Vniversitat de València

UNIVERSITAT DE VALÈNCIA

TECNOLOGÍA INFORMÁTICA
(Parte Segunda)

Ingeniería Informática

Fernando Pardo Carpio

Valencia, 8 de febrero de 2000

Prólogo

Los apuntes contenidos en las próximas paginas corresponden a parte de la asignatura de *Tecnología Informática* que se imparte en el segundo curso de la carrera de Ingeniería Informática de la Universidad de Valencia. Se trata de una asignatura optativa y cuatrimestral con un total de 4.5 créditos teóricos y 1.5 prácticos.

El objetivo de la asignatura es familiarizar al alumno con el flujo de diseño de circuitos electrónicos, desde su especificación hasta su realización. Este flujo comienza con la explicación de las principales herramientas y metodologías para la descripción del diseño. Se sigue con la presentación de conceptos sobre simulación tanto digital como eléctrica, y se termina por presentar dos formas en que pueden acabar los diseños electrónicos: circuitos integrados y circuitos impresos. Para cubrir estos objetivos el curso se ha dividido en cuatro materias si bien las dos últimas vienen unidas en una única parte que es la de realización. Estas cuatro materias son:

Lenguajes de descripción hardware En esta materia, que corresponde a la parte de descripción de circuitos, se analizan las diferentes formas de definir y describir circuitos. El tema principal de esta materia es el lenguaje VHDL.

simulación Esta materia cubre los conceptos básicos de simulación y comprobación de circuitos tanto digitales como analógicos.

Microelectrónica Ya en la parte de realización la primera materia es la de microelectrónica donde se explican los procesos de fabricación de circuitos integrados prestando especial atención al proceso CMOS.

Circuitos Impresos Por último se explica el proceso de fabricación de circuitos impresos o PCBs (*Printed Circuit Boards*) revisando las diferentes posibilidades tecnológicas tanto de encapsulados como de tolerancia al ruido, etc.

Los contenidos del curso, es decir, el repaso de todo el flujo de diseño desde la definición del problema hasta su realización practica, son extensos por lo que se ha dado prioridad a unos temas frente a otros.

Considerando los contenidos de otras asignaturas dentro de la carrera, y también las actuales tendencias y demandas de la industria y el diseño hardware, se ha optado por hacer hincapié en los lenguajes de descripción hardware. Por esta razón una gran parte del curso está dedicada al lenguaje VHDL como lenguaje de especificación de circuitos, tanto para síntesis como para la realización de modelos de simulación.

Los apuntes actualmente se encuentran en construcción. Se han completado en su mayor parte los temas de simulación pero aun faltan los de microelectrónica y diseño de circuitos impresos.

Fernando Pardo, en Valencia, febrero del 2000

Índice General

Capítulo 1

Simulación

1.1 Conceptos básicos de simulación

La simulación en una parte de la labor de desarrollo de casi cualquier sistema, no sólo electrónico, sino cualquier tipo de sistema que uno se puede encontrar. La simulación es importante porque permite comprobar un sistema sin tener que implementarlo, y va a ser útil en aquellos sistemas donde la implementación es muy costosa(organización de hospitales) o no es incluso viable (estrategia militar). Algunos sectores donde resulta interesante la simulación son los siguientes:

Hardware y Software: Es lo más cercano al informático. Habitualmente se simulan los computadores o circuitos antes de fabricarlos para ver los problemas que pueden tener. También con los programas se puede hacer algo parecido siempre que la prueba de los mismos en entornos reales suponga un riesgo, o simplemente para probar programas en determinadas circunstancias.

Armas y estrategia militar: Este es un campo muy interesante, ya que permite la comprobación de que determinados sistemas funcionan sin tener que probarlos. El coste en este caso no sólo es económico.

Control de tráfico, trenes, coches, aviones, etc: Este es un campo de cada día mayor aplicación de la simulación, y permite averiguar mejores trazados de las autopistas, determinación de trayectos, zonas de acumulación de tráfico, etc.

Organización de servicios: Normalmente este tipo de simulación se realiza en hospitales o incluso servicio de comidas rápidas donde una planificación del personal es importante. Con la simulación no es necesario movilizar a toda una plantilla para realizar los experimentos.

Sistemas económicos: Con un buen modelo económico se puede realizar una simulación de lo que acontecerá en el futuro. Esto puede servir para sanear la economía de un pais o a, menor escala, la de una empresa.

1.1.1 Sistema, modelo y simulación

Se puede decir que un *Sistema es una colección de entidades que interaccionan entre sí hacia la realización de algún fin lógico.* De forma un poco más práctica podemos

decir que un sistema tiene unas entradas y unas salidas que dependen de las entradas. En general el sistema estará definido por la función que relaciona las entradas y las salidas. Esta función a su vez tendrá una serie de variables internas, que unidas a las entradas y salidas, definen el sistema.

Atendiendo a la estructura interna de los sistemas hay dos tipos de sistemas:

Sistemas discretos: Son sistemas en los cuales las variables de estado cambian instantáneamente en puntos separados del tiempo una cantidad discreta.

Sistemas continuos: Son sistemas en las que las variables de estado pueden cambiar continuamente en el tiempo.

Se puede decir que el *Modelo de un sistema es una descripción del mismo tal que puede ser utilizada para simular el sistema.* En general el modelo estará formado, de una u otra forma, por la función de transferencia del sistema y sus variables de estado.

Existen muchos tipos de modelos dependiendo del criterio que se elija para clasificarlos. Estos son los criterios y los tipos según el cirterio:

Formalismo de la descripción. Según como vengan descritos se tienen los modelos:

- **Físicos:** Son modelos que tienen naturaleza física, como por ejemplo un avión a escala utilizado en un túnel de viento, etc.
- **Matemáticos:** Se tiene una descripción numérica abstracta del comportamiento del sistema.

Tiempo. Según que dependa del tiempo o no se tienen los modelos:

- **Estáticos:** El modelo no depende del tiempo, probablemente porque el sistema tampoco depende del tiempo o la característica que se pretende ver no depende del tiempo. Algunos ejemplos incluyen las simulaciones estadísticas tipo Monte Carlo, o la ley de Ohmm, etc.
- **Dinámicos:** La simulación depende del tiempo. El ejemplo más evidente es un modelo de circuito digital secuencial como pueda ser un procesador, etc. Para estos modelos se suele realizar una simulación donde se muestra la evolución de la salida en función del tiempo.

Estadística. Dependiendo de si el modelo es estadístico o no se tienen los modelos:

- **Estocásticos:** Contienen variables aleatorias en su descripción por lo que el resultado va a depender de estas variables.
- **Deterministas:** Como no poseen variables aleatorias en su descripción el resultado es siempre el mismo para unos datos de entrada.

Sistema. Atendiendo al tipo de sistema se tienen los dos tipos de modelos:

- **Discretos:** el modelo está descrito mediante variables de estados discretas. Normalmente esta simulación se le llama también simulación de eventos ya que los cambios tienen lugar en momentos instantáneos de tiempo. No siempre un modelo discreto se corresponde con un sistema discreto, de hecho los sistemas suelen ser continuos, pero esta discretización simplifica el modelado y simulación. Ejemplo: simulación digital.
- **Continuos:** El modelo se sirve de variables continuas para describir el comportamiento del sistema. Aquí puede ocurrir también que el modelo continuo corresponda a un sistema discreto, un ejemplo de esto es la propia física que describe con leyes continuas lo que en realidad está cuantizado, pero como la resolución de esta cuantización es tan alta, no hay problema en utilizar las ecuaciones analíticas continuas.

1.2 Tipos de simulación

Atendiendo al tipo de modelo de que se dispone se tienen los siguientes tipos de simulación:

1. Simulación Discreta.
2. Simulación Continua.
3. Simulación Discreta-Continua combinada.

Luego estos tres tipos se pueden combinar con lo que se llama *Simulación Monte Carlo* donde se utilizan variables aleatorias para simular el sistema bajo determinadas circunstancias de probabilidad.

1.2.1 Simulación discreta

En la simulación discreta las variables de los modelos cambian de estado instantáneamente en puntos del tiempo separados entre sí. Por lo tanto, entre un cambio y otro de una variable hay siempre un cambio discreto de la magnitud de dicha variable.

Un ejemplo de simulación discreta es la simulación de sistemas digitales donde las variables de estado toman valores discretos (0 y 1 en general), y además lo hacen en instantes de tiempo separados entre sí.

En la simulación discreta conviene introducir el concepto de *evento que es algo que ocurre instantáneamente que puede cambiar el estado del sistema.* Normalmente un evento va a ser el cambio de una variable de estado o una variable de entrada.

Una simulación discreta utiliza los eventos para ir entregando los resultados de la evolución del sistema. En la propia definición de simulación discreta aparece el tiempo, por lo tanto se trata siempre de simulaciones dinámicas donde se quiere ver el resultado de la simulación en función del tiempo.

En la simulación discreta es importante el avance del tiempo. Hay dos formas de implementar este avance de tiempo: por una lado se puede adelantar el tiempo a intervalos fijos y generar la salida a que diera lugar este *avance del tiempo por incremento fijo.* Por otro lado son los propios eventos quienes a su vez generan otros eventos en el futuro y es hacia el próximo evento hacia donde debe avanzar el tiempo, por tanto se produce un avance del tiempo por próximo evento. Ambos tipos se explican a continuación.

Avance del tiempo por incremento fijo

Un simulador discreto basado en avance del tiempo por incremento fijo sigue el siguiente algoritmo para el avance del tiempo:

1. Pone el reloj del sistema a cero ($T = 0$).
2. Si hay algún evento para procesar (inicialmente los eventos a procesar son precisamente los estímulos) se procesa, lo que generará nuevos eventos que tendrán lugar en el futuro.
3. Si aun queda algún elemento para procesar se salta al paso anterior, y si se han acabado los eventos de este tiempo se sigue.

4. Se incrementa el tiempo una cantidad fija ΔT. Si se llega al tiempo final marcado de antemano se detiene la simulación, y si no se salta al paso 2.

Este mismo algoritmo se aprecia esquemáticamente en la figura 1.1. En esta figura se ha introducido un paso, marcado como $\Delta\delta$, en el cual el tiempo no transcurre pero ocurren cosas porque los eventos se están sucediendo. A este $\Delta\delta$ se le suele llamar *paso de simulación.*

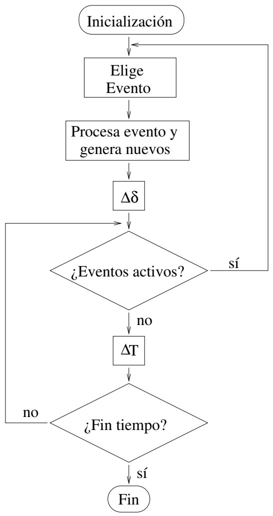

Figura 1.1: Flujo de simulación por incremento fijo

Este tipo de simulación discreta se utiliza sobre en todo en la simulación de sistemas digitales.

Tiene algunos problemas: Por un lado se produce un error en el tiempo ya que las cosas ocurren siempre en múltiplos de ΔT, por lo que si el intervalo es grande el error puede ser importante. Por otro lado, entre T y ΔT se pueden producir varios eventos no necesariamente en el mismo tiempo, en este caso el orden en que se procesan los eventos debe ser el indicado por el tiempo, pero como en este caso todos se supondrán que ocurrieron en ΔT se procesarán en cualquier orden, pudiendo aparecer un problema. También en este caso la elección de un intervalo más pequeño puede ayudar.

A pesar de estos problemas, que no tiene el método que se muestra a continuación, este tipo de simulador es el más empleado, ya que es más rápido y menos costoso de implementar que el otro.

