UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

REDUCCIÓN DE ELEMENTOS PARASITARIOS EN LISTA DE CONEXIONES POST-DISEÑO PARA APLICACIONES INDUSTRIALES COMPATIBLES CON ESP

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL ELÉCTRICO

ADOLFO JAVIER OBLIGADO DÍAZ

PROFESOR GUÍA:
FRANCISCO JAVIER RIVERA SERRANO

MIEMBROS DE LA COMISIÓN:
RICARDO ALBERTO FINGER CAMUS
SIMÓN ADRIÁN ZÚÑIGA ACUÑA

SANTIAGO DE CHILE
2022

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO
POR: ADOLFO JAVIER OBLIGADO DÍAZ
FECHA: 2022
PROF. GUÍA: FRANCISCO JAVIER RIVERA SERRANO

REDUCCIÓN DE ELEMENTOS PARASITARIOS EN LISTA DE CONEXIONES POST-DISEÑO PARA APLICACIONES INDUSTRIALES COMPATIBLES CON ESP

Una vez se termina el proceso de diseñar un circuito integrado, llega la tarea de "imprimirlo", para lo que se obtiene una lista de conexión post-diseño. Esta *netlist* contiene cada uno de los elementos necesarios para el funcionamiento del circuito, pero también con los llamados **elementos parasitarios**, la aparición de estos elementos responde a las distintas interacciones y características eléctricas de los componentes del circuito.

Para ayudar en todo el proceso relacionada con el desarrollo de circuitos integrados, empresas se han dedicado al desarrollo de *softwares* especializados, entre ellas se encuentra Synopsys. Dentro de las múltiples herramientas que desarrollan se encuentra *Embedded Symbolic Processor* o ESP por sus siglas en inglés, que corresponde a una herramienta de verificación formal. Las listas de conexión post-diseño, al incorporar los mencionados elementos parasitarios, traen complicaciones durante el proceso de verificación.

Este trabajo tiene como objetivo incorporar un método de reducción parasitaria que permita mejorar la calidad de la verificación realizada por la herramienta ESP. Para lo anterior todo inicia con una investigación bibliográfica, en búsqueda de los distintos métodos de reducción de orden existentes, a esto se le suma el estudio relacionado sobre la propia herramienta y las distintas alternativas de representación de circuitos.

Luego se presenta una colaboración con otro equipo que cuenta con su propia herramienta de reducción, esto con la idea de comprobar la mejora en el desempeño de ESP reduciendo o eliminando elementos parasitarios. Para las pruebas relacionadas se utilizaron dos librerías distintas, cada una con más de 150 elementos para verificar.

Avalados por los resultados entregados por la herramienta de StarRC, se da inicio al desarrollo de una solución propia de ESP, en búsqueda de eliminar, no reducir, los elementos parasitarios que se encuentran conectados entre las fuentes de voltajes del circuito y los transistores.

Los resultados la colaboración entre ESP y StarRC sorprendieron tanto que, para los clientes que cuentes con ambas licencias, la herramienta de reducción de StarRC se incorporó al flujo de ESP. Esta incorporación trae consigo una mejora de un %58 en los tiempos de verificación cuando solamente se eliminan los elementos parasitarios ligados a las fuentes de voltaje y un %99 de mejora si se realiza una reducción completa. El desarrollo e implementación de una solución propia de ESP, pensado en no depender de otra herramienta ni en las licencias que puede tener el cliente, sigue en pie.

A todos quienes creyeron en mi incluso cuando yo no lo hice,
muchas gracias.

Adolfo

Agradecimientos

Llegado este momento, que representa el fin de mi periodo de formación en la universidad, no queda nada más que agradecer a quienes me ayudaron en este camino, pido perdón si dejo a alguien fuera de esto.

Primero quiero agradecer a mi familia, sin su apoyo difícilmente podría estar aquí. Gracias papá por tus llamada incondicionales cada semana, o bien cuando te acordabas o querías sacar la vuelta, por mostrarme que se puede llegar lejos a pesar de tener que luchar día a día con tu propia mente. Gracias mamá por levantarte antes de que saliera el sol para asegurar de que nada nos faltara y perdón por hacer abuso de tu incapacidad de decirme que no a las cosas. Gracias Tata, por ser el ejemplo del trabajo, por tu enorme preocupación por si comía o no en la universidad y por esos churrascos chacareros que subsidiaste, me gustaría que aún estuvieras aquí, perdón por la demora. Gracias Tía Anita por soportarme, aconsejarme y proporcionarme un hogar cálido y lleno de amor durante todos estos años. Y no puede faltar la María José, que durante años pensé que era mi hermana, pero realmente es mi tía, gracias por ser la luz en la oscuridad, me ayudaste a salir del pozo más profundo y me devolviste las ganas de vivir, sin ti probablemente no estaría escribiendo esto.

En segundo lugar quiero dar las gracias a todos mis amigos de la vida y de la universidad. Gracias Matías Villegas por todos estos años de amistad, por los consejos, las ayudas en las tareas de programación y las salidas a comer, y perdón por todas las cosas que te convencí que compraras. Formamos un grupo muy lindo con Alfredo Gallegos, Vicente Caragol y Luis Osorio, sin ustedes mi paso por la universidad y por eléctrica hubiese sido insoportable, gracias por las ayudas en mis momentos de crisis y espero que nuestra amistad no termine con el fin de este ciclo. Gracias Gigi, por soportar mis largos periodos de ausencia como amigo, te prometo que lo voy a compensar. Gracias a mis virus de computador que me soporta diariamente, y me han acompañado durante todo este proceso Nato (me falta espacio para una dedicarotia solo para ti), Choco, Kirshan, Trompa, Indi, Panxo y puede que igual el Snako. Gracias Iquique por forjar a mis compas de la vida Franco, Pablo, Roberto y Jp. Finalmente a la Joce, gracias por hacerme parte de tu familia.

En tercer lugar quiero agradecer a los miembros de ESP Simón Zúñiga, Paola Yang (ex) y Tomás Wolf (ex), quienes me recibieron con los brazos abiertos para trabajar con ellos y me hicieron sentir parte importante del equipo desde el primer minuto.

Para finalizar quiero agradecer al profesor Francisco Rivera, quien gracias a sus distintos ramos me demostró, sin quererlo, que el camino que estaba siguiendo en mi formación no era lo que realmente quería.

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Capítulo 1

Introducción

1.1. Motivación y formulación del problema

La aparición de elementos parásitos en los circuitos eléctricos es inevitable, tanto así que incluso los propios elementos como lo son resistencias, capacitores e inductores presentan componente parásitas [1]. El simple hecho de que dos componentes conductores se encuentren cerca entre ellos genera una capacitancia parasitaria. Sumado a lo anterior, la cantidad de transistores cada vez es mayor en los circuitos integrados, donde se trata de llevar al límite las capacidades físicas y químicas del silicio, alcanzando transistores de hasta 7[nm] y tamaños menores están siendo estudiados.

En cada chip se pueden encontrar billones de transistores y, año tras año, la cantidad de estos elementos en la misma unidad de área aumenta. Entidades como *International Roadmap for Devices and Systems* [2] y *Semiconductor Industry Association* [3], constantemente se encuentran estudiando el estado actual del desarrollo de los semiconductores, incluso dan los lineamientos que este debe seguir durante los años venideros. El aumento en el número de transistores implica el aumento de lineas que los conecten entre sí. Dado que estas lineas no tienen comportamientos ideales, presentan tanto resistencias como inductancias [1] de naturaleza parasitaria.

Los elementos parasitarios no son un problema durante el proceso de diseño del circuito. estos se convierten en una problemática una vez el circuito en cuestión quiere ser sintetizado. Antes de la impresión del circuito se obtiene una lista de conexión particular, donde se describen todos los elementos que forman parte de este. Esta forma de representar los circuitos es bastante similar a SPICE, pero tiene sus particularidades que la diferencian de este. Si bien el nombre de este formato puede variar dependiendo de cada fabricante, de ahora en adelante, serán mencionados como archivos *.spf*

Una vez que se realiza el diseño de un circuito es necesario verificar su correcto funcionamiento, esto mediante el uso de distintos softwares disponibles en el mercado. En particular, se encuentra uno llamado *Embedded Symbolic Processor*, desde ahora en adelante ESP por sus siglas en inglés, la cual como sugiere su nombre, realiza el proceso de verificación de manera simbólica ([4], [5]). Esta herramienta es propiedad de la empresa Synopsys.

Dentro de este mismo documento se explicará para qué se utiliza y cómo funciona ESP

pero, por el momento, es interesante mencionar que la gran cantidad de elementos parasitarios presentes en una lista de conexión, conllevan a que dicha herramienta no funcione de manera óptima, ya que los niveles de voltaje aplicados en las entradas del circuito pueden no propagarse de manera correcta a lo largo de este. En el peor de los casos la herramienta es simplemente incapaz de realizar el proceso de verificación.

