Hasta	36 horas		C=1
Exceso de	36 horas	hasta 72 horas	C=0.9
Exceso de	72 horas	hasta 108 horas	C=0.8
Exceso de	108 horas	hasta 144 horas	C=0.7
Exceso de	144 horas	hasta 180 horas	C=0.65
Exceso de	180 horas	hasta 360 horas	C=0.60
Exceso de	360 horas	hasta 510 horas	C=0.55
Exceso de	510 horas	hasta 720 horas	C=0.50
Exceso de	720 horas	hasta 1080 horas	C=0.45
Exceso de	1.080 horas		C=0.40

Tabla 1 Coeficientes reductores.

Cálculo del presupuesto

1. Costes de debidos a los recursos humanos

En este apartado se incluyen los honorarios a percibir por el ingeniero técnico en el desarrollo del proyecto en función de las horas de trabajo que se ha empleado en la realización del mismo.

Particularizando para el proyecto que aquí se dispone, en la Tabla 2 establecemos unos valores indicativos del tiempo parcial empleado en cada fase del mismo.

DESCRIPCIÓN TIEMPO	PARCIAL (horas)
Búsqueda y estudio de la documentación	300
Estudio de la herramienta de diseño	80
Análisis y diseño del software	500
Realización de la memoria	360

Tabla 2 Tiempo empleado.

En definitiva, se necesitaron un total de 1240 horas para la realización de este proyecto, consideradas en su totalidad del tipo de jornada normal, con lo que el cálculo "H" resulta:

H = 1240 * 65 = 80600

Aplicando los coeficientes correctivos, dados por el COITT, a los tramos correspondientes resultan unos honorarios de:

H = 80600 · 0.40 = 32240 €

2. Costes de amortización de los equipos informáticos y herramientas software

A continuación se detallan, en las Tablas 3 y Tabla 4, los costes relacionados a la utilización de equipos y herramientas software empleados en la elaboración de este proyecto. Los costes están divididos entre el número de usuarios que accedan a estos, los cuales se han estimado en 50 usuarios.

Descripción	Tiempo de uso	Coste anual (€)		Total (€)
		Total	Usuario	
Sistema operativo SunOs Release 4.1.3, Open Windows y aplicaciones X11	11 meses	903	18.06	16.55
Entorno Windows NT	11 meses	306	6.12	5.61
Microsoft Office 2003/07	11 meses	449	8.98	8.23
Mantenimiento	11 meses	1445.31	28.906	26.49
Cadence con Kit de diseño	11 meses	1500	30	27.50
TOTAL				84.38

Tabla 3 Costes debidos a la utilización de herramientas software.

Descripción	Tiempo de uso	Coste anual (€)		Total (€)
		Total	Usuario	
Estación de trabajo SUN Sparc Modelo Sparc Station 10 Amortización 3 años	11 meses	5228.80	104.57	95.85
Mantenimiento	11 meses	1274.65	25.49	23.36
Servidor para simulación SUN Sparc Station 10 Amortización 3 años	11 meses	5068.53	101.37	92.92
Mantenimiento	11 meses	1547.65	30.953	28.37
Impresora Hewlett Packard Laserjet 4L Amortización 3 años	11 meses	360	7.20	6.60
Mantenimiento	11 meses	120.20	2.40	2.20
Ordenador Personal Pentium III 1 GHz Amortización 3 años	11 meses	360	7.20	6.60
Mantenimiento	11 meses	120.20	2.40	2.20
TOTAL				258.10

Tabla 4 Costes debidos a la utilización de equipos informáticos.

3. Otros costes

En este apartado se incluyen los costes debidos al uso de Internet, material fungible y la elaboración del documento final.

Descripción	Unidades	Costes unidad	Total (€)
Horas de uso de Internet	350	1.2 €/hora	450
Paquetes de DIN_A4 80 gr/m^2	3	5 €	15
Fotocopias	1000	0.04 €	40
Otros gastos			100
TOTAL			**575**

Tabla 5 Otros costes.

4. Presupuesto total

Para finalizar en la siguiente tabla se recoge el coste total del proyecto en función de los costes parciales comentados en las secciones anteriores.

Descripción	Gastos (€)
Costes de recursos humanos	32240
Costes de herramientas de software	84.38
Costes de equipos informáticos	258.10
Otros costes	575
PRESUPUESTO FINAL	**33157.48**
TOTAL (I.G.I.C 5%)	**34815.35**

Tabla 6 Presupuesto total.