Avance del tiempo por próximo evento

La otra posibilidad de avanzar el tiempo consiste en ver cuál es el próximo evento y avanzar el tiempo hasta él. El algoritmo sería el siguiente:

1. Pone el reloj del sistema a cero ($T = 0$).
2. Si hay algún evento para procesar (inicialmente los eventos a procesar son precisamente los estímulos) se procesa, lo que generará nuevos eventos que tendrán lugar en el futuro.
3. Si aun queda algún elemento para procesar en el tiempo actual se salta al paso anterior, y si no sigue.
4. Se mira a ver cuál es el próximo evento a ejecutar y se avanza el tiempo hasta el indicado por ese evento. Si no hay más eventos en el futuro, o se había indicado un tiempo final y se ha superado, entonces se detiene la simulación.

El flujo de este algoritmo es muy parecido al del tiempo fijo pero tiene algunas diferencias tal y como se muestra en la figura 1.2.

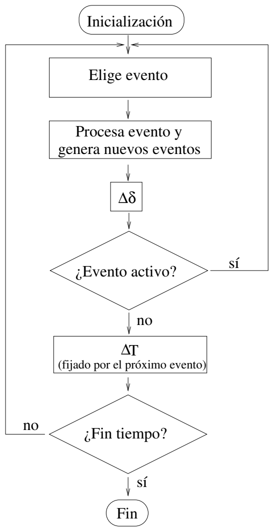

Figura 1.2: Flujo de simulación por próximo evento

Este algoritmo tiene la ventaja de no tener errores con respecto al tiempo, pero tiene el inconveniente de que su ejecución es más lenta por lo que resulta inadecuado para grandes sistemas con miles de eventos en periodos cortos de tiempo.

1.2.2 Simulación continua

Normalmente los sistemas continuos vienen descritos mediante las ecuaciones o fórmulas analíticas donde el resultado es directamente una función de las entradas. Esto resulta fácil de programar en un ordenador.

Muchas veces los sistemas no tienen una fórmula donde se da la salida en función de la entrada, sino que la salida forma parte de una ecuación junto con las entradas y no

se puede despejar. Hay otro caso similar y muy frecuente, sobre todo en modelos que dependen del tiempo, donde la ecuación está expresada de forma diferencial. En ambos casos se requiere de una solución numérica para la que existen diferentes métodos de resolución.

En la mayoría de los casos las soluciones numéricas se dan por aproximaciones sucesivas por lo que no es extraño encontrar algunos errores en este tipo de simulaciones. Los tipos de errores que se pueden dar son dos: por un lado puede que no aproxime lo suficiente por lo que el valor obtenido será parecido al real pero no idéntico, y estos errores pueden acumularse. Por otro lado, hay ocasiones donde los métodos numéricos fallan a la hora de encontrar una solución por problemas de convergencia del algoritmo para unos determinados valores, en estos casos la aproximación numérica puede devolver un valor muy diferente al real. Para evitar estos errores los modelos pueden incorporar parámetros de tolerancia que pueden servir para evitar que se produzcan errores de falta de convergencia o de falta de precisión.

La simulación de sistemas electrónicos, donde intervienen variables continuas como la tensión, corriente, etc., es un ejemplo evidente de simulación continua.

1.2.3 Simulación continua-discreta combinada

Existen muchos sistemas en los que unas partes conviene modelarlas de forma continua y otras partes de forma discreta. Interesa poder simular el sistema completo por lo que el simulador debe ser capaz de manejarse con los dos tipos de modelos comunicándolos entre sí.

Un ejemplo se tiene en los circuitos mixtos analógico-digital, donde se juntan por un lado circuitos digitales, que vienen descritos por modelos discretos, y por el otro por circuitos analógicos que vienen descritos por modelos continuos. Algunos simuladores como el PSpice permiten la simulación combinada de circuitos analógicos y digitales.

Para que un simulador sea capaz de combinar ambos tipos de modelos, debe cumplir las siguientes pautas:

1. Un evento discreto puede provocar un cambio discreto en el valor de una variable continua. Si se toma el ejemplo de un circuito analógico-digital se puede tener un puerta conectada a un condensador en serie con una resistencia conectada a tierra. Cuando la puerta cambia de 0 a 1 se produce un evento discreto que cambia la tensión en bornes del condensador un valor discreto, que en este caso es 5 voltios.
2. Un evento discreto puede cambiar la función que describe el comportamiento de una variable continua. Es decir, en vez de cambiar su valor, cambia la función asociada. Un ejemplo podría ser un filtro de audio que cambia su función de transferencia al cambiar cierta entrada digital, esto por ejemplo sería un equalizador digital.
3. Un variable de estado continua puede causar un evento discreto si sobrepasa un umbral. Por ejemplo, se puede conectar una resistencia y condensador a la entrada de una puerta, de manera que el condensador se va cargando hasta el momento en que se sobrepasa el umbral que a la entrada se entiende como un 1 (por ejemplo 2.5 voltios) momento en el cual se produce un evento a la entrada de la puerta lógica.

1.2.4 Simulación Montecarlo

Muchas veces no se conoce con precisión el valor de las variables que definen un sistema, sino que se conoce más o menos un valor aproximado y un error o tolerancia. En esos casos es muchas veces deseable saber cómo varía la salida en función de estos valores aproximados. Para esto se realiza una simulación estocástica llamada de Montecarlo.

En una simulación de Montecarlo se repite una y otra vez la simulación (discreta, continua, etc.) pero cambiando cada vez los valores de las variables de estado del sistema dentro de sus valores de tolerancia. Con esto se obtiene una distribución de los valores de salida que da idea del valor de salida más probable. Los valores para las variables de estado se cambian de forma aleatoria siguiendo unos patrones que dependerán del tipo de variable de que se trate, es decir, van a seguir una distribución estadística determinada que muchas veces se puede elegir. En la mayoría de los casos se suele utilizar una distribución Gaussiana de los valores, aunque a veces puede interesar una simple distribución plana entre los umbrales de tolerancia de la variable.

1.3 Simulación eléctrica

La simulación eléctrica es una simulación continua, ya que maneja variables continuas como son la tensión, corriente, resistencia, etc. Por tanto no es necesaria la noción de evento a no ser de que se esté realizando la simulación mixta con circuitos digitales y analógicos. En lo que sigue se entenderá que la simulación es puramente eléctrica, también llamada analógica y por tanto continua.

Los simuladores eléctricos deben ser capaces al menos de realizar los siguientes tipos de simulación:

Simulación en continua: Aunque el término es un poco ambiguo se refiere a la simulación que se puede hacer en un circuito cuando no varía ningún valor con el tiempo (todo permanece en *continua*). Esta simulación se utiliza en aquellos casos en los que se quieren ver los parámetros de salida en **función de algún parámetro del sistema**. El ejemplo más típico es el de dar la tensión continua de salida de un circuito en función de la tensión de entrada, pero hay muchos otros como ver la corriente que circula por una resistencia en función de la resistencia o de la temperatura, etc.

Simulación en alterna: Se supone que en el circuito se tienen diversas fuentes de alterna a diferentes frecuencias, entonces lo que se pretende con esta simulación es ver la variación de los parámetros de salida en **función de la frecuencia** de estas fuentes de alterna. El ejemplo típico es ver la respuesta de un filtro pasabanda, pasaaltos o pasabajos, en función de la frecuencia para detectar las diferentes frecuencias de corte. En la de alterna se suele incluir la *simulación de ruido*, ya que éste suele depender de la frecuencia y al fin y al cabo el ruido es una fuente alterna más.

Simulación de transitorios: Consiste en simular el circuito y dar los diferentes parámetros de salida en **función del tiempo**. El ejemplo típico es ver la respuesta de diferentes tipos de señales (cuadrada, sinusoidal, etc.) de un amplificador o incluso de una puerta lógica pero dado su esquema eléctrico.

Existen muchos simuladores eléctricos que realizan estas tareas básicas, pero el que ha ganado mayor popularidad y sobre el que se basan la mayoría de simuladores actuales

es el SPICE.

SPICE viene de *Simulation Program with Integrated Circuit Emphasis* (Programa de simulación con énfasis en los circuitos integrados)

El primer SPICE con la forma que se conoce hoy en día y del cual derivan la mayoría de simuladores actuales, fue el llamado SPICE2 desarrollado en la Universidad de California Berkeley a mediados de los 70. Este simulador provenía del primer SPICE (SPICE1) desarrollado alrededor de 1972 que a su vez era la mejora de otro simulador anterior llamado CANCER. Después de esta breve historia de apenas 5-6 años aparece el SPICE2 que se ha convertido en casi un estándar industrial que muchos fabricantes han incorporado a sus propias herramientas de simulación, lo cual ha sido posible porque este software fue financiado de forma pública y por tanto es de libre distribución. En Berkeley se ha seguido mejorando SPICE y actualmente existe un SPICE3 que probablemente sigue mejorando día a día. Sin embargo, los usuarios no suelen utilizar este software ya que aunque es gratuito no dispone de soporte técnico. Por esta razón numerosas compañías han creado sus propios simuladores siguiendo el casi estándar SPICE2 incluyendo sus propias ampliaciones y adornando el simulador original con numerosas herramientas muy útiles que dan soporte a la simulación.

Las diferencias de unas versiones de SPICE a otras van desde la incorporación de nuevos tipos de análisis y nuevos modelos de dispositivos, hasta las herramientas que permiten hacer de interface para mostrar los resultados, pasando por incluso el ordenador sobre el que pueden ejecutarse. Algunas versiones famosas de SPICE son las siguientes:

PSpice: Es el simulador SPICE para PC más distribuido y conocido. Es bastante compatible con el SPICE2 aunque tiene sus diferencias en cuanto a tipos de análisis (PSpice tiene más) como a sintaxis. Aparte del simulador se distribuyen otras herramientas que sirven para captura de esquemas y para visualizar los resultados (PROBE).

IS-SPICE: Es también un SPICE para PC muy parecido al SPICE2 pero no ha alcanzado la popularidad del PSpice.

HSPICE, IG-SPICE, I-SPICE: Todos estos simuladores son las versiones comerciales de SPICE2 para supercomputadores y workstations. Probablemente el más conocido y utilizado es el HSPICE de Meta-Software, especialmente en la simulación de circuitos integrados CMOS donde la gran disponibilidad de modelos lo ha convertido en el más utilizado. IG-SPICE es de A.B. Associates y I-SPICE es de NCSS timesharing.

AccuSim, Spectre, SPICE-Plus: Estos simuladores, junto con el **PSpice**, forman los simuladores basados en SPICE2 más completos y con más soporte actualmente. Accusim es de Mentor Graphics y es una herramienta para trabajo en workstations, lo mismo le ocurre al Spectre de Cadence. SPICE-Plus es de Valid Logic y PSpice de MicroSim.

1.3.1 Introducción a PSpice y sintaxis

En las explicaciones que siguen se explicará el funcionamiento de PSpice que es un simulador muy parecido al SPICE de Berkeley. Se ha elegido PSpice por ser uno de los simuladores eléctricos basados en SPICE más extendidos actualmente.

El PSpice, al igual que el SPICE, son programas pensados para trabajar por lotes, es decir, cuando no existían ordenadores potentes lo que se solía hacer es lanzar el programa y dejarlo todo el día funcionando hasta que daba un resultado. Esto no es tanto así hoy en día, ya que los ordenadores son cada vez más potentes por lo que el resultado tarda pocos minutos en obtenerse (salvo para circuitos muy complejos donde sigue siendo válido lanzar el programa y esperar horas hasta obtener el resultado).

Como el programa está pensado para funcionar en *background* no hay forma de interaccionar con él. El programa simplemente toma un fichero de entrada (normalmente con la extensión .CIR) donde viene especificado el circuito y los análisis que se quieren hacer, y después de ejecutar PSpice se obtienen ficheros de salida con los resultados de la simulación (normalmente con la extensión .OUT y .DAT).