Como es de esperar, esta no es la primera vez que la cantidad de elementos y, por lo tanto, de conexiones es una preocupación, por lo mismo existen los llamados **Métodos de Reducción de Orden** ([6], [7], [8]), que son utilizados en toda clase de sistemas, no solamente en modelos eléctricos. Hay una wiki relacionado solamente con los modelos de reducción de orden [9], incluso existe la *European Network for Model Reduction* [10].

Con todo lo anterior en mente, se pueden plantear una serie de objetivos que se pretenden alcanzar con la realización de este trabajo de título.

1.2. Objetivos generales

Con el objetivo de asegurar un buen funcionamiento de ESP con listas de conexión post-diseño, se busca desarrollar un método de reducción de elementos parasitarios que se pueda acoplar al código base de la herramienta, mejorando la calidad de la verificación.

En pocas palabras, una vez que se implemente el método de reducción, se espera que todas las listas de conexión post-diseño puedan ser procesadas y que exista una mejora en los tiempos de procesamiento en aquellas que podían ser procesadas previamente.

1.3. Objetivos específicos

Una serie de objetivos de carácter específico se puede obtener durante realización de esta memoria, los cuales se encuentran definidos a continuación:

- Estudiar los distintos métodos teóricos de reducción parasitaria propuestos en la actualidad.
- Identificar que elementos parasitarios presente en las *netlist* puede simplemente ser eliminado
- Elaborar un código propio que implemente la reducción parasitaria
- Probar el rendimiento de herramientas ya elaboradas de reducción parasitaria
- Lograr que herramientas ya propuestas puedan ser utilizadas directamente por ESP

Capítulo 2

Marco Teórico y Estado del Arte

2.1. Ley de Moore

En el año 1965 el ingeniero Gordon Moore, quien fuera co-fundador de Intel, publicó en "Electronics Magazine" [11] un artículo en donde realiza una observación en cómo evolucionaría la cantidad de transistores en un microprocesador con el pasar de los años. Con información empírica, indica que dicha cantidad se duplicaría cada dos años, presentando la proyección que se ve en la figura 2.1:

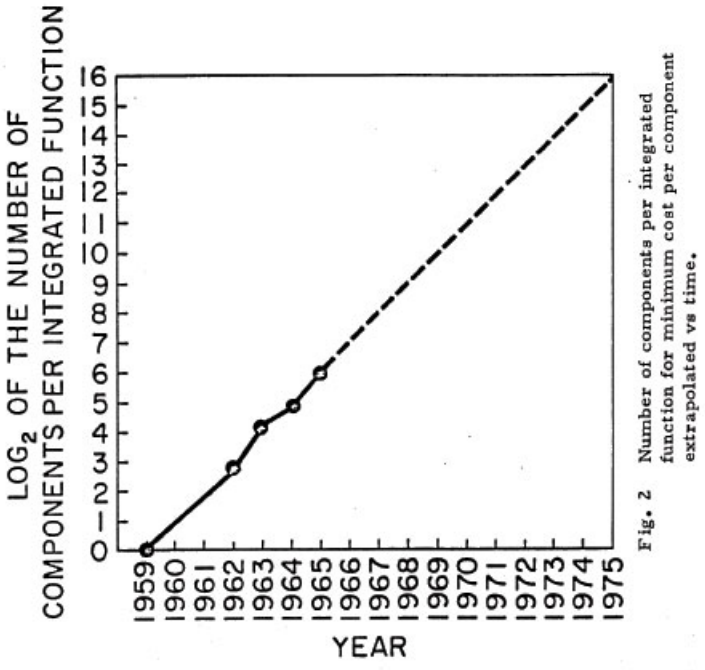


Figura 2.1: Gráfico original presentado por Moore en "Electronics Magazine".

La verdad es que nadie sabe si ese comportamiento realmente se iba a dar si es que Moore no daba esas declaraciones, pero no solamente se cumplieron sus palabras, sino que se quedaron cortas, ya que la cantidad de transistores se duplicó cada 18 meses.

En la figura 2.2 se aprecia un gráfico que muestra la cantidad de transistores en los circuitos integrales con el avance de los años, donde se aprecia la duplicación de esta cantidad.

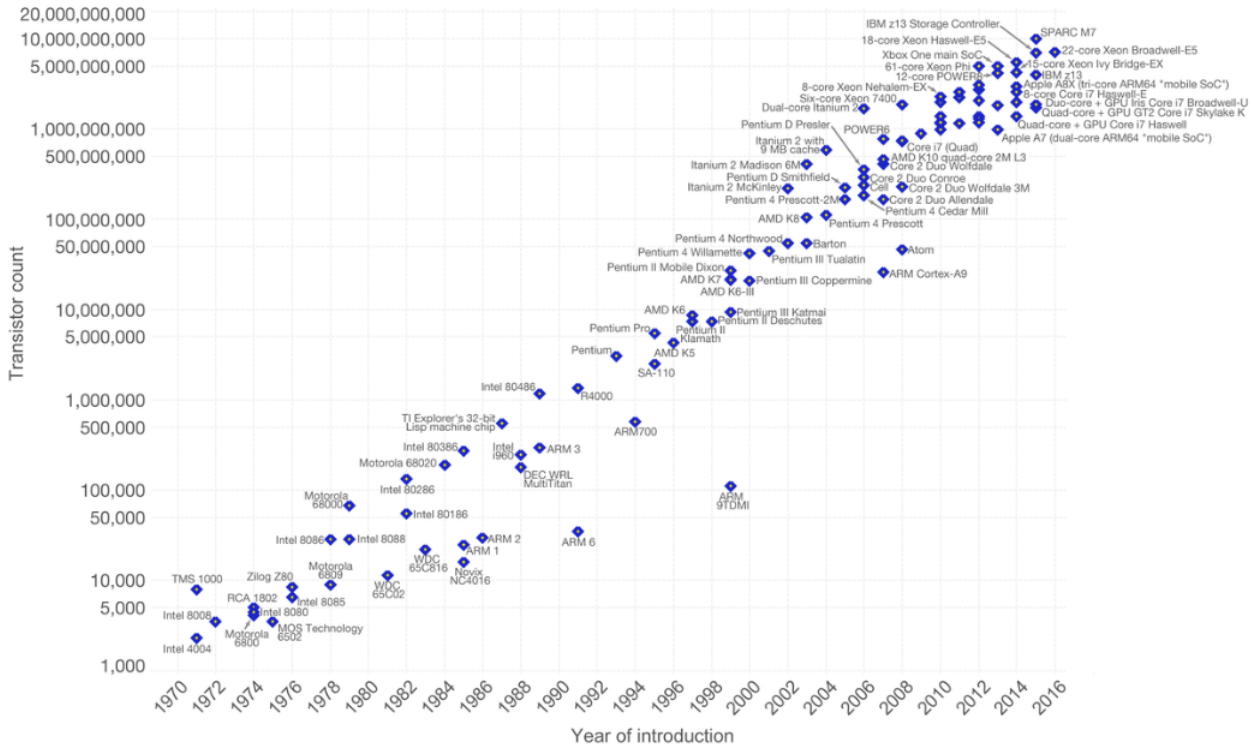


Figura 2.2: Gráfico que muestra la evolución en la cantidad de transistores con el pasar de los años.

La ley de Moore resulta importante hasta el día de hoy, a pesar de que ya han pasado más de 50 años desde la publicación del artículo, donde se han forjado términos como **More Moore** [12], *Beyond Moore* [13] y **More Than Moore** [14] para nombrar estrategias a seguir en búsqueda del aumento de elementos por microprocesador. Incluso *Synopsys* acuña su propia estrategia llamada *Sysmoore* [15].

La relevancia y la razón de mencionar la Ley de Moore, como un aspecto relevante para comprender el tema principal del trabajo, es que a medida que aumentan los elementos hay una mayor cantidad de conexiones y por ende una mayor interacción entre elementos conductores. Lo anterior conlleva a una mayor aparición de elementos parásitos.

2.2. Elementos Parasitarios

Durante la formación profesional, es muy común escuchar el termino "ideal" como una característica de los elementos que se están estudiando, dado que tomar en cuenta todas las variables que se pueden presentar en el mundo real, complicaría de manera excesiva la explicación de su comportamiento.

Un elemento parásito se puede definir como un elemento circuital que posee un comportamiento eléctrico conocido, pero cuya incorporación al circuito no estaba considerada durante el diseño. La aparición de elementos parasitarios dentro de los circuitos corresponde justamente a las propiedades no "ideales" de los componentes.

Una resistencia "ideal" cuenta con una impedancia que no depende de la frecuencia de operación [1]:

$$Z_{resistenciaideal} = R \tag{2.1}$$

Una resistencia real incluye una inductancia parásita, producto de las características geométricas del elemento conductor utilizado para su construcción, y una capacitancia parásita entre los conectores de la resistencia, producto del tiempo relacionado con la propagación de la corriente, esto se muestra en la figura 2.3.