D. José Cristo González González declara que el proyecto “Desarrollo de una herramienta Web para selección de bobinas integradas” asciende a un total de treinta y cuatro mil ochocientos quince euros con treinta y cinco céntimos.

Fdo. José Cristo González González

DNI: 54084258-B

Las Palmas, a de Junio de 2009

Bibliografia

Referencias

[1] Informe Sociedad de la Información España 2005, http://www.telefonica.es/sociedaddelainformacion.

[2] K. O, "Estimation methods for quality factors of inductors fabricated in silicon integrated circuit process technologies," *IEEE J. Solid-State Circuits,* vol. 33, no.8, pp. 1249–1252, August 1998.

[3] K. B. Ashby, I. A. Koullias, W. C. Finley, J. J. Bastek, and S. Moinian, "High Q inductors for wireless applications in a complementary silicon bipolar process," *IEEE J. Solid-State Circuits,* vol. 31, pp. 4-9, Jan. 1996.

[4] A. Sutono, A. Pham, J. Laskar, and W.R. Smith, "Development of three dimensional ceramic-based MCM inductors for hybrid RF/microwave applications," *in Proc. IEEE RFIC Symposium,* 1999, pp. 175-178.

[5] L. H. Guo, Q. X. Zhang, G. Q. Lo, N. Balasubramanian, and D.-L. Kwong, "High-performance inductors on plastic substrate" *IEEE Trans. Electron Devices Lett.*, vol. 26, pp. 619-621, Sept. 2005.

[6] J.-B. Yoon, B.-K. Kim, C.-H. Han, E. Yoon, and C.-K. Kim, "Surface micromachined solenoid on-Si and on-glass inductors for RF applications," *IEEE Trans. Electron Devices Lett.*, vol. 20, pp. 487-489, Sept. 1999.

[7] X-N. Wang, X-L. Zhao, Y. Zhou, X-H. Dai, and B-C. Cai, "Fabrication and performance of a novel suspended RF spiral inductor," *IEEE Trans. Electron Devices*, vol. 51, pp. 814-816, May 2004.

[8] T. M. Liakopoulos, and C. H. Ahn, “3-D microfabricated toroidal planar inductors with different magnetic core shemes for MEMs and power electronic applications,” *IEEE Trans. Magnetics*, vol. 35, issue 5, part 2, pp. 3679-3681, Sept. 1999.

[9] W. Y. Liu, J. Suryanarayanan, J. Nath, S. Mohammadi, L. P. B. Katehi, and M. B. Steer, “Toroidal inductors for radio-frequency integrated circuits,” *IEEE Trans. Microw. Theory Tech.*, vol. 52, pp. 646-654, Feb. 2004.

[10] M. D. Phillips, and R. K. Settaluri, “A novel toroidal inductor structure with through-hole vias in ground plane,” *IEEE Trans. Microw. Theory Tech.*, vol. 54, pp. 1325-1330, April 2006.

[11] B. Orlando, R. Hida, R. Cuchet, M. Audoin, B. Viala, D. Pellissier-Tanon, X. Gagnard, and P. Ancey, “Low-resistance integrated toroidal inductor for power management,” *IEEE Trans. Magnetics*, vol. 42, pp. 3374-3376, Oct. 2006.

[12] J. Gil, S.-S. Song, H. Lee, and H. Shin, “A 119.2 dBc/Hz at 1 MHz, 1.5 mW, fully integrated, 2.5-GHz, CMOS VCO using helical inductors,” *IEEE Microw. Wireless Compon. Lett.*, vol. 13, pp. 457-459, Nov. 2003.

[13] M.-H. Chang, K.-H. Lin, J.-W. Huang, and A.-K. Chu, “On-chip solenoid inductors with high quality factor for high frequency magnetic integrated circuits,” *IEEE Microw. Wireless Compon. Lett.*, vol. 16, pp. 203-205, April 2006.

[14] J. Craninckx, M.S. J. Steyaert, “A 1.8 GHz Low-Phase-Noise CMOS VCO Using Optimized Hollow Spiral Inductors,” *IEEE Journal of Solid-State-Circuits*, vol. 32, no. 5, pp. 736-744, 1997.

[15] N. M. Nguyen, and R. G. Meyer, “Si IC-compatible inductors and LC passive filters,” *IEEE J. Solid-State Circuits*, vol. 25, pp. 1028-1031, Aug. 1990.

[16] M. Soyuer, J. N. Burghartz, K. A. Jenkins, S. Ponnapalli, J. F. Ewen, and W. E. Pence, “Multilevel monolithic inductors in silicon technology,” *Electronics letters*, vol. 31, no. 5, pp. 359-360 , March 1995.