Con lo que se ha visto resulta que el PSpice carece de una interface con el usuario. Inicialmente el fichero de entrada se programaba a mano, siguiendo una sintaxis sencilla, y los resultados se veían directamente un el fichero .OUT de salida, que era un fichero ASCII con evidentes limitaciones gráficas. Lo más urgente entonces era disponer de una herramienta para poder visualizar los datos de forma interactiva una vez obtenidos. En PSpice la herramienta para hacer esto se llama PROBE y utiliza los datos recogidos en el fichero .DAT para mostrar los resultados por pantalla de forma gráfica e interactiva. Aunque menos imprescindible también resulta disponer de una herramienta de captura de esquemas en vez de programar directamente el circuito en un fichero .CIR. En PSpice esta herramienta se llama Schematics o PSCHED y sirve para describir el circuito en forma de esquema y especificar el análisis que se quiere realizar. Una vez se tiene el esquema se puede generar el fichero .CIR correspondiente que es lo que luego lee el simulador PSpice. Actualmente casi cualquier herramienta de captura de esquemas (Ej. OrCad) permite describir un circuito para luego ser simulado con SPICE.

En esta sección se explica la sintaxis de descripción PSpice, ya que es lo que va a ser común a cualquier simulador eléctrico tipo SPICE. La captura de esquemas o la visualización puede cambiar de un simulador a otro, pero una descripción utilizando la sintaxis de PSpice podrá ser simulada en otros simuladores sin más que hacer pequeños cambios. Se muestra a continuación un ejemplo introductorio a la sintaxis.

Ejemplo de descripción PSpice

Una descripción SPICE tiene dos partes, por un lado se describe el circuito que se quiere simular, sería por tanto una parte puramente de *Netlist*, por otro se describen los análisis y las opciones de simulación. Primeramente se mostrará un ejemplo de descripción del circuito sin entrar en la simulación en sí. Más adelante se explican los tipos de análisis que existen y cómo se especifican mediante el lenguaje.

La figura 1.3 muestra un ejemplo sencillo de circuito. La descripción SPICE de este circuito se muestra a continuación:

```
La primera línea es siempre un comentario
* Las líneas que empiezan por * son comentarios

Vcc 1 0 DC=3
R1  1 2 100
R2  2 0 200
C1  2 0 10u
.END
```

La primera línea de la descripción se entiende que es un comentario, así que no se

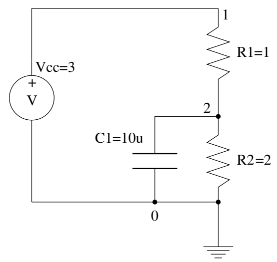

Figura 1.3: Circuito del ejemplo simple con PSpice

debe empezar la descripción del circuito en esta línea. Si más adelante se quieren poner comentarios se indica con un asterisco al inicio de la línea.

El circuito tiene un total de 4 componentes: una fuente de ténsión (`Vcc`), dos resistencias (`R1` y `R2`) y un condensador (`C1`). SPICE distingue unos dispositivos de otros por la primera letra del nombre. De esta manera, cualquier dispositivo que empiece por `V` se entiende que es una fuente de tensión, cualquiera que empiece por `R` se entiende que es una resistencia y si empieza por `C` SPICE entiende que es un condensador. El texto que sigue a la primera letra sirve para distinguir unos dispositivos de otros si son del mismo tipo.

Cada línea de texto en el fichero describe un dispositivo. La interconexión se realiza mediante nodos, dos nodos que tienen el mismo nombre están interconectados. Normalmente los nodos son el primer parámetro de la descripción de los dispositivos. Actualmente los nodos pueden tener cualquier nombre, pero en el SPICE original sólo se permitían números como nombre de los nodos.

El primer dispositivo que se describe es la fuente de tensión `Vcc`. Los dos números que siguen son los nodos a los que está conectada y son los nodos `1` y `0`. El nodo `0` es un nodo especial que en SPICE significa tierra, masa o *ground*, y es el nodo de referencia de todos los demás. Cualquier circuito en SPICE debe tener siempre un nodo que se llame `0`. El último parámetro es la tensión que este caso es continua e igual a 3 Voltios. SPICE siempre utiliza el sistema internacional para las unidades.

Los dos dispositivos siguientes son las resistencias. La primera está conectada entre el nodo `1` y `2`, y la segunda entre el `2` y el `0`. El tercer parámetro es la resistencia en Ohmios. El último dispositivo es el condensador conectado entre el nodo `2` y el `0`, que tiene una capacidad de 10 μF. La `u` en un número indica que viene multiplicada ese número por el factor 10^{-6}.

Finalmente comentar que SPICE no distingue entre mayúsculas y minúsculas. A lo largo del texto se intentará poner los dispositivos e instrucciones en mayúsculas y los parámetros y nodos en minúsculas.

Unidades y magnitudes eléctricas

Los números en SPICE se pueden expresar como en cualquier lenguaje de programación utilizando la notación exponencial (Ej. 1.05e-6) o el punto decimal (Ej. 21.4502). Todos los números se entiende que son reales aunque no lleven el punto decimal.

Unidad	Sufijo	Factor
Femto	f	10^{-15}
Pico	p	10^{-12}
Nano	n	10^{-9}
Micro	u	10^{-6}
Mili	m	10^{-3}

Unidad	Sufijo	Factor
Kilo	K	10^{3}
Mega	MEG	10^{6}
Giga	G	10^{9}
Tera	T	10^{12}
Metric	MIL	$25.4 \cdot 10^{-6}$

Tabla 1.1: Sufijos de SPICE y su factor correspondiente

Una particularidad que presenta SPICE frente a otras convenciones a la hora de expresar números, es la posibilidad de incluir un sufijo junto con el número. Este sufijo va a ser un factor que multiplique a la cantidad numérica que le precede. En la tabla 1.1 se muestran estos sufijos y el factor a que equivalen. Algunos se han puesto en mayúsculas y otros en minúsculas, ya que existe la convención de poner en mayúsculas los factores con exponente positivo y en minúsculas los que tienen exponente negativo, por supuesto se pueden poner como se quieran.

El sufijo `MIL` se utiliza para la conversión de metros a pulgadas. Es muy frecuente al principio cometer el error de poner `1M` para indicar 10^6, cuando en realidad es lo mismo que `1m` que es 10^{-6}. Hay que recordar que es SPICE no se distinguen las mayúsculas de las minúsculas.

En cuanto a las magnitudes eléctricas no es necesario indicarlas de ninguna manera. SPICE utiliza siempre las unidades del sistema internacional, si hay una resistencia el parámetro se entenderá que está en Ohmios, si es una fuente de tensión el voltaje estará en voltios, la capacidad de los condensadores viene en faradios, la inducción de las bobinas en henrios, la frecuencia en herzios, etc.

En SPICE sin embargo se pueden poner letras después del número y no ocurre nada, siempre naturalmente que el texto que sigue al número no sea uno de los sufijos predefinidos. Así un `3`, es lo mismo que `3V` o `3A`, etc., pero no será lo mismo que `3F` debido a que `F` es uno de los sufijos predefinidos. A un número que le siga un sufijo predefinido también se le puede añadir texto, así por ejemplo es normal encontrar magnitudes como `3mV`, `4 uAmp` que son lo mismo que `3e-3` y `4e-6` respectivamente. El texto después de un número (a no ser de que sea un sufijo predefinido) es meramente informativo y en ningún caso cambia la magnitud eléctrica del número.

1.3.2 Descripción de componentes pasivos

Los componentes pasivos que se pueden describir usando SPICE son las resistencias, condensadores, bonbinas, líneas de transmisión, inductores acoplados y líneas de transmisión acopladas.

Descripción de resistencias (R)

Los elementos más sencillos para describir en PSpice son los componentes pasivos, es decir, resistencias, condensadores y bobinas.

Para indicar la sintaxis de cada cosa se utilizará la convención de que lo que vaya entres corchetes `[]` es opcional y lo que vaya entre desigualdades ¡¿ son parámetros que

se deben poner pero que decide el usuario su contenido. Se muestra a continuación la sintaxis de la resistencia para PSpice:

```
R<nombre> <n+> <n-> [<nombre_modelo>] <valor> [TC=<TC1>[,<TC2>]]
```

El nombre de la resistencia puede ser cualquiera y sirve para distinguir unas resistencias de otras. Los dos parámetros siguientes son los nodos, el primero el nodo positivo y el segundo el negativo. Normalmente las resistencias no tienen polaridad así que en este caso da igual el orden en que se pongan.

El `nombre_modelo` sirve para hacer referencia a un modelo de transistor en una biblioteca de componentes. También sirve para crear un modelo propio mediante la instrucción `.MODEL`:

```
.MODEL <nombre_modelo> RES [(parámetros)]
```

Los parámetros opcionales del final tanto del modelo como de la resistencia sirven para modelar con mayor precisión el comportamiento de la resistencia. Esto se explica a continuación.

PSpice utiliza la siguiente fórmula para calcular la resistencia real del componente. Se han puesto en mayúsculas los parámetros que se pueden poner en el modelo:

$$Resistencia = valor \cdot R \cdot \left(1 + TC1 \cdot (Temp - Tnom) + TC2 \cdot (Temp - Tnom)^2\right)$$

donde $Tnom$ es el parámetro `TNOM` que se puede fijar en PSpice con la instrucción `.OPTIONS`. $Temp$ es un parámetro que se fija con la instrucción de PSpice `.TEMP`.

Si en el modelo se especifica un parámetro llamado `TCE` entonces la fórmula es diferente:

$$Resistencia = valor \cdot R \cdot 1.01^{TCE \cdot (T - Tnom)}$$

Ejemplos:

```
R1 1 0 23K
R2 alto bajo TC=0.01
Rsalida sal ent 2MEG TC=0.01,-0.002
Rmod 45 54 RMOD1 10K
.MODEL RMOD1 RES (TC1=0.01 TC2=-0.02)
```

Descripción de condensadores (C)

La forma general del condensador es la siguiente:

```
C<nombre> <n+> <n-> [<nombre_modelo>] <valor> [IC=<tensión_inicial>]
```

El formato es muy similar al de la resistencia, pero en vez de tener los parámetros que dependen de la temperatura tiene las condiciones iniciales. Esto es importante porque al hacer la simulación de transitorios (función del tiempo) las condiciones iniciales son importantes. Estas condiciones iniciales se pueden indicar en los componentes o en una instrucción aparte llamada `.IC`.

El formato de la instrucción del modelo para los condensadores es la siguiente:

```
.MODEL <nombre_modelo> CAP [(parámetros)]
```

Los parámetros son los que aparecen en mayúscula en la siguiente fórmula que es también la que utiliza PSpice para saber la capacidad real en función de la tensión y temperatura del condensador:

$$Capacidad = valor \cdot C \cdot (1 + VC1 \cdot v + VC2 \cdot v^2) \cdot \cdot (1 + TC1 \cdot (Temp - Tnom) + TC2 \cdot (Temp - Tnom)^2)$$

El parámetro v en este caso es la tensión a la que se encuentra el condensador en un instante dado. Los parámetros $Tnom$ y $Temp$ son equivalentes a los mostrados en la resistencia.

Ejemplos:

```
C1 pos neg 10uF IC=5v
C2 4 6 CMOD1 20nF
.MODEL CMOD1 CAP (VC1=0.1 TC1=.02)
```

Descripción de bobinas (L)

La bobina también es un dispositivo con una fuerte componente temporal por lo que al igual que el condensador se debe poder especificar las condiciones iniciales. La sintaxis para describir una bobina es como sigue:

```
<nombre> <n+> <n-> [<nombre_modelo>] <valor> [IC=<corriente_inicial>]
```

El formato de la instrucción del modelo para las bobinas es la siguiente:

```
.MODEL <nombre_modelo> IND [(parámetros)]
```

Los parámetros que se pueden especificar en el modelo son los que aparecen en mayúsculas en la expresión de la inductancia real de la bobina:

$$Inductancia = valor \cdot L \cdot (1 + IL1 \cdot i + IL2 \cdot i^2) \cdot \cdot (1 + TC1 \cdot (Temp - Tnom) + TC2 \cdot (Temp - Tnom)^2)$$

En este caso lo que influye en la bobina es la corriente i que la atraviesa en un instante dado y la temperatura como en cualquier dispositivo.