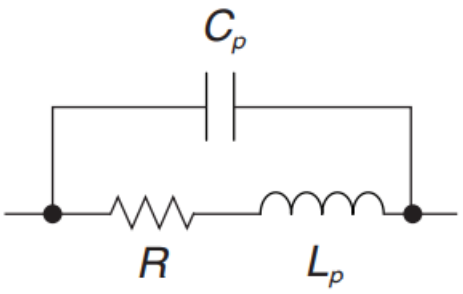


Figura 2.3: Resistencia R con su inductancia parásita en serie L_p y su capacitancia parásita en serie C_p

Con lo anterior, se puede calcular la impedancia real de la resistencia:

$$Z_{resistenciareal}(s) = \frac{Ls + R}{LCs^2 + RCs + 1} = \frac{R(1 + \frac{Ls}{R})}{LCs^2 + RCs + 1} \tag{2.2}$$

Sustituyendo $s = jw$ se obtiene la impedancia real de la resistencia:

$$Z_{resistenciareal}(jw) = \frac{jwL + R}{(1 - w^2LC) + jwRC} \tag{2.3}$$

Se puede realizar el mismo ejercicio con un capacitor, el cual cuenta con la siguiente

impedancia ideal [1]:

$$Z_{capacitorideal} = \frac{1}{jwC} \tag{2.4}$$

Como pasa con las resistencias, las capacitancias tienen elementos parásitos. Se presentan una resistencia y una inductancia en serie, ambas por las propiedades de los elementos conductores. Lo que se puede ver con claridad en la imagen 2.4.

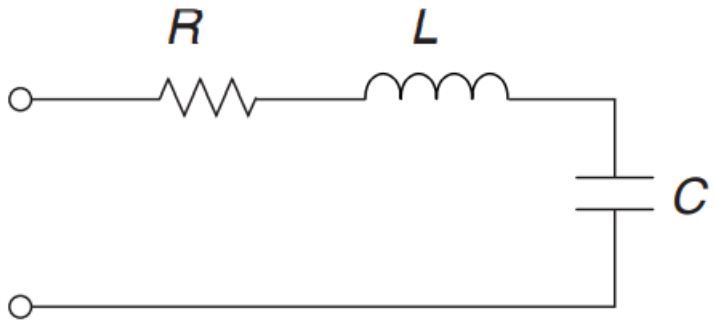


Figura 2.4: Capacitor C con su inductancia parásita en serie L y su resistencia parásita en serie R

Ahora se puede calcular la impedancia real de la capacitancia sumando los elementos en serie:

$$Z_{capacitorreal} = \frac{1}{jwc} + R + jwL = \frac{(1 - w^2LC) + jwRC}{jwC} \tag{2.5}$$

Otro componente básico de la electrónica es el inductor, que tiene la siguiente impedancia ideal [1]:

$$Z_{inductorideal} = jwL \tag{2.6}$$

Nuevamente, se tienen elementos parasitarios, uno de ellos corresponde a una resistencia relacionada con el material conductor utilizado para su construcción y una capacitancia producto del enrollado que posee el inductor. A modo de representación, esto se puede ver en la figura 2.5.

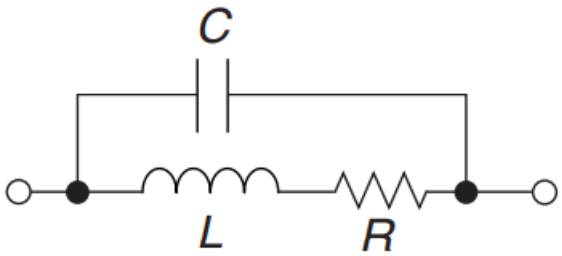


Figura 2.5: Inductor L con su capacitancia parásita en paralelo C y su resistencia parásita en serie R

Con lo anterior, se puede obtener la impedancia real de la inductancia:

$$Z_{inductanciareal} = \frac{jwL + R}{(1 - w^2LC) + jwRC} \tag{2.7}$$

El simple hecho de utilizar un cable para conectar dos elementos trae consigo la aparición de una resistencia y una inductancia parasitaria; la primera, aparece por las características conductivas del material a utilizar, recordando que ya no es ideal; la segunda, se debe a que existe corriente que circula por el cable, generando un campo alrededor de este. Se puede apreciar el efecto de este campo mediante el análisis de elementos finitos sobre una micro linea conductora en una *Printed Circuit Board* en la figura 2.6:

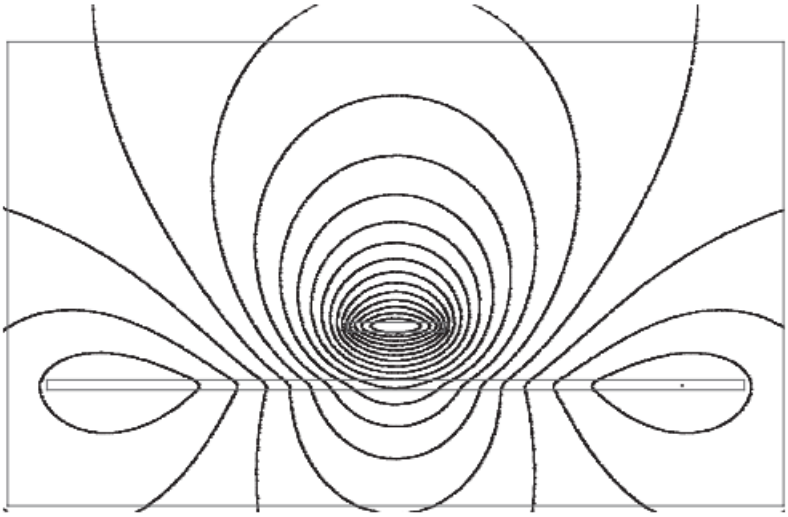


Figura 2.6: Análisis de elementos finitos para el campo producido por una micro linea conductora en una PCB

2.3. Sobre ESP

Si bien no se ahondará demasiado en la herramienta ESP, es necesario comentar y mostrar un poco sobre su funcionamiento, así se puede ligar esta herramienta a la necesidad de implementar un método de reducción de elementos parasitarios.

ESP es una herramienta de verificación formal, utilizando un método de verificación simbólica ([4] , [5]), lo que significa que se propagan símbolos y no 1 o 0. La herramienta producirá una ecuación, que puede ser fácilmente comparada, para comprobar el correcto funcionamiento del circuito.

Se presenta una serie de figuras para poder ejemplificar de mejor manera el funcionamiento de la herramienta. En la figura 2.7 se muestra un circuito cualquiera que no tiene valores asignados en sus entradas.

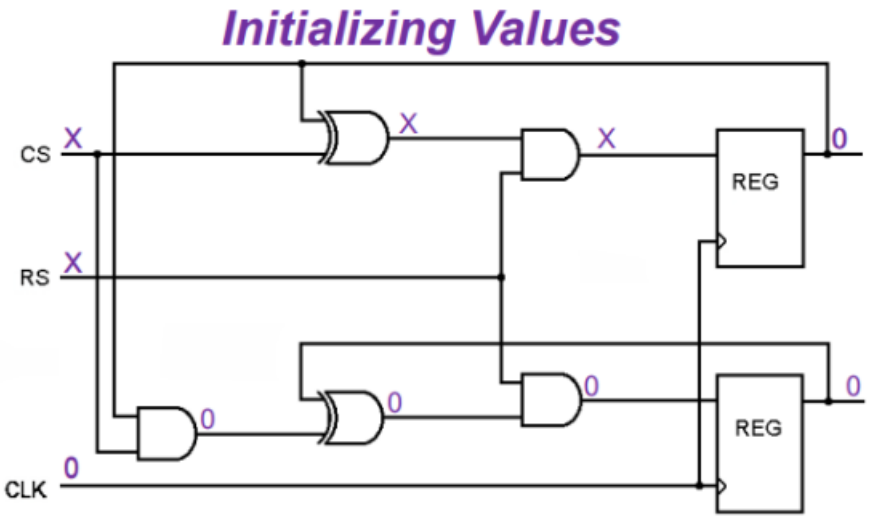


Figura 2.7: Previo a aplicar valores.

Como se mencionó previamente, en esta figura 2.8 se asigna un valor simbólico a una de las entradas del circuito.