[17] J. M. López-Villegas, J. Samitier , C. Cané, P. Losantos and J. Bausells, “Improvement of the quality factor of RF integrated inductors by layout optimization,” *IEEE Trans. on Microwave Theory and Techniques*, vol. 48, no. 1, pp. 76-83, Jan. 2000.

[18] K. Kihong, and K. K. O, “Characteristics of an integrated spiral inductor with an underlying n-well,” *IEEE Trans. Electron Devices*, vol. 44, pp. 1565-1567, Sept. 1997.

[19] H. B. Erzgraber, T. Grabolla, H. H. Richter, P. Schley, and A. Wolff, “A novel buried oxide isolation for monolithic RF inductors on silicon,” *in Proc. IEDM*, 1998, pp. 535-539.

[20] L. L. L. Lurng Shehng, L. Chungpin, L. C.-L. L. Chung-Len Lee, H. Tzuen-His, D. D. –L. D. Duan-Lee Tang, D. T. D. Ting Shien, and Y. Tsing-Tyan, “Isolation on Si wafers by MeV proton bombardment for RF integrated circuits,” *IEEE Trans. Electron Devices*, vol. 48, pp. 928-934, May 2001.

[21] N. Choong-Mo, and K. Young-Se, “High-performance planar inductor on thick oxidized porous silicon (OPS) substrate,” *IEEE Microwave Guided Wave Lett.*, vol. 7, pp. 236-238, Aug. 1997.

[22] X. Ya-Hong, M. R. Frei, A. J. Becker, C. A. King, D. Kossives, L. T. Gomez, and S. K. Theiss, “An approach for fabricating high-performance inductors on low-resistivity substrates,” *IEEE J. Solid-State Circuits*, vol. 33, pp. 1433-1438, Sept. 1998.

[23] J. Y.-C. Chang, A. A. Abidi, and M. Gaitan, “Large suspended inductors on silicon and their use in a 2-μm CMOS RF amplifier,” *IEEE Electron Device Lett.*, vol. 14, pp. 246–248, May 1993.

[24] D. Hisamoto, S. Tanaka, T. Tanimoto, Y. Nakamura, and S. Kimura, “Silicon RF devices fabricated by ULSI processes featuring 0.1-/spl mu/m SOI-CMOS and suspended inductors,” *in Dig. Tech. Papers Symp. VLSI Technology*, 1996, pp. 104–105.

[25] Y. Sun, H. van Zejl, J. L. Tauritz, and R. G. F. Baets, “Suspended membrane inductors and capacitors for application in silicon MMIC’s,” *in Dig. Papers IEEE Microwave and Millimeter-Wave Monolithic Circuits Symp.*, 1996, pp. 99–102.

[26] M. Ozgur, M. E. Zaghloul, and M. Gaitan, “High Q backside micromachined CMOS inductors,” *in Proc. IEEE Int. Symp. Circuits and Systems*, 1999, pp. 577–580.

[27] N. P. Pham, K. T. Ng, M. Bartek, P. M. Sarro, B. Rejaei, and J. N. Burghartz, “A micromachining post-process module for RF silicon technology,” *in IEDM Tech. Dig.*, 2000, pp. 481–484.

[28] J. Hongrui, W. Ye, J.-L. A. Yeh, and N. C. Tien, “On-chip spiral inductors suspended over deep copper-lined cavities,” *IEEE Trans. Microwave Theory Tech.*, vol. 48, pp. 2415–2423, Dec. 2000.

[29] H. Lakdawala, X. Zhu, H. Luo, S. Santhanam, L. R. Carley, and G. K. Fedder, “Micromachined high-Q inductors in a 0.18-/spl mu/m copper interconnect low-k dielectric CMOS process,” *IEEE J. Solid-State Circuits*, vol. 37, pp. 394–403, Mar. 2002.

[30] H.-L. Tu, I.-S. Chen, P.-C. Yeh, and H.-K. Chiou, “High performance spiral inductor on deep-trench-mesh silicon substrate,” *IEEE Microw. Wireless Compon. Lett.*, vol. 16, pp. 654-656, Dec 2006.

[31] S-M Yim, T. Chen, and K. K. O, “The effects of a ground shield on the characteristics and performance of spiral inductors,” *IEEE Journal of Solid State Circuits*, vol. 37, no. 2, pp. 237-244, Feb. 2002.

[32] C. P. Yue and S. S. Wong, “On-chip spiral inductors with patterned ground shields for Si-based RF ICs,” *IEEE Journal of Solid State Circuits*, vol. 33, no. 5, pp. 743-752, May 1998.