Líneas de transmisión (T)

Una línea de transmisión es un dispositivo pasivo especial que pretende modelar la transmisión de información eléctrica a través de un medio. En general esto constituye un dispositivo complejo que necesita de varios parámetros para ser descrito. Dependiendo de si la línea es ideal o tiene pérdidas se tienen dos posibilidades en la descripción. A continuación se muestra la **línea ideal**:

```
T<nombre> <n1+> <n1-> <n2+> <n2-> [<nombre_modelo>] Z0=<valor>
+[TD=<valor>][F=<valor> [NL=<valor>]]
```

En primer lugar el signo `+` no es propia de la instrucción sino que sirve para indicarle a SPICE que la línea anterior continua aunque en realidad se trata de dos líneas, es la forma que tiene SPICE de poder especificar un mismo dispositivo en más de una línea de texto.

Las líneas de transmisión tienen cuatro terminales o nodos, dos en un lado de la línea y dos en el otro. El parámetro `Z0` es el único obligatorio en la línea de transmisión

ideal y sirve para especificar la impedancia característica de la línea, tiene por tanto dimensiones de resistencia expresada en Ohmios. El resto de parámetros sirven para especificar la longitud de la línea. Esto se puede hacer, bien dando el retardo de la línea en segundos con `TD`, o bien, dando `F` y `NL` que son una frecuencia y la longitud de onda relativa a esa frecuencia.

`NL` por defecto vale 1/4 de onda. Aunque `TD` y `F` aparecen como opcionales, al menos se debe especificar uno de ellos.

Si se quiere especificar una línea de transmisión que tiene pérdidas entonces hay que recurrir a la siguiente sintaxis específica para la **linea con pérdidas**:

```
T<nombre> <n1+> <n1-> <n2+> <n2-> [<nombre_modelo>[longitud_eléctrica]]
+LEN=<valor> R=<valor> L=<valor> C=<valor> G=<valor>
```

Con esta sintaxis se utiliza un modelos para la línea de transmisión que utiliza las características pasivas del mismo. Los parámetros `LEN`, `R`, `L`, `C` y `G` son respectivamente la longitud expresada en metros, la resistencia expresada en Ohmios/metro, la inductancia en Henrios/metro, la capacidad en Faradios/metro y finalmente la conductancia en mhos/metro.

Tanto si la línea es ideal o tiene pérdidas la especificación del modelo es común para los dos casos:

```
.MODEL <nombre_modelo> TRN [(parámetros)]
```

Los parámetros que se pueden especificar en el modelo son los mismos que pueden aparecer en la especificación de la línea de transmisión.

Ejemplos:

```
T1 1 2 3 4 Z0=50 TD=200ns
T2 a1 a2 b1 b2 Z0=5MEGHz NL=1
T3 1 2 3 4 LEN=1 R=0.2 L=0.3uH C=23pF G=1u
T4 1 2 3 4 TRASMOD 1
.MODEL TRASMOD TRN R=0.2 L=0.3uH C=23pF G=1u
```

Inductores o líneas de transmisión acopladas (K)

Un dispositivo pasivo típico es el transformador que se trata en realidad de dos inductores acoplados. Si en vez de bobinas el acoplamiento se realiza a través de dos líneas de transmisión entonces se tiene un modelo algo más complejo pero que también está incluido en PSpice. La descripción de **inductores acoplados** es la siguiente:

```
K<nombre> L<nombre> <L<nombre>>* <coeficiente_acoplo>
+[<nombre_modelo> [factor_tamaño]]
```

De estas dos formas se puede describir el acoplo de dos o más inductores (el `*` indica que se puede repetir una o más veces lo que está entre las desigualdades). El `coeficiente_acoplo` da idea del acoplamiento mutuo de las bobinas entre sí. Este coeficiente es cualquier valor entre (0,1], tomando el valor 1 en transformadores con núcleo de ferrita y geometrías convencionales, y valores menores de 1 en inductores acoplados por aire o alejados.

Si se desea un mejor modelado se puede recurrir al modelo donde se dispone de varios parámetros para modelar las numerosas no linearidades que aparen en un dispositivo de este tipo.

Si la descripción que se quiere hacer es la de dos **líneas de transmisión acopladas** entonces la sintaxis sería la siguiente:

```
K<nombre> T<nombre> T<nombre>
+Cm=<capacidad_acoplo> Lm=<inducción_acoplo>
```

Esta descripción no permite la especificación de un modelo aparte, ya que `Cm` y `Lm` bastan para modelar el acoplo entre las líneas de transmisión.

El acoplo entre inductores sí que tiene la posibilidad de especificar un modelo bastante completo del acoplamiento. La sintaxis de la especificación del modelo es la siguiente:

```
.MODEL <nombre_modelo> CORE [(parámetros=valor)]
```

Ejemplos:

```
Kaire L1 L2 0.89
Ktrafo L1 L2 1
K3 L1 L2 L3 0.98 nucleo3
.MODEL nucleo3 CORE (LEVEL=2 ALPHA=0 MS=400K C=0.5)

Ktrans T1 T2 Cm=0.4pF Lm=120uH
```

1.3.3 Descripción de componentes semiconductores

En SPICE se dispone de modelos para los siguientes componentes semiconductores: diodos, transistores bipolares, transistores FET de unión, transistores de arseniuro de galio (GaAsFET), y transistores MOSFET.

Dada la complejidad de estos dispositivos existen modelos bastante completos de cada uno de los dispositivos. Incluso algunos fabricantes diseñan sus propios modelos, ya que la teoría subyacente a cada dispositivo resulta a veces insuficiente.

Diodos (`D`)

El diodo es el elemento semiconductor más simple de todos, tiene dos terminales ánodo y cátodo, y su característica básica es que deja pasar la corriente sólo en un sentido. La sintaxis de descripción de un diodo es la siguiente:

```
D<nombre> <na> <nk> <nombre_modelo> [<área>]
```

Los nodos `na` y `nk` son el ánodo y cátodo del diodo respectivamente. Al contrario que en los dispositivos vistos hasta ahora en los semiconductores siempre se debe especificar el nombre de un modelo. Como parámetro opcional está el área de la unión que se trata en realidad de un factor de escala y no el área real en m^2, su valor por defecto es 1.

Para modelar el diodo hay que añadir siempre una línea de modelo que para el diodo tiene la sintaxis:

```
.MODEL <nombre_modelo> D [(parámetros)]
```

Existen alrededor de unos 25 parámetros que modifican el comportamiento del diodo. Todos ellos tienen unos valores por defecto que modelizan al diodo casi como un dispositivo real. Se pueden modificar algunos de estos parámetros para darles características más realistas.

Ejemplos:


```
D1 1 2 Dsimple
.MODEL Dsimple D
D2 1 2 Dmod 2.0
.MODEL Dmod D (IS=1uA RS=2 CJO=2pF)
```

Transistor bipolar (`Q`)

Transistores bipolares hay de dos tipos: NPN y PNP, dependiendo de si la base es P o N. La descripción es igual para los dos:

```
Q<nombre> <nc> <nb> <ne> [<ns>] <nombre_modelo> [<área>]
```

El dispositivo tiene tres nodos que hay que especificar siempre que son el colector `nc`, la base `nb` y el emisor `ne`. Existe la opción de poner un cuarto nodo que hace de substrato `ns` que no es necesario normalmente en los transistores discretos pero que puede ser imprescindible en los transistores de un circuito integrado. Por último se especifica obligatoriamente el nombre del modelo y opcionalmente el factor de escala que por defecto es 1.

El modelo debe tener en cuenta si el transistor es NPN o PNP, per además existen los transistores laterales LPNP que tienen su modelos propio. En total hay tres:

```
.MODEL <nombre_modelo> NPN [(parámetros)]
.MODEL <nombre_modelo> PNP [(parámetros)]
.MODEL <nombre_modelo> LPNP [(parámetros)]
```

El modelo de SPICE para el transistor bipolar permite modificar unos 55 parámetros que definen el comportamiento del transistor. Normalmente el transistor se utiliza como un elemento discreto de diseño, por lo que suele existir una biblioteca donde viene el transistor que se esté utilizando.

Ejemplos:

```
Qtipico 1 2 3 Q2N2222A
Q1 c b e trmod1 2.0
Q2 1 2 3 sub trmod2
.MODEL trmod1 NPN
.MODEL trmod2 PNP (CJE=5pF CJC=6pF)
```

Transistor de unión JFET (`J`)

El transistor JFET (*Junction Field Effect Transistor*) es un transistor de unión de efecto de campo. Normalmente suele tratarse de componentes discretos que no forman parte de un circuito integrado. La descripción es la siguiente:

```
J<nombre> <nd> <ng> <ns> <nombre_modelo> [<área>]
```

Un transistor de unión de efecto de campo tiene tres terminales: el drenador `nd` (*drain*), la puerta `ng` (*gate*), y la fuente o surtidor `ns` (*source*). El nombre del modelo es también obligatorio y se puede poner opcionalmente un factor de escala.

Como hay dos tipos de transistores JFET, dependiendo del tipo de semiconductor aparece al crearse el canal en la unión, hay dos formas de poner el modelo:

```
.MODEL <nombre_modelo> NJF [(parámetros)]
.MODEL <nombre_modelo> PJF [(parámetros)]
```

El primero de ellos es lo que se llama JFET de canal N y el segundo es el de canal P. Como SPICE estaba orientado desde un principio a la simulación de circuitos integrados, los modelos de los elementos discretos no son tan sofisticados como aquellos susceptibles de integrarse. Por esta razón, aunque el modelo para el JFET es bastante completo, su modelo sólo tiene unos 21 parámetros que definen su comportamiento.

Ejemplos:

```
J1 1 2 3 Jmod1 3.0
J2 4 5 6 Jmod2
.MODEL Jmod1 NJF
.MODEL Jmod2 PJF (BETA=.3 VTO=-6)
```

Transistores FET de Arseniuro de Galio (`B`)

En los últimos años se ha desarrollado mucho la tecnología de fabricación de circuitos integrados utilizando Arseniuro de Galio (AsGa) en lugar de silicio. Las ventajas de estos circuitos son muchas pero su alto precio sigue siendo prohibitivo. No obstante, y como cada vez se utilizan más, PSpice ha incorporado el modelo de estos transistores. La sintaxis es la siguiente:

```
B<nombre> <nd> <ng> <ns> <nombre_modelo> [<área>]
```

Como se trata de un transistor de efecto de campo sus nodos son el drenador, puerta y surtidor. La especificación del modelo es:

```
.MODEL <nombre_modelo> GASFET [(parámetros)]
```

Ejemplos:

```
B1 1 2 3 Ga1 4.0
B2 1 2 3 Ga2
.MODEL Ga1 GASFET
.MODEL Ga2 GASFET (VTO=-3 BETA=1m)
```

Transistores MOSFET (`M`)

El transistor más utilizado actualmente para la fabricación de circuitos integrados es el transistor MOSFET (*Metal Oxide Semiconductor Field Effect Transistor*). Al mismo tiempo es también uno de los dispositivos semiconductores más difíciles de modelar dada su complejidad teórica. La forma de especificar un transistor MOSFET (conocidos habitualmente como MOS) es la siguiente:

```
M<nombre> <nd> <ng> <ns> <nb> <nombre_modelo> [L=<largo>] [W=<ancho>]
[AD=<valor>] [PD=<valor>] [NRD=<valor>] [NRG=<valor>]
[AS=<valor>] [PS=<valor>] [NRS=<valor>] [NRB=<valor>] [M=<valor>]
```

El transistor MOS, como es un transistor que siempre va en un circuito integrado, necesita cuatro terminales, las tres típicas de los transistores de efecto de campo (drenador, puerta y surtidor) la de substrato `nb` (*bulk*). El nombre del modelo es también obligatorio.