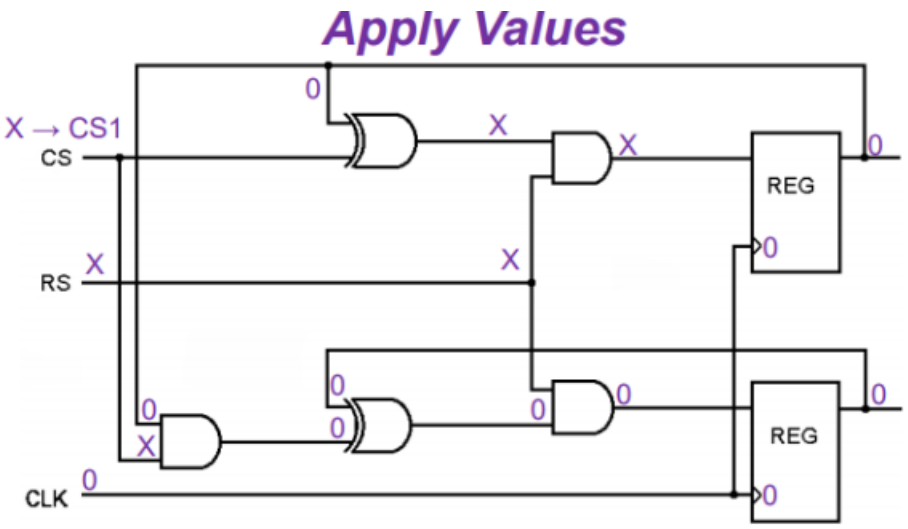


Figura 2.8: Se fija el valor simbólico CS1.

La entrada se propaga a lo largo del circuito como se puede notar en la imagen 2.9.

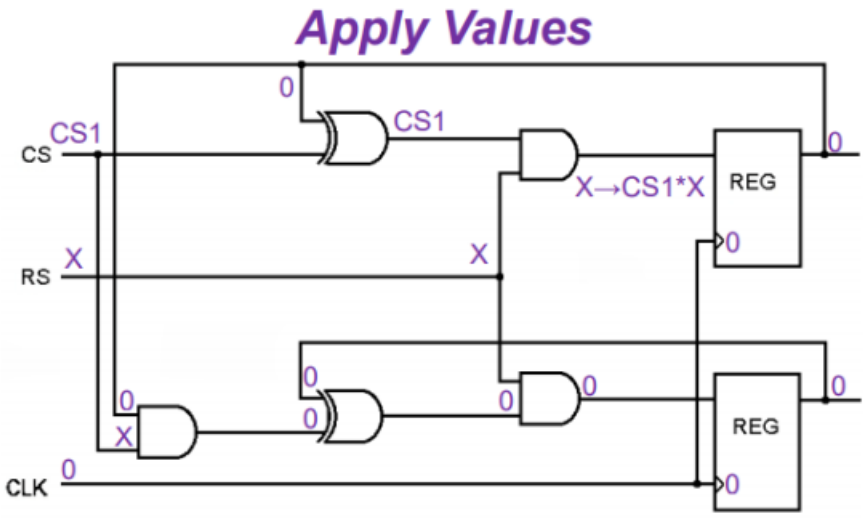


Figura 2.9: El valor se propaga a lo largo del circuito.

Para seguir con el proceso de verificación, se le asigna un valor simbólico a la otra entrada, como se muestra en la figura 2.10

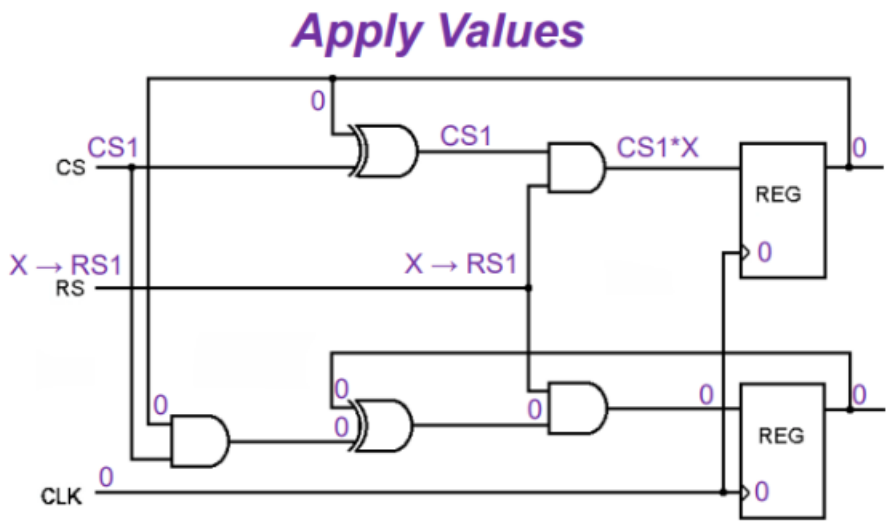


Figura 2.10: Se fija el valor simbólico RS1.

El valor asignado previamente se propaga, al igual que la entrada anterior, como se puede ver en la figura 2.11.

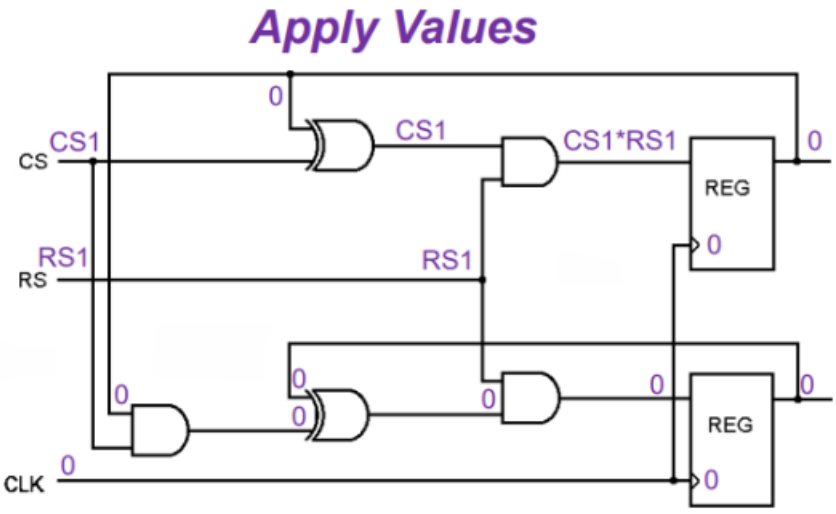


Figura 2.11: El nuevo valor se propaga a lo largo del circuito.

Con los valores ya propagados por le circuito, se produce el primer ciclo del reloj, entregando el resultado que se puede ver en la figura 2.12

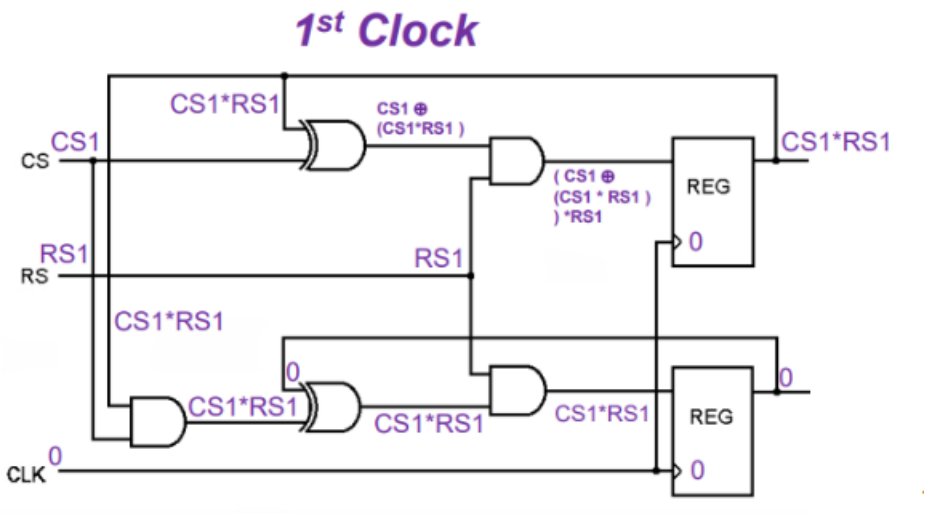


Figura 2.12: Final del primer ciclo de reloj.

En la figura 2.13 se muestra el inicio del segundo ciclo, aquí se aplican nuevos valores a las entradas del circuito.

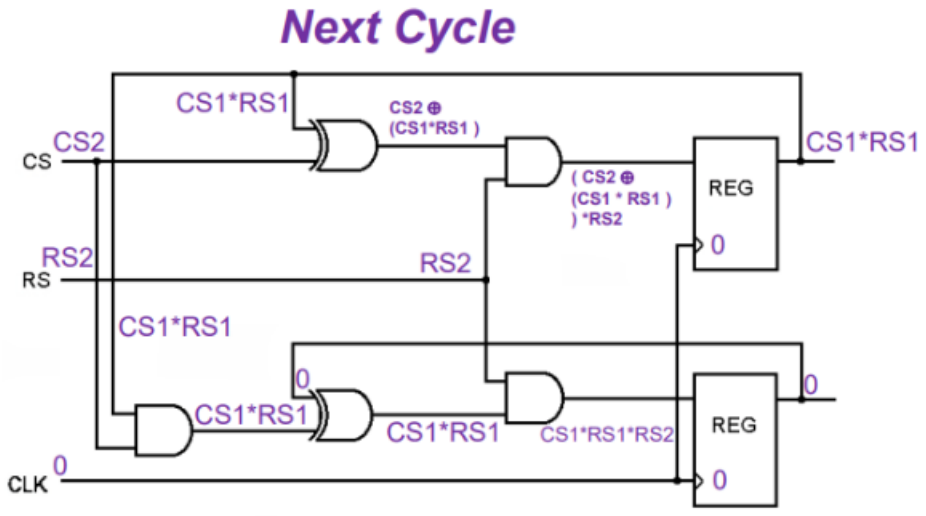


Figura 2.13: Siguiente ciclo de reloj.