[33] C. P. Yue and S. S. Wong, “Physical modeling of spiral inductors on silicon,” *IEEE Trans. Electron Devices*, vol. 47, no. 3, pp. 560-568, Mar. 2000.

[34] Y. Cao, R. A. Groves, X. Huang, N. D. Zamdmer, J. Plouchart, R. A. Wachnik, T. King, C. Hu, “Frequency-independent equivalent circuit model for on-chip spiral inductors,” *IEEE Journal of Solid-State-Circuits*, vol. 38, no. 3, pp. 419-426, March 2003.

[35] J. Lee, S. Lee, P. Roblin and S. Bibyk, “Experimental analysis of spiral integrated inductors on low cost integrated circuit processes,” *in Proc. 2005 IEEE SoutheastCon.*, pp.116-120.

[36] J. N. Burghartz and B. Rejaei, “On the design of RF spiral inductors on silicon,” *IEEE Trans. Electron Devices*, vol. 50, no. 3, pp. 718-729, Mar. 2003.

[37] Y. K. Koutsoyannopoulos, and Y. Papananos, “Systematic analysis and modelling of integrated inductors and transformers in RF IC design,” *IEEE Trans. Circuits and Systems II: Analog and Digital Signal Processing*, vol 47, no.8, pp. 699-713, August 2000.

[38] A. Zolfaghari, A. Chan, and B. Razavi, “Stacked inductors and transformers in CMOS technology,” *IEEE Journal of Solid-State-Circuits*, vol. 36, no. 4, pp. 620-628, April 2001.

[39] S. C. Rustagi, and C.-G. Tan, “Equivalent circuit models for stacked spiral inductors,” *in Proc. of IEEE ISCAS* 2003, vol.1, pp. 789-792.

[40] W.Y. Yin, S.J. Pan, L.W. Li, and Y.B. Gan, “Modelling on-chip circular double-spiral stacked inductors for RFICs,” *in Proc. IEE Microw. Antennas Propag.*, 2003, vol. 150, no. 6, pp. 463-469.

[41] C. C. Tang, C. H. Wu, and S. I. Liu, “Miniature 3-D inductors in standard CMOS process,” *IEEE Journal of Solid-State-Circuits*, vol. 37, no. 4, pp. 471-480, April 2002.

[42] F. J. del Pino, “Modelado y aplicaciones de inductores integrados en tecnologías de silicio”, Tesis Doctoral, Universidad de Las Palmas de Gran Canaria, Abril 2002.

[43] J. Van Hese, “Accurate Modeling of Spiral Inductors on Silicon for Wireless RF IC Designs”, *Agilent Technologies*, November 2001.

[44] S. Chaki, S. Aono, N. Andoh, Y. Sasaki, N. Tanino, and O. Ishihara, “Experimental study on spiral inductors”, *in Proc. IEEE MTT-S Int. Microwave Symp. Digest*, 1995, vol. 2, pp. 753-756.

[45] T.H. Lee, “The Design of CMOS RF Integrated Circuits,” Cambridge University Press, pp. 34-57, 1998.

[46] Octavio Medina Day “Diseño de una librería de inductores integrados en tecnología UMC 0.18µm”.

[47] http://www.austriamicrosystems.com

[48] F. Ling, J. Song, T. Kamgaing, Y. Yang, W. Blood, M. Petras, and T. Myers, "Systematic analysis of inductors on silicon using EM simulations," *Electronic Components and Technology Conference*, 2002.

[49] J. del Pino, "Modelado y Aplicaciones de Inductores Integrados en Tecnologías de Silicio", Tesis doctoral, Universidad de Las Palmas de G.C., Abril 2002.

[50] Amaya Goñi Iturri "Aportaciones al diseñp, simulación, caracterización y modelado de inductores integrdos sobre silicio".

[51] A. Zolfaghari, A. Chan, B. Razavi, "Stacked Inductors and Transformers in CMOS Technology", *IEEE Journal of Solid-State Circuits*, vol. 36, no.4, pp. 620-628, Abril 2001.

[52] B. A. Georgescu, "Spiral Inductor. Q-Enhancement Techniques", Department of Electrical and Computer Engineering, University of Calgary, Alberta, Abril 2003.

[53] K. Bohannan, "ADS Momentum, A Half-Day Seminar", Agilent Technologies, Abril 2003.

[54] http://www.cadence.com

[55] "Propuesta de baremos orientativos para el cálculo de honorarios" establecida por el Colegio Oficial de Ingenieros Técnicos de Telecomunicación a partir de 1-01-2006