Los parámetros opcionales `L` y `W` son respectivamente la longitud y anchura del canal y son los parámetros más importantes del transistor, ya que escalan todas las características del transistor. Los parámetros `AD`, `PD`, `AS` y `PS` son las áreas y perímetros del drenador y surtidor. Los parámetros `NRD`, `NRG`, `NRS` y `NRB` son las resistividades,

relativas dadas en Ohms/cuadrado del drenador, puerta, surtidor y substrato. Por último el factor `M` sirve para simular `M` transistores puestos en paralelo, por defecto vale 1.

Los modelos de los transistores MOS se especifican mediante dos instrucciones según se trate de un transistor de canal N (NMOS) o canal P (PMOS):

```
.MODEL <nombre_modelo> NMOS [(parámetros)]
.MODEL <nombre_modelo> PMOS [(parámetros)]
```

En principio se podría pensar que sólo hay dos modelos para el transistor MOS, pero esto no es así, hay muchos más. En concreto PSpice soporta 4 modelos para cada tipo de transistor. El modelo que se desea utilizar se especifica mediante el parámetro `LEVEL` que puede tomar los siguientes valores:

`LEVEL=1` es el modelo de Shichman-Hodges. Es un modelo teórico sencillo del comportamiento del transistor. Resulta conveniente cuando no se requiere mucha precisión, o se precisa de gran rapidez en la obtención de los resultados de simulación, o se tienen problemas de convergencia. No resulta adecuado si se quiere simular el transistor trabajando en su zona de inversión débil o con transistores de canal pequeño.

`LEVEL=2` es otro modelo teórico mucho más avanzado con el que se obtienen resultados precisos incluso con canales moderadamente pequeños o trabajando en la zona de inversión débil. Tarda más en ejecutarse y a veces tiene problemas de convergencia.

`LEVEL=3` es un modelo semi-empírico para canal corto. Representa una mejora respecto al anterior tanto en tiempo de ejecución como en convergencia siendo de similar precisión aunque mejora la respuesta en transistores de canal pequeño. El problema que puede tener es que algunos parámetros se extraen de forma empírica, mientras que en los otros se puede tener el modelo sin necesidad de medir.

`LEVEL=4` es el modelo BSIM (*Berkeley Short-Channel IGFET Model*). Es el modelo más completo que incluye unos 60 parámetros. Da buenos resultados para geometrías de transistor inferiores a las soportadas por el resto de modelos siendo también más preciso. Es también un modelo empírico por lo que la mayoría de sus parámetros son en realidad las medidas que se hacen en el proceso de caracterización de un determinado proceso de fabricación de circuitos integrados. Este modelo no se puede utilizar si no se conocen todos los parámetros, ya que carece de valores por defecto.

Aparte de estos modelos existen muchos otros que no están soportados por PSpice pero sí por otras herramientas de simulación. El número de modelos crece cada año al tiempo que evoluciona el proceso de fabricación de los circuitos integrados. Por poner un ejemplo, el nivel 4 representa el mejor modelo que tiene PSpice, sin embargo no resulta apropiado para transistores con un canal con una longitud por debajo de 1 μm cuando hoy en día la tecnología permite 0.15 μm que es casi 10 veces menor.

Ejemplos:

```
Mn out in 0 0 Nmod L=1u W=50u
Mp out in vdd vdd Pmod L=1u W=100u
.MODEL Nmod NMOS (LEVEL=3 VTO=5v)
.MODEL Pmod PMOS
```

1.3.4 Descripción de fuentes

Un elemento importante en cualquier circuito electrónico son las fuentes de alimentación y señal. Las fuentes más sencillas son las independientes que generan una tensión o corriente generada internamente. Luego están las fuentes cuya salida depende de la entrada y son bastante más complejas permitiendo poder ser empleadas en el modelado de dispositivos no presentes en SPICE.

Fuentes independientes de tensión y corriente

Una fuente independiente genera una tensión o corriente a la salida siguiendo un patrón determinado. Si la especificación de la fuente empieza por `V` se entenderá que es de tensión, y si empieza por `I` de corriente. Esta es la sintaxis de la fuente de tensión independiente:

```
V<nombre> <n+> <n-> [[DC] <valor>] [AC <magnitud>[<fase>]]
+[PULSE (<v1> <v2> [ <td> [ <tr> [ <tf> [ <pw> [ <per> ]]]]] )]
+[SIN (<vo> <va> [ <freq> [ <td> [ <df> [ <fase> ]]]] )]
+[EXP (<v1> <v2> [ <td1> [ <t1> [ <td2> [ <t2> ]]]] )]
+[SFFM (<vo> <va> [ <fc> [ <mdi> [ <fm> ]]] )]
+[PWL (<t1> <v1> ...  <tn> <vn>)]
```

La fuente independiente de corriente es igual que la de tensión pero empieza con una I. Además los parámetros de tensión son de corriente:

```
I<nombre> <n+> <n-> [[DC] <valor>] [AC <magnitud> [<fase>]]
+[PULSE (<i1> <i2> [ <td> [ <tr> [ <tf> [ <pw> [ <per> ]]]]] )]
+[SIN (<io> <ia> [ <freq> [ <td> [ <df> [ <fase> ]]]] )]
+[EXP (<i1> <i2> [ <td1> [ <t1> [ <td2> [ <t2> ]]]] )]
+[SFFM (<io> <ia> [ <fc> [ <mdi> [ <fm> ]]] )]
+[PWL (<t1> <i1> ...  <tn> <in>)]
```

La corriente positiva va del nodo `n+` al nodo `n-` a través de la fuente de corriente, esto quiere decir que externamente por el nodo `n+` entra corriente y por el `n-` sale, por lo tanto, la tensión de `n-` puede ser positiva con respecto a la de `n+`.

A partir de ahora se explica la fuente de tensión siendo todo válido para la fuente de corriente sin más que cambiar tensión por corriente.

En una fuente de tensión se tienen tres partes bien diferenciadas. La que aparece en primer lugar es la especificación de la tensión continua y consiste en poner simplemente un valor de tensión o añadir `DC` con un valor. Este parámetro sólo tiene sentido si se está haciendo un **análisis de continua**, es decir, se pretenden ver unos parámetros en función de otros parámetros.

El otro parámetro que se puede especificar es la componente de alterna con `AC` y magnitud que correspondería a la amplitud de una onda sinusoidal. Opcionalmente se puede especificar una fase. La especificación de la componente alterna sólo tiene sentido en **análisis de alterna**, es decir, en simulaciones donde se pretenden ver algunos parámetros en función de la frecuencia.

Lo último que se puede especificar es la componente temporal que sólo sirve para el **análisis de transitorios**, es decir, en simulaciones donde se quiere ver cómo evolucionan los parámetros del sistema en función del tiempo. Al contrario que en el

resto de análisis, en el de transitorios se pueden especificar varias señales, desde ondas sinusoidales hasta pulsos. Sólo una de estas formas de onda se puede especificar en la parte de transitorios.

En PSpice se pueden realizar al mismo tiempo los tres tipos de análisis: continua, alterna y transitorios. Esto quiere decir que se pueden especificar los tres tipos de magnitudes en la fuente independiente, siendo bastante frecuente encontrar una definición de fuente con una componente de continua, otra de alterna, y una tercera de transitorios (un pulso por ejemplo). Normalmente las magnitudes de un tipo no influyen en los análisis que no sean de su tipo, así por ejemplo un análisis de transitorios no tendrá en cuenta el valor de continua o alterna que se hubiera especificado en la fuente. Sólo hay una excepción a esto y es la componente de continua que sí tiene un efecto importante sobre la simulación en alterna. En efecto, si en la simulación de alterna no se pone componente de continua se supondrá que la señal alterna está superpuesta a una señal continua de cero voltios, pero si se especifica una componente de continua entonces la señal alterna viene superpuesta a esta señal continua. Por decirlo de alguna manera la señal continua fija el punto de operación sobre el cual se superpone la señal de alterna, y esto es muy importante.

El transitorio se puede especificar de varias maneras dependiendo de que lo que se quiera sea una onda sinusoidal, un tren de pulsos, una onda sinusoidal modulada en frecuencia o una onda cualquiera modelada con segmentos. Sólo una de estas formas de onda puede ser especificada a un tiempo. A continuación se muestra el significado de los parámetros de cada forma de onda.

```
PULSE (<v1> <v2> [ <td> [ <tr> [ <tf> [ <pw> [ <per> ]]]]] )
```

Esto sirve para especificar un tren de pulsos. Los únicos parámetros obligatorios son la tensión inicial `v1` y la tensión secundaria `v2`. Por lo tanto esta especificación sirve para describir pulsos que cambian entre `v1` y `v2`. El resto de parámetros son opcionales y tienen el siguiente significado: `td` especifica el retraso del tren de pulsos, `tr` y `tf` son los tiempos de subida y bajada del pulso, no deben ser cero puesto entonces se tiene una singularidad y la simulación no converge en esos puntos, `pw` es la anchura del pulso a la tensión `v2`, y por último `per` es el periodo del tren de pulsos. Los valores por defecto que tomas estos parámetros son función de la duración total de la simulación, y se calculan de manera que aparezca un único pulso en toda la duración de la simulación.

```
SIN (<vo> <va> [ <freq> [ <td> [ <df> [ <fase> ]]]] )
```

Esta es la forma en que se define una onda sinusoidal. Los parámetros `vo` y tt va son los únicos obligatorios y definen la componente continua de la onda (*offset*) y la amplitud del seno respectivamente. El resto de parámetros son opcionales y tienen el siguiente significado: `freq` especifica la frecuencia del seno, `td` es el retraso antes de que empiece la onda sinusoidal, `df` es un factor de decaimiento en que caso de que se quiera un seno que se va atenuando, y por último `fase` es la fase de la onda. Los parámetros por defecto toman unos valores tales que se define un ciclo completo durante la duración de la simulación. La función que se utiliza para calcular el seno en función de los parámetros vistos es la siguiente:

$$V(t) = \begin{cases} vo + va \cdot \sin\left(2\pi + \frac{fase}{360}\right) & 0 \leq t < td \\ vo + va \cdot \sin\left(2\pi\left(freq(t-td) + \frac{fase}{360}\right)\right) \mathrm{e}^{-(t-td)\cdot df} & td \leq t \end{cases}$$

```
EXP (<v1> <v2> [ <td1> [ <tf1> [ <td2> [ <tf2> ]]]] )
```

Con la instrucción anterior se especifica una forma de onda exponencial. Los parámetros obligatorios son `v1` y `v2` que definen la tensión inicial y secundaria de la onda (normalmente tensión baja y alta). A partir del tiempo de retraso `td1` la tensión cambia de `v1` a `v2` de forma exponencial con un factor temporal `tf1`, y a partir del retraso `td2` la tensión cambia entre `v2` y `v1` de forma exponencial con un factor `tf2`. Es decir, la tensión toma los valores descritos en la siguiente ecuación:

$$V(t) = \begin{cases} v1 & 0 \leq t < td1 \\ v1 + (v2 - v1)\left(1 - \mathrm{e}^{\frac{t-td1}{tc1}}\right) & td1 \leq t < td2 \\ v1 + (v2 - v1)\left(\left(1 - \mathrm{e}^{\frac{t-td1}{tc1}}\right) - \left(1 - \mathrm{e}^{\frac{t-td2}{tc2}}\right)\right) & td2 \leq t \end{cases}$$

```
SFFM (<vo> <va> [ <fc> [ <mdi> [ <fm> ]]] )
```

Con esta instrucción dentro de la fuente independiente se especifica una onda sinusoidal modulada en frecuencia. Los parámetros `vo` y `va` tienen el mismo significado que en el seno, el primero es la componente de continua y el segundo la amplitud de la onda, siendo ambos obligatorios. El parámetro `fc` es la frecuencia de la portadora, `mdi` es el índice de modulación y `fm` es la frecuencia de modulación. La expresión que da la forma de onda en función de estos parámetros y el tiempo es la siguiente:

$$V(t) = vo + va \cdot \sin\left(2\pi \cdot fc \cdot t \cdot mdi \cdot \sin(2\pi \cdot fm \cdot t)\right)$$

```
PWL (<t1> <v1> ...  <tn> <vn>)
```

La especificación `PWL` sirve para describir una onda lineal a tramos. Simplemente consiste en una lista de pares tiempo-voltaje de manera que la forma de onda resultante es la unión mediante líneas rectas de dichos puntos.