Finalmente, en la figura 2.14, se pueden ver las ecuaciones en la salida del circuito al finalizar el segundo ciclo del reloj.

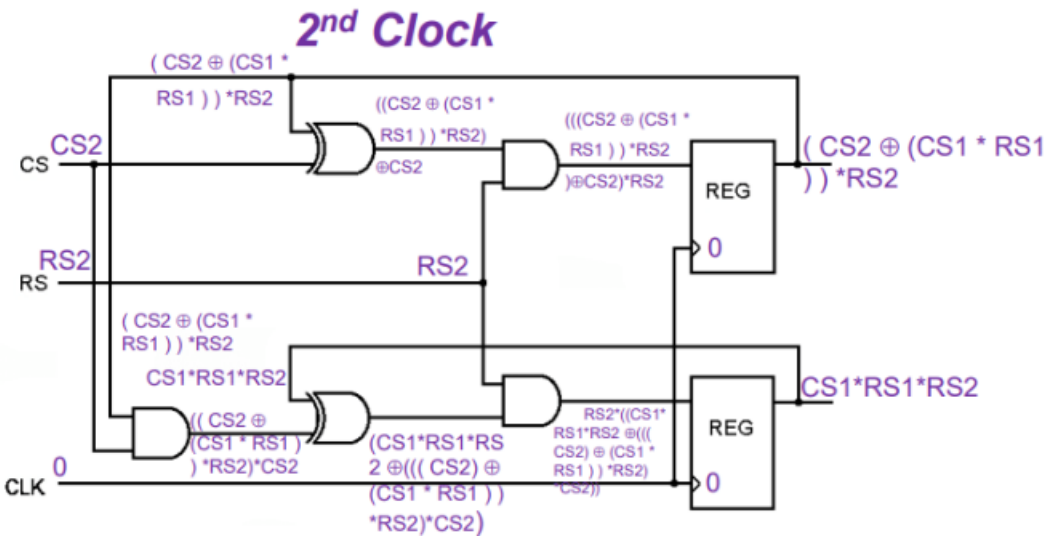


Figura 2.14: Valores finales después del segundo ciclo de reloj.

El proceso descrito se realiza eventualmente con dos circuitos, los cuales se quieren comparar. Si las ecuaciones obtenidas de ambos circuitos son equivalentes la verificación está completa, en caso contrario, se identifica la primera discrepancia y se muestra en pantalla para registro del usuario.

Junto a la verificación se realiza una simulación, que es desde dónde se entregan los resultados independiente de si la verificación es o no correcta, pero dicho proceso no se muestra en pantalla. Si el usuario necesita observar detenidamente los resultados y el *waveform* de la simulación, esta última se entrega en formato estándar (*.vcd* o *.fsdb*) para que se pueda acceder a ella sin problema.

Una de las principales ventajas que tiene ESP, es que puede realizar verificación entre circuitos presentados a un nivel de transistores, como lo sería si se analiza un circuito descrito en SPICE, o bien puede ser una representación comportacional, como lo sería en caso de ocupar Verilog. Además no es necesario que el circuito sea sintetizable para poder realizar su verificación. Esta última característica permite cierta holgura al momento de implementar la reducción parasitaria.

Para poder lidiar con la gran variedad de "sabores" en los que se presentan los circuitos, ESP genera una nueva versión del circuito de interés en formato *gate verilog*, el cual tiene por extensión *.gv.* bajo dicho formato, realiza las simulaciones correspondientes. Si bien se trata de un formato conocido, la versión que entrega la herramienta contiene elementos personalizados.

2.4. Modelos de Reducción de Orden

A continuación se presentan una serie de contenidos importantes para entender los modelos de reducción de orden, los cuales pueden ser utilizados para la eliminación de elementos parasitarios de la red. Pero antes de eso, es bueno recalcar cuatro puntos importantes que se deben cumplir al momento de realizar este procedimiento.

1. El modelo obtenido, después de realizar la reducción parasitaria, debe contar con algún criterio que permita corroborar la igualdad entre input-output con el modelo original.
2. El modelo obtenido debe ser de carácter pasivo, dado que interconecta sistemas pasivos, por lo que no puede producir energía por si solo.
3. La técnica escogida debe ser computacionalmente eficiente.
4. El modelo obtenido debe ser más rápido al momento de realizar simulaciones con él.

2.5. Moment Matching

Es importante estudiar este término debido a que es la base de los distintos métodos de reducción parasitaria presentes en la literatura.

Para poder definirlo se pueden expresar las conexiones utilizando análisis nodal modificado [16], lo que permite expresar el sistema en su transformada de Laplace, obteniendo una representación matricial de las admitancias:

$$\begin{cases} Gx(s) + sCx(s) = Bu(s) \\ y(s) = L^T x(s) \end{cases} \tag{2.8}$$

donde las matrices G, C $\in \mathbb{R}^{m\times m}$ son la matriz de conductancia y capacitancia, respectivamente. Por otra parte $u(s), y(s) \in \mathbb{R}^p$ representan los inputs y outputs del circuito. Además, la matriz $B \in \mathbb{R}^{m\times p}$ representa las conexiones entre los elementos. Finalmente el vector $x(s) \in \mathbb{R}^m$ corresponde a los voltajes nodales, corrientes y corrientes de inductores.

La relación entre entradas y salidas se relacionan de la siguiente manera:

$$y(s) = H(s)u(s) \tag{2.9}$$

donde

$$H(s) = L^T(G + sC)^{-1}B \tag{2.10}$$

con lo anterior, es posible calcular el momento de la siguiente manera:

$$M_k = \frac{1}{k!}\frac{\mathrm{d}^k H(s)}{\mathrm{d}s^k}|_{s=0} \tag{2.11}$$

para todo $k = 0, 1, 2, ...$

En términos de las matrices, el momento se puede definir como:

$$M_k = L^T(-G^{-1}C)^k G^{-1}B \tag{2.12}$$

para todo $k = 0, 1, 2, ...$

Con el momento definido se puede avanzar a los distintos tipos de modelos de reducción de orden.

2.6. Tipos de Modelos de Reducción de Orden

2.6.1. *Explicit Moment Matching Methods*

Dentro de los distintos tipos de métodos de reducción, este es uno de los primeros estudiados. Como lo dice su nombre, se basa en calcular de manera explícita el momento del circuito original. dada la relación que existe con la función de transferencia, se utiliza una aproximación de padé para calcular la función de transferencia del modelo reducido.

Dentro de este método la técnica más conocida es la *Asymptotic Waveform Evaluation* ([17], [18]).

2.6.2. *Implicit Moment Matching Methods*

El *moment matching* sigue siendo el principal criterio dentro de estos modelos de reducción de orden, pero dentro de esta categoría, el cálculo del *moment matching* no es explícito. Para esto, se realiza el siguiente cambio de variable $x = Q\hat{x}$ en la red original, con lo que se obtiene el siguiente set de ecuaciones:

$$\begin{cases} W^T G Q\hat{x}(s) + sW^T C Q\hat{x}(s) = W^T B u(s) \\ \hat{y}(s) = L^T Q\hat{x}(s) \end{cases} \tag{2.13}$$

las matrices del modelo reducido son:

$$\hat{G} = W^T GQ,\ \hat{C} = W^T CQ,\ \hat{B} = W^T B,\ \hat{L} = W^T L \tag{2.14}$$

Las matrices W y Q son matrices de proyección, que cumplen con ser de rango completo. Además, se satisface la siguiente relación $W^T Q = I$, donde $I \in \mathbb{R}^{m\times n}$ es la matriz identidad. Si es que $W = Q$, la ecuación 2.13 es una proyección ortogonal conocida como Proyección de Galerskin, de no cumplir dicha condición, se trata de una proyección bi-ortogonal.

Las matrices W y Q se construyen utilizando el concepto de Subespacio de Krylov [19]. Suponiendo que contamos con una matriz A y un vector c, el subespacio de orden n de Krylov se define como:

$$K_n(A, c) = span\{c, Ac, A^2c, ..., A^{n-1}c\} \tag{2.15}$$

En especifico para, solucionar este problema, se seleccionan dos matrices: A y R que se relacionan con las ecuaciones planteadas del sistema en la ecuación 2.8, por lo que se tiene el siguiente subespacio de Krylov:

$$K_n(A, R) = span\{R, AR, A^2R, ..., A^{n-1}R \tag{2.16}$$

W y Q se forman de tal manera que sus columnas forman los vectores base del subespacio de Krylov. De esta manera, el modelo reducido de la ecuación 2.13 contará con momentos iguales al del sistema original, pero solamente hasta cierto orden.