Las opciones aquí comentadas, salvo por los paréntesis de los transitorios que son exclusivos de PSpice, coinciden con las especificaciones originales de SPICE. PSpice además añade algunas opciones y parámetros extras pero que son de uso menos frecuente y por eso no se han incluido aquí.

Fuentes controladas

Las fuentes controladas, a diferencia de las independientes, tienen una entrada de manera que la salida es función de esta entrada. Fuentes controladas hay de cuatro tipos:

Fuente de tensión controlada por tensión: En estas fuentes la tensión de salida es función de la tensión de entrada. Una fuente de este tipo suele modelar habitualmente un amplificador por lo que la función de transferencia suele ser simplemente una ganancia aunque existen muchas posibilidades. La forma de describirse en PSpice es como sigue:
`E<nombre> <n+> <n-> <nc+> <nc-> <ganancia>`
Esta es la forma básica, pero en vez de la ganancia hay muchas otras formas de especificar la relación entre la tensión de entrada de control (tensión entre los nodos `nc+` y `nc-`) y la salida (`n+` y `n-`). Se puede especificar una función polinómica con `POLY`, una fórmula aritmética con `VALUE`, una tabla con `TABLE`, una transformación de tipo Laplace con `LAPLACE`, una frecuencia con `FREC` y un filtro de Chebyshev con `CHEBYSHEV`. En estos casos los nodos de control están incluidos en la definición de la función.

Fuente de corriente controlada por corriente: La corriente de salida es función de la corriente que pasa por una fuente de tensión independiente. La forma típica es la siguiente:
`F<nombre> <n+> <n-> V<nombre> <ganancia>`
En el PSPICE original la única forma que había de especificar una corriente por una rama era referenciando una fuente de tensión en esa rama; si no la había entonces se creaba una con 0 de diferencia de potencial lo que equivale a una conexión directa. Es por esta razón que la corriente de referencia se toma de una fuente de tensión (`V<nombre>`). El factor `ganancia` da la ganancia del amplificador de corriente. Al igual que en la fuente de tensión se puede especificar la función de otras maneras aunque en las fuentes controladas por corriente esto queda limitado a la forma polinómica `POLY`.

Fuente de corriente controlada por tensión: Es una fuente cuya corriente de salida es función de una tensión de control en la entrada. La forma habituañ es como sigue:
`G<nombre> <n+> <n-> <nc+> <nc-> <transconductancia>`
Como la relación entre la corriente y tensión es la transconductacia se le llama así al factor interno que regula la fuente de corriente controlada por tensión. Al igual que la fuente de tensión controlada por tensión en este caso se admiten todas las posibilidades de definición de la función de transferencia vistas en aquella fuente.

Fuente de tensión controlada por corriente: Es una fuente cuya tensión depende de la corriente que atraviesa una fuente de tensión independiente. La forma común de utilización es esta:
`H<nombre> <n+> <n-> V<nombre> <transresistencia>`
Como se trata de una fuente controlada por corriente la única variedad permitida para la descripción de la función de transferencia es la polinómica (`POLY`).

1.3.5 Análisis eléctrico con PSpice

Los principales análisis eléctricos son tres: análisis de transitorios (función del tiempo), análisis de alterna (función de la frecuencia) y análisis de continua (función de algún parámetro estático). Combinados con estos análisis se pueden hacer otros como el de ruido, montecarlo, etc. También es importante calcular el punto de operación, ya que aunque se trata de un análisis muy simple que se pasa por alto, es también la base para el resto de análisis.

Todos estos análisis se pueden realizar a un tiempo sobre el mismo circuito, dependiendo del tipo de análisis elegido se tendrán en cuanta unas componentes u otras de las fuentes. Así para un análisis de continua sólo se tiene en cuenta la componente continua (`DC`) de las fuentes salvo en aquellas en que esta componente esté variando a causa de la instrucción de análisis, en el análisis de alterna se tienen en cuenta la componente continua (`DC`) y la alterna (`AC`), en el análisis de transitorios sólo se tiene en cuenta la componente de transitorios, es decir, definiciones de pulsos, senos, etc.

El punto de operación (`.OP`)

El cálculo del punto de operación de un circuito es el análisis básico de SPICE. Consiste en calcular todas las tensiones de todos los nodos y todas las corrientes de todas las ramas suponiendo que no hay fuentes que dependan del tiempo (ni transitorios ni corrientes alternas), y que el circuito se encuentra completamente estabilizado. Lo que se hace en realidad es eliminar todos aquellos componentes que tienen una dependencia con el tiempo, así los condensadores simplemente se eliminan del circuito, y las bobinas se sustituyen por conexiones directas.

Cuando se especifica un análisis de este tipo se genera una lista de nodos con sus tensiones así como una lista de ramas con sus corrientes. No tiene sentido hacer gráficos con estos datos por lo que aparecen en el fichero de salida estándar de SPICE, o sea, el fichero con extensión OUT. La forma para decirle a PSPICE que realice este análisis es con la instrucción:

```
.OP
```

Es un análisis muy básico y normalmente son pocas las ocasiones en que interesa hacerlo. De todas formas, para el resto de análisis, sobre todo para el de alterna, conocer el punto de operación del circuito es fundamental por lo que SPICE lo calcula, sin mostrar el resultado, aunque no se haya especificado.

Análisis de continua (`.DC`)

Un análisis de continua sirve para ver cómo varían unos parámetros del circuito en función de otros. En los inicios de SPICE la simulación de continua sólo servía para ver la evolución de la tensión o corriente en algún punto (la salida por ejemplo) en función de la tensión o corriente en otro (la entrada por ejemplo). Pronto se extendió este mismo análisis a otros parámetros como la temperatura, valores de los componentes como la resistencia, capacidad, longitud del transistor, etc.

El análisis de continua consiste entonces en calcular sucesivamente el punto de operación cambiando cada vez un parámetro, que puede ser casi cualquier parámetro de especificación del circuito. Con todos los puntos de operación se puede hacer una gráfica para mostrar cómo varían la tensión o corriente en función del parámetro que va variando. La forma de especificar un análisis de continua es la siguiente:

```
.DC [LIN] <variable> <inicio> <fin> <incremento> [anidamiento]
.DC {OCT | DEC} <variable> <inicio> <fin> <puntos> [anidamiento]
.DC <variable> LIST <valor> [,<valor>...]  [anidamiento]
```

La idea básica de esta instrucción es la de indicar el parámetro y cómo varía. En las dos primeras se dan dos formas de variación: la lineal que se supone por defecto si no se especifica nada, o la logarítmica que puede aumentar por décadas `DEC` u octavas `OCT`. La única diferencia entre las dos es el parámetro de incremento en una y puntos en la otra: SPICE calcula el punto de operación para cada valor que toma la variable, si la variación es lineal la variable se va incrementando según lo especificado en el parámetro `incremento`, mientras que en la logarítmica el parámetro que va en esta posición indica el número de puntos por década u octava para los cuales se va a calcular el punto de operación, ya que un incremento fijo en la escala logarítmica no tiene demasiado sentido. La tercera posibilidad consiste en especificar la variable y dar directamente los valores que va tomando en forma de lista (`LIST`).

Opcionalmente, al final de cada una de estas posibilidades se puede especificar otro análisis DC completo, es decir, otra variable que también tendrá su especificación de variación. Esto sirve para poder ver la variación de los parámetros del circuito en

función de dos parámetros. Lo que se obtiene es una colección de curvas cada una de ellas correspondientes a un valor de la variable anidada. Esto también se puede hacer con otras instrucciones como `.STEP`.

La variable que va cambiando puede ser casi cualquier cosa:

Fuentes: Es lo más habitual y consiste en poner el nombre de una fuente o corriente, de manera que cada vez se calcula el punto de operación con un valor diferente de tensión o corriente en la fuente.

Parámetro global: En PSpice se pueden definir variables o parámetros con la instrucción `.PARAM` de manera que estos parámetros se puede usar en la descripción de los circuitos como resistencias, etc. Para esto hay que poner `PARAM` seguido del nombre del parámetro en la definición del análisis de continua.

Parámetro de un modelo: Se pueden cambiar también los parámetros internos de los modelos. Para ello se pone el nombre del modelo seguido del parámetro interno encerrado entre paréntesis. No todos los parámetros son accesibles por este método.

Temperatura: Se puede variar la temperatura global del circuito sustituyendo la variable por `TEMP` que es la variable global en SPICE que lleva la temperatura del sistema.

Ejemplos:

```
.DC Ventrada 0 5 0.1
.DC Vce 0 10v 0.2v OCT Ib 0.1mA 10mA 2
.DC PARAM resistencia LIST 1k 2k 10k 12k
.DC Nmos1(VTO) 0 1V 0.1
```

Análisis de alterna (`.AC`)

El objetivo del análisis de alterna es obtener las tensiones y corrientes del circuito en función de la frecuencia. En este caso no se trabaja con tensiones o corrientes reales sino con sus valores eficaces o sus partes reales (amplitud) e imaginarias (fase).

La instrucción para que PSpice realice el análisis de alterna tiene la siguiente forma general:

```
.AC {[LIN] | DEC | OCT} <puntos> <frec_inicio> <frec_final>
```

Al igual que antes el primer parámetro indica si la frecuencia, que es la variable que cambia en este caso, crece de forma lineal o logarítmica. Si no se especifica nada se supone que es lineal. El siguiente parámetros son los puntos, si la simulación es logarítmica serán los puntos por década u octava, y si es lineal serán los puntos totales en todo el rango de frecuencias especificado. Por último se especifican la frecuencia de inicio y la de final, si la variación se ha elegido logarítmica la frecuencia inicial no puede ser cero. Habitualmente los análisis en función de la frecuencia se suelen realizar con una variación logarítmica siempre que la variación de frecuencias sea grande, si la variación es pequeña entonces es preferible el lineal.

Ejemplos:

```
.AC 100 1Hz 2Hz
.AC DEC  2 10 1MEG
.AC OCT 10Hz 22KHz
```

Es importante conocer de qué manera se realiza el análisis de alterna en SPICE, ya que este tipo de simulación puede dar resultados que pueden parecer erróneos si no se sabe lo que está haciendo realmente. Los pasos que se siguen en la simulación son los siguientes:

1. En primer lugar se calcula el punto de operación del circuito. Esto sirve para saber las componentes continuas de tensiones y corrientes en nodos y ramas. Esto es importante porque las componentes de alterna se superponen a estas componentes de continua.
2. Una vez se tienen las componentes continuas SPICE calcula todos los parámetros de alterna de todos los modelos para esas condiciones concretas de tensión y corriente. Estos parámetros describen el comportamiento del circuito de forma lineal alrededor de un infinitesimal alrededor del punto de operación.
3. Estos parámetros de alterna que dan el comportamiento alrededor del punto de operación permiten calcular las componentes alternas de tensión y corriente en todos los nodos y ramas del circuito.

Esta forma de calcular las componentes alternas resultantes del análisis introduce un error en los resultados. Esto es así porque los parámetros de alterna de los modelos se calculan para el punto de operación y sólo sirven para un infinitesimal alrededor de este punto. Esto quiere decir que si las amplitudes de las señales alternas son grandes el resultado será incorrecto debido a la aproximación que se ha hecho. Sólo en el caso de que las amplitudes de las señales alternas sean lo suficientemente pequeñas, como para suponer que en el rango de variación la respuesta varía de forma lineal, se podrá tener confianza en los resultados, si no es así los resultados pueden ser incorrectos. Por este motivo al análisis de alterna se le suele llamar también **análisis de pequeña señal**.

Un ejemplo del error que se puede cometer al utilizar incorrectamente el análisis de alterna se tiene en un amplificador normal. Supongamos que se tiene un circuito amplificador bastante sencillo alimentado con una batería de 12 voltios. El amplificador está diseñado para tener una ganancia de 100 de manera que la tensión de salida es 100 veces la de entrada. Se puede realizar la simulación asignándole a la entrada una componente alterna de 5 voltios por ejemplo. Se realiza el análisis de alterna se obtiene que la componente alterna de la salida es de 500 voltios!!! Naturalmente el amplificador como está alimentado con 12 voltios no podrá dar más tensión que esta a la salida por lo que es imposible obtener 500 voltios. Esto es así porque al realizar el análisis se han calculado los parámetros del circuito, en este caso la ganancia, en el punto de operación, esto quiere decir que la ganancia es efectivamente 100 en el punto de operación y en un infinitesimal alrededor, pero fuera de aquí se pierde la linealidad y los resultados son erróneos. Para realizar correctamente la simulación habría que haber elegido una tensión de 0.1 voltios como mucho que a la salida darían 10 voltios que probablemente caigan en la zona lineal del amplificador. Por último decir que se pueden hacer amplificadores que den efectivamente 500 voltios a la salida aunque la alimentación sea de 12 voltios, pero se estaba suponiendo un amplificador sencillo sin estas sofisticaciones.

Análisis de transitorios (`.TRAN`)

Este tipo de análisis pretende calcular las tensiones y corrientes en todo el circuito y mostrar cómo varían en función del paso del tiempo. En este caso se ignoran las componentes alternas y continuas y se consideran solamente las componentes de transitorio.

Esta es la forma es que se especifica el análisis de transitorios:

```
.TRAN [/OP] <paso_impresión> <tiempo_final>
+[retraso_impresión [paso_máximo]] [UIC]
```

El parámetro opcional `/OP` incluye en el fichero de salida los resultados del cálculo de operación que en el caso de los transitorios es algo diferentes al punto de operación en continua tal y como se explica más adelante. El parámetro `paso_impresión` sirve para especificar cada cuanto se quiere mostrar los resultados de forma gráfica o numérica, un valor pequeño de este parámetro dará gráficas más redondeadas pero generará un fichero de datos más grande. En ningún caso se gana en precisión numérica por el hecho de disminuir este parámetro que sólo tiene influencia en la visualización de los datos. El `tiempo_final` indica hasta cuando debe simularse. El siguiente parámetro es opcional y sirve para ver a partir de qué momento se desea generar los datos. El siguiente parámetro, también opcional, permite modificar el intervalo de cálculo, y este sí que puede tener cierta influencia sobre la precisión de los resultados de salida, a menor intervalo mayor precisión; también permite en ocasiones solucionar algunos problemas de convergencia en encontrar la solución numérica. El parámetro `UIC` (*Use Initial Conditions*) es opcional y cuando se especifica sirve para indicarle al simulador que utilice las condiciones iniciales de tensión y corriente especificadas con el parámetro `IC` en condensadores y bobinas o mediante la instrucción `.IC` de PSpice.

El punto de partida de una simulación de transitorios es siempre $t = 0$. En la simulación de transitorios es importante conocer las condiciones iniciales ($t = 0$) en que se encuentra el circuito. Si no se ha especificado la opción `UIC` en `.TRAN`, entonces lo que hace SPICE es calcular el punto de operación del circuito pero con una diferencia respecto del `.OP`: no se tienen en cuenta las componentes `DC` de las fuentes, sino que en su lugar se calcula las componentes transitorias de estas fuentes tomando $t = 0$. Una vez calculado el punto de operación de esta manera se tienen las condiciones iniciales de tensión y corriente en todos los nodos y ramas y puede dar comienzo la simulación de transitorios.

Otros análisis e instrucciones

Estos análisis básicos se pueden combinar con otros tipos de análisis que dan información adicional sobre los circuitos que se están simulando. Una de las características que más se utiliza es la de repetir la misma simulación pero cambiando algún parámetro del circuito. Para empezar, si el parámetro que se quiere cambiar no existe se crea con la siguiente instrucción:

```
.PARAM <parámetro> = <valor>
.PARAM <parámetro> = {<expresión>}
```

Las dos formas en realidad son lo mismo, en la primera se le asigna un valor al parámetro y en la segunda una expresión. Es importante destacar que las expresiones en PSpice van siempre encerradas entre llaves. (En este caso las llaves que aparecen en la definición son literales que hay que poner.) Cuando se hace uso de estos parámetros se entiende que son por sí mismos expresiones por lo que deben siempre estar encerrados entre llaves.

Estos parámetros se pueden utilizar en casi cualquier parte de la descripción del diseño y sustituyen a valores numéricos, como por ejemplo resistencias, componentes de tensión en fuentes, parámetros en algunos dispositivos, etc. En una misma instrucción

`.PARAM` se pueden especificar varias variables unas seguidas de otras.

Para repetir un mismo análisis (sea de transitorios, alterna o continua) para valores diferentes de algún parámetro, se utiliza la siguiente instrucción cuya sintaxis es prácticamente idéntica a la de la instrucción de análisis de contínua (`.DC`):

```
.STEP [LIN] <variable> <inicio> <fin> <incremento>
.STEP {OCT | DEC} <variable> <inicio> <fin> <puntos>
.STEP <variable> LIST <valor> [,<valor>...]
```

La única diferencia con la instrucción del análisis de continua es que esta última permitía una variable adicional anidada. Al igual que en el análisis de continua la variable que va cambiando puede ser el valor de una fuente, un parámetro global, un parámetro de un modelo o la temperatura `TEMP`.

Ejemplos de parámetros y `.STEP`:

```
.PARAM res=10K
.PARAM otrares={2*res} cap=2u
R1 1 0 {res}
C1 2 1 {cap}
.STEP PARAM res LIST 1K 2K 7K 10K 20K
.STEP DEC I2 1uA 1A 4
```

Aunque en el ejemplo anterior se muestran varias instrucciones `.STEP` sólo se permite una por descripción, de la misma manera que sólo se permite una instrucción de análisis de continua, una de alterna y una de transitorios.

De uso menos frecuente son los siguientes análisis disponibles en el simulador PSpice:

.TF Calcula la función de transferencia entre una variable de salida y una fuente de entrada. Típicamente se utiliza para calcular la ganancia en un circuito. El resultado es un valor constante calculado para el punto de operación del circuito. La salida se da en forma de texto en el fichero `OUT`.

.SENS Calcula el efecto de todos los componentes del circuito sobre una variable de salida, o varias, que además son los únicos parámetros que acepta esta instrucción. El resultado se da en un fichero de texto `OUT`.

.NOISE Realiza un análisis de ruido y debe especificarse junto con una simulación de alterna (`.AC`). Los resultados del análisis se muestran en función de la frecuencia.

.FOUR Realiza una descomposición en armónicos de Fourier de los resultados obtenidos de un análisis de transitorios (`.TRAN`). Esto significa que este análisis se debe realizar junto con el de transitorios. Se especifica la frecuencia fundamental y el número de armónicos que se desean, devolviendo el resultado en el fichero `OUT`.

.TEMP Fija la temperatura de funcionamiento del circuito. Si se especifican varios valores entonces se repiten los análisis especificados para cada valor de la temperatura.

.MC Realiza una simulación de Monte Carlo que consiste en repetir el mismo análisis varias veces cambiando los valores de ciertos parámetros de unos valores de tolerancia. Se especifica en la propia instrucción el tipo de análisis que se quiere realizar, los parámetros a cambiar, etc. La salida puede ir dirigida al fichero de salida de texto `OUT` o a uno de datos para su representación gráfica dependiendo de lo que se haya especificado.

.WCASE Realiza una simulación parecida pero más sistemática lo que permite descubir el peor caso, es decir, aquella combinación de los parámetros tal que la

desviación de los que se está mirando (por ejemplo la tensión de salida) es máxima. La forma de operar es como sigue: la primera pasada se hace con los valores nominales, las siguientes pasadas se realizan cambiando cada vez un parámetro distinto a sus valores de tolerancia extremos, con esto se calcula la sensibilidad de cada parámetro sobre la salida. Conocida la sensibilidad de cada parámetro sobre la salida se realiza una última pasada cambiando todos los parámetros para que la salida tenga la mayor desviación.

1.3.6 Ejemplo: Análisis del transistor en conmutación

La mayoría de los análisis vistos hasta ahora se pueden poner en un ejemplo típico que es el del análisis de una puerta inversora CMOS. La simulación de continua da la función de transferencia de la puerta, lo que permite medir los niveles de ruido de la familia lógica, tensión de conmutación, etc. La simulación de alterna tiene menos sentido ya que se trata de un dispositivo digital, pero se puede experimentar con el inversor en la zona de transición de alto a bajo, ya que justo en esa zona se comporta como un amplificador de ganancia bastante grande, esto exige que se polarice correctamente la componente de continua del punto de operación para que trabaje en esta zona. Por último se tiene la simulación de transitorios donde se pueden medir los retrasos de la puerta sin más que colocar un tren de pulsos a la entrada. También se puede hacer una simulación de transitorios donde se introduzca una onda sinusoidal para ver la posible distorsión que se produce respecto de la onda original. Todos estos análisis, junto con la repetición de cada uno cambiando algún parámetro de diseño, se muestran en el siguiente ejemplo. Alguna líneas están comentadas debido a que sólo se puede cambiar un parámetro cada vez con `.STEP` o porque sólo se puede especificar una sola componente de transitorios en la fuente.

Ejemplo:

```
* Ejemplo de análisis de un inversor
.PARAM ancho=1u ampli=0.1v
.PARAM carga=1pF

Vsrc vdd 0 DC=5v   ; Fuente de alimentación del circuito.

Vin in 0           ; Fuente que da los estímulos:
+ DC=2.45v                          ; polarización para análisis de alterna
+ AC=0.1v                           ; componente alterna
+ PULSE 0v 5v 0s 1ns 1ns 10ns 20ns  ; tren de pulsos
* + SIN 2.45v {ampli}               ; sin más parámetros especifica una onda

Mp out in vdd vdd modp L=1U W={3*ancho}    ; transistor PMOS
Mn out in 0 0     modn L=1U W={ancho}      ; transistor NMOS
Cload out 0 {carga}                        ; Condensador de carga a la salida
.MODEL modp PMOS                           ; Modelos para ambos transistores
.MODEL modn NMOS

.DC Vin 0v 5v 0.1v     ; Da la función de transferencia entre salida/entrada.
.AC DEC 10Hz 1THz 2    ; Pretende mostrar las frec. de corte, pero con los modelos
                       ; que se han dado no hay f. de corte.
.TRAN 0.1ns 40ns       ; Muestra un par de pulsos.

.STEP PARAM ancho 1u 100u 5u         ; Repite cada análisis con anchos diferentes.
* .STEP PARAM carga 1pF 100pf 10pF   ; Repite con cargas diferentes.
* .STEP PARAM ampli 0.05v 0.4v 0.05v ; Repite con amplitudes diferentes

.PROBE   ; Genera fichero con extensión .DAT para la representación gráfica de datos.
.END
```

1.3.7 Convergencia y errores

Dado que muchas veces se utilizan soluciones numéricas para llegar al resultado, es posible que el algoritmo no converja para determinada combinación de los datos de entrada. En estos casos el resultado que se obtiene no es exacto y no debe ser tenido en cuenta. Normalmente SPICE no avisa cuando se ha producido un error de este tipo, y debe ser el diseñador que, a la vista de los resultados, determine si son fiables o no.