La construcción de estas matrices varía dependiendo si se trata de una proyección ortogonal o biortogonal. Lamentablemente, si se utiliza la proyección biortogonal no se puede garantizar que el modelo reducido cumpla con que se mantenga la pasividad, por lo que no se puede llevar a cabo un modelo de reducción de orden utilizando este tipo de proyección.

Uno de los métodos de reducción de orden más conocido es PRIMA, que utiliza *Implicit Moment Matching* [20]. Es muy común que muchos de los nuevos métodos propuestos para la reducción de orden se comparen con los resultados obtenidos desde PRIMA.

2.6.2.1. PRIMA

Dada su popularidad, parece prudente explicar de manera breve en qué consiste este método de reducción de orden. *Passive Reduced-Order Interconnect Macromodeling Algorithm* utiliza *implicit moment matching* como base en su funcionamiento.

Mediante proyección ortogonal, genera las matrices del sistema reducido de la ecuación 2.14 por lo que, en este caso, $W = Q$ y $Q^TQ = I$. La matriz Q se contruye de tal manera que sus columnas forman vectores base del subespacio de Krylov de orden q, donde $A = -G^{-1}C$ y $R = G^{-1}B$. Por lo que se tiene:

$$colspan(Q) = K_q(-G^{-1}C, G^1B) \tag{2.17}$$

donde $Q \in \mathbb{R}^{m \times pq}$, dado que $W = Q$, se obtienen las matrices reducidas producidas por PRIMA:

$$\hat{G} = Q^TGQ \tag{2.18}$$
$$\hat{C} = Q^TCQ \tag{2.19}$$
$$\hat{B} = Q^TB \tag{2.20}$$
$$\hat{L} = Q^TL \tag{2.21}$$

ahora el orden del modelo pq, este coincidirá con los primeros q momentos del modelo original. Esto cambia si se da el caso de que la red solamente contenga elementos RC, con lo que se tendría una coincidencia en los primeros $2q$ momentos.

El cálculo de las columnas de la matriz Q se hace con el proceso de Arnoldi [21], que corresponde a un procedimiento de ortogonalización.

2.6.3. Truncated Balanced Realization Method

Al utilizar como base este método, se deja de lado el moment matching, y se hace uso de la teoría de control. La idea principal es identificar y descartar los estados controlables y observables más débiles ([22], [23]).

Los estados eliminados también cumplen con ser difíciles de excitar y, una vez excitados, no contribuyen de manera significativa a las salidas del circuito. La eliminación se debe realizar con cuidado, debido a que no es posible saber a priori cuántos estados se pueden eliminar sin afectar los resultados numéricos que aseguran la similitud entre los input-output del modelo original con el modelo reducido.

Capítulo 3

Distintas representaciones de circuitos

Como fue mencionado previamente, ESP tiene la capacidad de trabajar con una gran cantidad de representaciones de circuitos distintos. Es más, el proyecto relacionado con la eliminación de elementos parasitarios nace ante la necesidad de incluir otro formato a esta lista. Dentro de los formatos más conocidos con los que trabaja ESP se encuentran SPICE, Verilog y SystemVerilog.

En particular, son dos los formatos que son importantes de estudiar y sobres los cuales más se va a comentar durante el desarrollo de este documento; el primero, corresponde al problemático formato con extensión *.spf*, que es una modificación del SPICE. Este se obtiene una vez el proceso de diseño se termina, representando al circuito que finalmente será impreso. Es aquí donde los elementos parasitarios se pueden apreciar por primera vez; el segundo, corresponde a Gate Verilog, cuando se quiere llevar a cabo la verificación de algún circuito. ESP genera un archivo de este tipo para realizar las respectivas simulaciones.

3.1. Algunas características de un archivo *.spf*

Un circuito representado en este tipo de formato, que finalmente es una modificación del formato SPICE, se encuentran dividido en pequeños sub-circuitos, donde en cada uno de estos se especifican los elementos que perteneces a él y cada una de las conexiones tanto internas como aquellas que comparte con otro sub-circuito. Por ejemplo, si se tiene un Circuito A, el cual tiene ciertos *inputs* y *outputs* como en la figura 3.1.

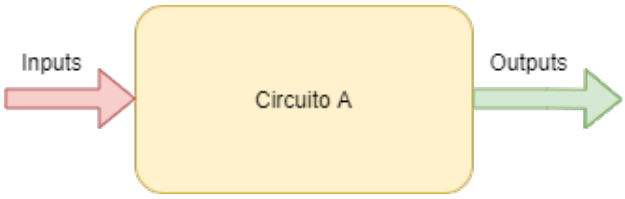


Figura 3.1: Un circuito simple representado por un bloque A.

Se podría dividir en los sub-circuitos como se muestra en la figura 3.2

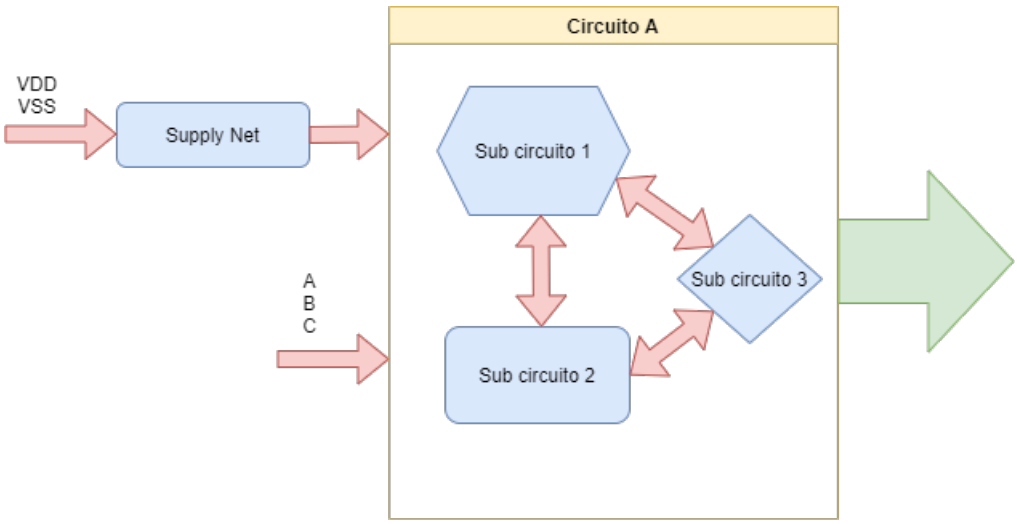


Figura 3.2: Se representa a los distintos circuitos que conforman el bloque A.

Aquí se hace una de las primeras definiciones importantes para el desarrollo del proyecto, que es la llamada *supply RC net*, que si bien se define al igual que cualquier otro sub-circuito dentro del archivo *.spf*, tiene la particularidad de que representa a las conexiones entre las fuentes de voltaje definidas, en este caso **VDD** y **VSS**, y los respectivos terminales de los transistores presentes dentro del circuito. Esto teniendo en cuenta que el comportamiento de los transistores depende de los voltajes aplicados en sus terminales.

Dentro de estos sub-circuitos, es común encontrar copias de ciertos elementos, por ejemplo, si se definió a **vcc** como una fuente de voltaje, dentro del módulo **NET vcc**, se tiene algo como lo que aparece en la figura 3.3:

```
R7_1207 vcc_47 Mqp3_SRC
R7_1208 vcc_48 vcc_65
R7_1209 vcc_48 Mqp4_SRC
R7_1210 vcc_49 vcc_66
R7_1211 vcc_49 Mqp4__2_SRC
R7_1212 vcc_50 vcc_66
R7_1213 vcc_50 Mqp3__2_SRC
R7_1214 vcc_51 vcc_67
R7_1215 vcc_51 Mqp3__3_SRC
R7_1216 vcc_52 vcc_67
R7_1217 vcc_52 Mqp4__3_SRC
R7_1218 vcc_53 vcc_69
R7_1219 vcc_53 Mqp4__4_SRC
R7_1220 vcc_54 vcc_69
R7_1221 vcc_54 Mqp3__4_SRC
R7_1222 vcc_55 vcc_71
R7_1223 vcc_55 Mqp3__5_SRC
R7_1224 vcc_56 vcc_71
R7_1225 vcc_56 Mqp4__5_SRC
```

Figura 3.3: Elementos que pertenecen al módulo NET vcc

Aquí, tanto la fuente de voltaje **vcc**, como los terminales *source* de los transistores **Mqp3** y **Mqp4**, se encuentran en más de una ocasión, siendo que realmente representan al mismo elemento. Esto va a ser una problemática estudiada durante el desarrollo del proyecto, lo cual será detallado más adelante. Cabe destacar que existen muchas más conexiones dentro del módulo, 834 elementos solamente en NET vcc, teniendo en cuenta que solamente este circuito presenta 25 módulos; se trata de una gran cantidad de elementos por circuito.