Normalmente los errores en la convergencia de los algoritmos de cálculo se pueden detectar fácilmente mirando las gráficas de resultados. Si de repente una respuesta lineal o curva monótona tiene un pico en un lugar en que no debería, entonces probablemente se trata de un error de cálculo de SPICE. En estos casos se debe repetir el análisis cambiando ligeramente alguno de los parámetros del circuito y así descubrir si se trata de un error de cálculo o realmente ese pico debe estar ahí.

Hay otra serie de parámetros que también pueden modificarse para aumentar la precisión de los cálculos. Muchas veces cambiando estos valores, incluso disminuyendo la precisión, pueden hacer que desaparezcan los errores de convergencia. Estos parámetros le sirven a SPICE para realizar sus cálculos y son parámetros internos del simulador. Hay más de 40 parámetros que controlan el funcionamiento de SPICE y que se pueden cambiar dependiendo de la aplicación de que se trate.

Los parámetros de SPICE se especifican mediante la instrucción `.OPTIONS` que tiene exactamente la misma sintaxis que `.PARAM` con la diferencia de que sólo se pueden especificar los parámetros predefinidos de SPICE. Se han seleccionado a continuación aquellos parámetros que tienen que ver con la precisión de SPICE a la hora de realizar los cálculos:

ABSTOL Especifica la máxima precisión para corrientes. Su valor por defecto es `1pA`.

CHGTOL Es la máxima precisión para cargas. Por defecto toma el valor `0.01pC`.

VNTOL Es la máxima precisión para tensiones y toma el valor de `1uV`.

RELTOL Es la tensión relativa para tensiones y corrientes. Viene dada en tanto por uno y su valor por defecto es `0.001` (0.1%).

1.4 Simulación digital

1.4.1 El ciclo de diseño de circuitos digitales

La simulación es una parte fundamental dentro del ciclo de diseño de circuitos tal y como se muestra en la figura 1.4. Todo el flujo de diseño se divide en tres partes: descripción, simulación-prueba, realización. La parte de simulación y prueba consiste en, dada la descripción del circuito, comprobar mediante un modelo del circuito, que funciona correctamente.

Dentro de la simulación hay diferentes niveles dependiendo de lo que se esté simulando; un primer paso consiste en simular el circuito lógico (si es que se está simulando un sistema eléctrico) para ver si la respuesta es la correcta. Un siguiente paso consiste en añadir los retrasos propios de las puertas, lo cual depende de la tecnología utilizada; en este caso se quiere ver que el circuito funciona y además cumple con los requisitos de tiempo. Un refinamiento, que se da sobre todo en los circuitos integrados, es realizar otra simulación después de que se ha realizado el trazado de pistas, ya que este cableado

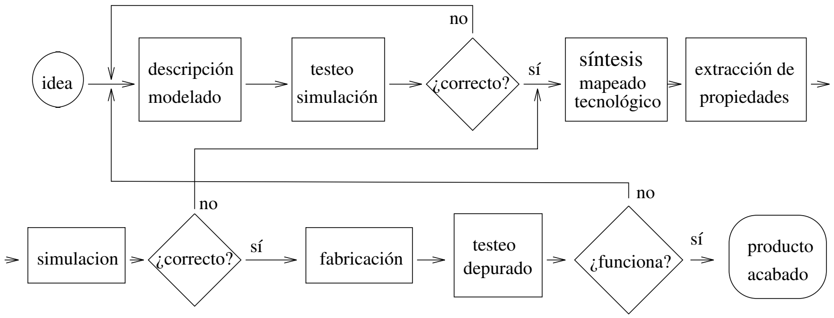

Figura 1.4: Flujo de diseño de circuitos

introduce retrasos adicionales que igual no se han tenido en cuenta. Es lo que se llama la simulación *postlayout* o post-trazado. Por último, existen otro tipo de simulaciones que no se limitan a ver cómo funciona el circuito sino que extraen estadísticas que pueden resultar importantes en algunos contextos, como por ejemplo la simulación de faltas.

Capítulo 2

Circuitos integrados. VLSI

2.1 Introducción

2.2 Procesos de fabricación

2.3 Librería de celdas

2.4 Síntesis automática de circuitos integrados

Capítulo 3

Diseño de circuitos impresos (PCBs)

3.1 Diversas tecnologías de circuitos impresos. SMT y MCM

3.2 Encapsulados

3.3 Reglas de diseño

3.4 Emplazado óptimo. Estrategias

3.5 Trazado de pistas. Algoritmos de trazado de pistas

3.6 Estrategias de rutado de pistas

3.7 Fabricación y costes

3.8 Interferencias electromagnéticas

Apéndice A

Comentarios sobre la bibliografía

A continuación se exponen unos comentarios sobre la bibliografía de la asignatura. Se resaltan los libros clave pero se dan también otros donde ampliar información.

[1] Este libro contiene todos los contenidos de la parte de VHDL de la asignatura además de seguir la misma estructura explicativa. También incluye algunos contenidos de simulación digital que se ven en la asignatura después del VHDL.

[2] Este libro también está en castellano. Libro muy completo que trata el VHDL y otros temas de diseño lógico. No sigue la estructura del curso pero es una muy buena lectura adicional.

[3] Son apuntes sobre VHDL. Tienen interés por encontrarse en castellano pero las anteriores referencias son sin duda mejores.

[4] Libro dedicado al uso del VHDL en la síntesis automática de circuitos, sin dejar de lado las características propias del lenguaje.

[5] De los libros en inglés es uno de los más claros sobre VHDL.

[6] Uno de los mejores libros sobre VHDL que se han escrito. Es realmente extenso cubriendo prácticamente todos los aspectos del lenguaje.

[7] Fue uno de los primeros en publicarse y no es tan bueno como los presentados anteriormente. Lo más interesante de este libro es que está disponible en la biblioteca.

[8] Son unos apuntes básicos sobre VHDL que se pueden conseguir en la red. Están bien como introducción pero nada más.

[9] Este libro presenta aspectos básicos sobre la simulación en general, se corresponde con el tema de introducción a la simulación.

[10] Libro muy bueno sobre el simulador SPICE en general.

[11] Este está particularizado al PSpice.

[12], [13] Otros libros sobre Spice.

[14] Interesante libro que presenta una introducción a la fabricación y diseño de circuitos integrados. Hay varias copias en la biblioteca.

[15], [16] Dos libros del mismo autor sobre la fabricación de circuitos integrados y dispositivos MOS. Es bastante avanzado.

[17] Otro libro de diseño de circuitos integrados.

[18] Libro introductorio sobre el diseño de circuitos impresos (PCBs).

[19] En este se trata en profundidad el diseño para alta velocidad.

[20], [21] Estos dos se centran en los dispositivos de montaje superficial.

[22] Libro interesante sobre circuitos impresos pero que aun no está en la biblioteca.

Bibliografía

[1] Fernando Pardo y José A. Boluda. *VHDL, lenguaje para síntesis y modelado de circuitos*. RA-MA, 1999. BIBLIOTECA: CI 681.3 PAR (2 copias), CI-Informática (1 copia). http://tapec.uv.es/VHDL/libro.html.

[2] Lluís Terés, Yago Torroja, Serafín Olcoz, Eugenio Villar, *et al.* *VHDL, lenguaje estándar de diseño electrónico*. McGraw-Hill, 1997. BIBLIOTECA: CI 681.3.06 VIL (2 copias), CI-Informática (1 copia).

[3] Rafael Gadea Gironés. *Lenguajes de descripción hardware, VHDL y Verilog*. Universidad Politécnica de Valencia, servicio de publicaciones, 1995. BIBLIOTECA: CI 681.3.06 LEN (4 copias).

[4] Kevin Skahill. *VHDL for programmable logic*. Addison-Wesley, 1996. BIBLIOTECA: CI 681.3 SKA (1 copia), CI-Informática (1 copia).

[5] Douglas L. Perry. *VHDL*. McGraw-Hill, segunda edición, 1993. BIBLIOTECA: CI-Informática (1 copia).

[6] Peter J. Ashenden. *The designer's guide to VHDL*. Morgan Kaufmann, 1996. BIBLIOTECA: CI-Informática (1 copia).

[7] Roger Lipsett, Carl Schaefer, y Cary Ussery. *VHDL: Hardware Description and Design*. Kluwer Academic Publishers, 1991. BIBLIOTECA: CI 681.3 LIP (2 copias).

[8] Peter J. Ashenden. *The VHDL Cookbook*, 1990. http://informatica.uv.es/guia/asignatu/TI/vhdl-cookbook.ps.tar.gz.

[9] Averill M. Law y W. David Kelton. *Simulation Modelling & Analysis*. McGraw-Hill, segunda edición, 1991. BIBLIOTECA: CI 519.8 LAW (2 copias), CI-Estadística (1 copia).

[10] Paolo Antognetti y Giuseppe Massobrio. *Semiconductor Device Modelling with Spice*. McGraw-Hill, segunda edición, 1988 y 1993 (2nd ed.). BIBLIOTECA: CI 681.3 MAS (1 copia), CI-Informática (2 copias).

[11] Paul W. Tuinenga. *A guide to circuit simulation & analysis using Pspice*. Prentice Hall, segunda edición, 1992. BIBLIOTECA: CI 621.3 TUI (1 copia).

[12] Lawrence G. Meares y Charles E. Hymowitz. *Simulating with Spice*. Intusoft, 1988. BIBLIOTECA: CI 681.3.06 MEA (1 copia).

[13] Karl Heinz Müller. *A Spice Cookbook*. Intusoft, 1991. BIBLIOTECA: CI-Informática (1 copia).

[14] Eugene D. Fabricius. *Introduction to VLSI design*. McGraw-Hill, 1990. BIBLIOTECA: CI 621.3 FAB (2 copias), CI-Informática (1 copia).

[15] S.M. Sze. *Semiconductor Devices, Physics and Technology*. John Wiley & Sons, 1985. BIBLIOTECA: CI 538.9 SZE (2 copias), CI-Fís.Aplicada (1 copia).

[16] S.M. Sze. *Physics of Semiconductor devices*. John Wiley & Sons, 1981. BIBLIOTECA: CI 538.9 SZE (3 copias), CI-IFIC (2 copias).

[17] John P. Uyemura. *Fundamentals of MOS Digital Integrated Circuits*. Addison Wesley, 1988.

[18] Michael Flatt. *Printed Circuit Board Basics: An introduction to the PCB industry*. Miller Freeman Publications, tercera edición, 1992. BIBLIOTECA: CI 537.8 FLA (1 copia).

[19] Howard W. Johnson y Martin Graham. *High-Speed Digital Design: A handbook of black magic*. Prentice Hall, 1993. BIBLIOTECA: CI 621.3 JOH (1 copia), CI-Informática (1 copia), CI-Electrónica (1 copia).

[20] Vern Solberg. *Design Guidelines for Surface Mount Technology*. TAB Books Inc., 1990. BIBLIOTECA: CI-Informática (1 copia).

[21] Frank Classon. *Surface Mount Technology for Concurrent Engineering and Manufacturing*. McGraw-Hill, 1993. BIBLIOTECA: CI-Informática (1 copia).

[22] Clyde F. Coombs Jr. *Printed Circuits Handbook*. McGraw-Hill, tercera edición, 1988. BIBLIOTECA: CI-Informática (1 copia).