Los elementos que se encuentran dentro de estos módulos son, en su totalidad, elementos resistivos y capacitivos, especificando cual es la conexión a cada uno de sus dos terminales. Además, se entrega información sobre sus características físicas como lo son el largo, ancho y el valor de la resistencia o capacitancia.

Luego de la definición de cada uno de estos sub-circuitos, se encuentra una sección donde se entregan las características de cada uno de los transistores que forman parte del circuito, incluso de aquellos que se encuentran duplicados. Aquí, la cantidad de detalle con la que se entrega la información respecto a sus propiedades físicas es muy elevada, donde se tienen largo, ancho, alto, niveles de voltaje, entre otras.

3.2. Algunas características de un archivo *.gv file*

Antes de la definición del módulo, se pueden ver las distintas características de los elementos que forman parte de este circuito en la figura 3.4:

```
(* esp_attribute = "imp_top", INV_imp=4*)
(* esp_attribute = "n" , INNODEFAULT=1 *)
(* esp_attribute = "p" , INNODEFAULT=1 *)
```

Figura 3.4: Características de los elementos dentro del circuito.

Luego, inicia la definición del módulo, que en este caso corresponde simplemente a un inversor, como se puede observar en la figura 3.5

```
module INV_imp (A, Z);
   parameter LP=[redacted] WP=[redacted], LN=[redacted], WN=[redacted];
   input A;
   output Z;
   wire a, a_1, a_11, b, b_1, b_11, c, c_1, d, d_1;
```

Figura 3.5: Inicio de la definición del módulo.

Se puede notar la similitud a cómo sería definido un módulo en formato Verilog, donde se definen las entradas, salidas y "cables" de ser necesario.

Luego, continuando con la definición del módulo, se pueden ver las distintas fuentes de voltaje que interfieren en el funcionamiento del circuito, las cuales se pueden ver en la figura 3.6. Estas fuentes son definidas por el usuario mediante uno de los comandos provistos por ESP. La manera de hacerlo será especificada más adelante en este mismo documento.

```
(* esp_set_voltage=1.000000e+00 *)
  supply1 VDD;
(* esp_set_voltage=0.000000e+00 *)
  supply0 VSS;
```

Figura 3.6: Fuentes de voltajes definidas para el funcionamiento del circuito.

A continuación, se tiene la definición de todas las resistencias y capacitancias del circuito. Aquí, primero se especifica cuáles son las conexiones de cada uno de sus terminales y sus propiedades físicas, como pueden ser su largo, ancho y sus respectivos valores resistivos y capacitivos, como en la figura 3.7:

```
(*esp_attribute = "Resistor",R=1.000000e+00 *)
   tran R_1(VDD, a);
   supply0 innovss; // For use by capacitor model
(* esp_attribute = "Capacitor", c=1.000000e-07 *)
 nmos C_1(a, VDD, innovss);
(*esp_attribute = "Resistor",R=1.000000e+00 *)
   tran R_11(a, a_1);
(*esp_attribute = "Resistor",R=1.000000e+00 *)
   tran R_111(a_1, a_11);
(*esp_attribute = "Resistor",R=1.000000e+00 *)
   tran R_2(VSS, b);
(* esp_attribute = "Capacitor", c=1.000000e-07 *)
 nmos C_2(b, VSS, innovss);
(*esp_attribute = "Resistor",R=1.000000e+00 *)
   tran R_22(b, b_1);
(*esp_attribute = "Resistor",R=1.000000e+00 *)
   tran R_222(b_1, b_11);
(*esp_attribute = "Resistor",R=1.000000e+00 *)
   tran R_3(VDD, c);
(*esp_attribute = "Resistor",R=1.000000e+00 *)
   tran R_33(c, c_1);
(* esp_attribute = "Capacitor", c=1.000000e-07 *)
 nmos C_3(c, VDD, innovss);
(*esp_attribute = "Resistor",R=1.000000e+00 *)
   tran R_4(VSS, d);
(*esp_attribute = "Resistor",R=1.000000e+00 *)
   tran R_44(d, d_1);
(* esp_attribute = "Capacitor", c=1.000000e-07 *)
 nmos C_4(d, VSS, innovss);
```

Figura 3.7: Resistencias y capacitancias que forman parte del circuito.

Los últimos elementos que se definen son los transistores junto con sus propiedades físicas y el nombre de cada una de sus terminales, como se observa en la figura 3.8:

```
(* esp_attribute = "p" , w=WP , l=LP  *)
  tranif0  MI1 (Z, c_1, A);
(* esp_attribute = "n" , w=WN , l=LN  *)
  tranif1  MI2 (Z, d_1, A);
```

Figura 3.8: Transistores presentes dentro del inversor.

A este inversor se le agregaron, intencionalmente, elementos capacitivos y resistivos que permiten emular la aparición de una *supply RC net* en este archivo, la cual está presente para ambas fuentes **VDD** y **VSS**. Esto se hizo en la definición del inversor en el archivo SPICE como en la figura 3.9, el cual fue verificado con su símil en Verilog.

```
.SUBCKT INV Z A LP=0.18u WP=2.40u LN=0.18u WN=1.20u

MI1 Z A c_1 a_11 P W=WP L=LP
MI2 Z A d_1 b_11 N W=WN L=LN

R_1 VDD a 1
C_1 VDD a 1.0e-19
R_11 a a_1 1
R_111 a_1 a_11 1
R_2 VSS b 1
C_2 VSS b 1.0e-19
R_22 b b_1 1
R_222 b_1 b_11 1

R_3 VDD c 1
R_33 c c_1 1
C_3 VDD c 1.0e-19
R_4 VSS d 1
R_44 d d_1 1
C_4 VSS d 1.0e-19

.ENDS
```

Figura 3.9: Circuito inversor representado en SPICE, al cual se le agregaron elementos para simular una *supply RC net*.

En el siguiente capitulo quedarán claros todos los conceptos respecto a la utilización de ESP, al menos en lo que respecta la verificación de circuitos.

Capítulo 4

Experimentación y resultados previos a la implementación

4.1. Encontrando el Problema dentro de ESP

Si bien es sabido que la herramienta tiene problemas para procesar los archivos de formato *.spf*, no se tiene certeza de cuáles son los factores que influyen en el mal comportamiento de la herramienta.

Otra cosa que es importante, previo a plantear cualquier mejora, es tener claridad en el momento del proceso de verificación en el que se presentan las anomalías que derivan, eventualmente, en un error. Con lo anterior, es posible tener claridad cuáles son las funciones implementadas dentro del código que deben ser intervenidas.

4.1.1. Utilizando ESP

Si bien se hizo una breve explicación de cómo funcionaba ESP, es necesario entender cómo realizaría un usuario la verificación de un circuito común.

Para este ejemplo, supongamos que se quiere verificar un circuito simple como lo es un inversor. Primero, se necesitan dos caracterizaciones distintas de dicho circuito, en este caso se tiene un archivo verilog **test.v**, el cual se puede ver en la figura 4.1, y otro archivo en formato spice **inv.sp**, que se muestra en la figura 4.2. Cabe destacar que no es necesario que ambos archivos tengan el mismo nombre, lo importante es que los nombres de los módulos dentro de ellos sí sean iguales. Los comandos para leer ambos archivos son *read_verilog netlist.v* y *read_spice netlis.sp*.

```
module INV (a, z);
  output z;
  input  a;
   reg   z;

  always @(a)
    z = ~a;
endmodule
```

Figura 4.1: Representación de un circuito inversor en Verilog.

```
.SUBCKT INV Z A LP=0.18u WP=2.40u LN=0.18u WN=1.20u
.CONNECT A Z
MI1 Z A VDD VDD P W=WP L=LP
MI2 Z A VSS VSS N W=WN L=LN
.ENDS
```

Figura 4.2: Representación de un circuito inversor en Spice.

Uno de estos archivos debe ser declarado como la *referencia* respecto a la cual se hace la verificación. Dado que en verilog resulta muy simple comprobar si el circuito funciona correctamente, en este caso el archivo **inv.v** se declara como referencia. El otro circuito debe ser declarado como *implementación*. Esto se define dentro del mismo comando para leer los archivos que, finalmente, queda *read_verilog -r test.v* y *read_spice -i inv.sp*, para el archivo verilog y spice, respectivamente.

A continuación se debe especificar cuales son las fuentes de voltajes presentes en el circuito, en este caso en especifico son **VDD** y **VCC**, el comando correspondiente para esto es *set_supply_net supplyname*

Luego, simplemente se debe agregar el comando de verificación **verify**.

En general, todo lo anterior se agrega en un archivo de extensión *.tcl*, principalmente porque resulta la forma más fácil de realizar múltiples verificaciones y replicarlos en caso de ser necesarios. El archivo creado para esta prueba queda como se muestra en la figura 4.3

```
set sh_continue_on_error true
# Set style which generates single testbenches
set testbench_style symbolic

# Prepare environment to check design
read_verilog -r test.v
read_spice -i inv.sp
set_supply_net -i -type real -logic 1 VDD
set_supply_net -i -type real -logic 0 VSS

verify

quit
```

Figura 4.3: Contenido del archivo .tcl para la verificación de un circuito inversor.

La ejecución se realiza directamente en un *command shell*, no cuenta con una interfaz de usuario para la ejecución de sus comandos, pero sí cuenta con variadas herramientas para la revisión de las simulaciones que se estimen convenientes. Realizar esta simple verificación se ve como sigue, en la figura 4.4:

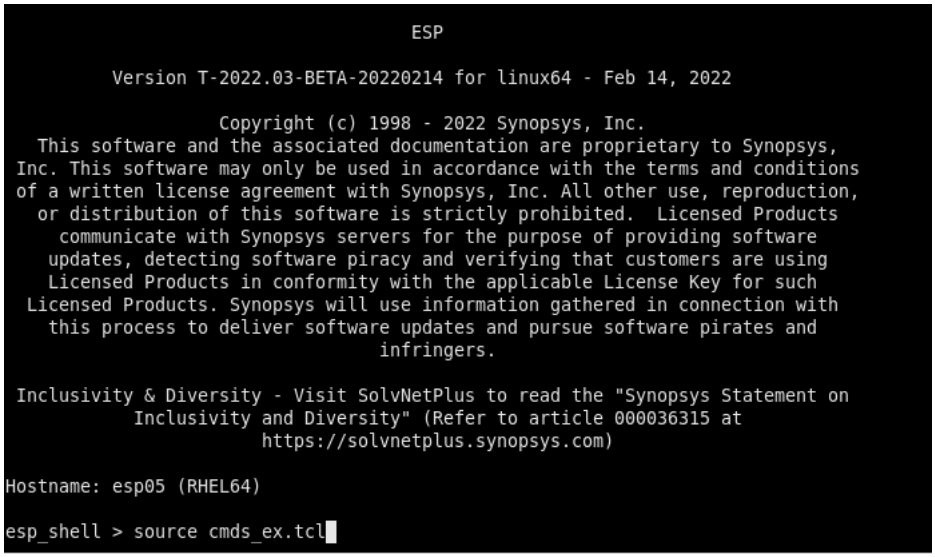


Figura 4.4: ESP en ejecución ingresando el archivo .tcl para realizar la verificación del circuito inversor.

Aquí, la verificación se realizó con éxito, lo que quiere decir, en pocas palabras, que los circuitos descritos en ambos archivos hacen exactamente lo mismo.

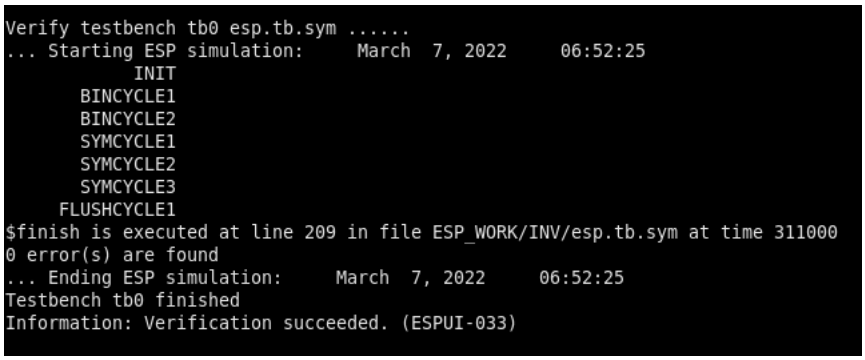


Figura 4.5: Finaliza la verificación con éxito.

Durante el proceso de verificación se crea una carpeta que da acceso a archivos específicos que fueron utilizados para las pruebas necesarias. En particular, se puede revisar el *testbench* generado y el archivo *.gv*, con el cual se realizan todas las simulaciones necesarias para comprobar el funcionamiento del circuito. Para este caso particular, este último se puede observar en la siguiente figura 4.6

```
(* esp_attribute = "imp_top", INV_imp=4*)

(* esp_attribute = "n" , INNODEFAULT=1 *)

(* esp_attribute = "p" , INNODEFAULT=1 *)

(* esp_db_module *)
(* esp_spice_subckt_name="INV" *)
 (* esp_attribute = "rcdelays" *)
module INV_imp (A, Z);
   parameter LP=1.800000e-07, WP=2.400000e-06, LN=1.800000e-07, WN=1.200000e-06;
   input A;
   inout Z;
   wire VDD, VSS;

 (* esp_attribute = "p" , w=WP , l=LP  *)
   tranif0  MI1 (Z, VDD, A);
 (* esp_attribute = "n" , w=WN , l=LN  *)
   tranif1  MI2 (Z, VSS, A);
endmodule
```

Figura 4.6: Representación de un inversor en formato *.gv*, el cual es utilizado por ESP para realizar las simulaciones durante el proceso de verificación.

4.1.2. Problemas al verificar post-layout netlist

Para las pruebas iniciales relacionadas con la búsqueda particular de las fallas dentro del proceso de verificación, se cuenta con un solo circuito, pero que cuenta con todos los elementos que un usuario podría llegar a tener en su *.spf netlist*.

El primero de los problemas no tarda en aparecer, debido a uno de los componentes presentes en la lista de conexión, que corresponde a una especie de transistores que no interfieren en el funcionamiento del circuito, conocidos como PODE (*Polysilicon On Difussion Edge*). están incluidos para "proteger" al resto de los elementos funcionales del circuito durante el proceso de fabricación de estos, de hecho su terminal *Gate* se encuentra completamente desconectado.

Cada uno de los fabricantes cuenta con un archivo, conocido como EDM, donde definen las características de los elementos que utiliza dentro de cada circuito. Dado que los PODE no están pensados como un elemento funcional en el comportamiento del circuito, no se encuentran definidos dentro de este archivo. Por lo anterior, en el momento en que ESP trata de crear la representación en formato *.gv*, no tiene noción de como interpretar el elemento, provocando el fin de la verificación, acompañado del mensaje en la figura 4.7:

```
Warning: SPICE instance Xld_X1 missing subckt npode_lvt_mac in file HDN6PLVT08_LVLDBUFE0_IV4Y2_2.spf:7217 (ESPSPC-212)
Warning: SPICE instance Xld_X10 missing subckt npode_lvt_mac in file HDN6PLVT08_LVLDBUFE0_IV4Y2_2.spf:7218 (ESPSPC-212)
Warning: SPICE instance Xld_X11 missing subckt npode_lvt_mac in file HDN6PLVT08_LVLDBUFE0_IV4Y2_2.spf:7219 (ESPSPC-212)
Warning: SPICE instance Xld_X12 missing subckt npode_lvt_mac in file HDN6PLVT08_LVLDBUFE0_IV4Y2_2.spf:7220 (ESPSPC-212)
Warning: SPICE instance Xld_X13 missing subckt npode_lvt_mac in file HDN6PLVT08_LVLDBUFE0_IV4Y2_2.spf:7221 (ESPSPC-212)
Warning: SPICE instance Xld_X14 missing subckt ppode_lvt_mac in file HDN6PLVT08_LVLDBUFE0_IV4Y2_2.spf:7222 (ESPSPC-212)
Warning: SPICE instance Xld_X15 missing subckt ppode_lvt_mac in file HDN6PLVT08_LVLDBUFE0_IV4Y2_2.spf:7223 (ESPSPC-212)
Warning: SPICE instance Xld_X16 missing subckt ppode_lvt_mac in file HDN6PLVT08_LVLDBUFE0_IV4Y2_2.spf:7224 (ESPSPC-212)
Warning: SPICE instance Xld_X17 missing subckt ppode_lvt_mac in file HDN6PLVT08_LVLDBUFE0_IV4Y2_2.spf:7225 (ESPSPC-212)
Warning: SPICE instance Xld_X18 missing subckt ppode_lvt_mac in file HDN6PLVT08_LVLDBUFE0_IV4Y2_2.spf:7226 (ESPSPC-212)
Warning: SPICE instance Xld_X19 missing subckt ppode_lvt_mac in file HDN6PLVT08_LVLDBUFE0_IV4Y2_2.spf:7227 (ESPSPC-212)
Warning: SPICE instance Xld_X2 missing subckt npode_lvt_mac in file HDN6PLVT08_LVLDBUFE0_IV4Y2_2.spf:7228 (ESPSPC-212)
Warning: SPICE instance Xld_X3 missing subckt npode_lvt_mac in file HDN6PLVT08_LVLDBUFE0_IV4Y2_2.spf:7239 (ESPSPC-212)
Warning: SPICE instance Xld_X4 missing subckt npode_lvt_mac in file HDN6PLVT08_LVLDBUFE0_IV4Y2_2.spf:7250 (ESPSPC-212)
Warning: SPICE instance Xld_X5 missing subckt npode_lvt_mac in file HDN6PLVT08_LVLDBUFE0_IV4Y2_2.spf:7261 (ESPSPC-212)
Warning: SPICE instance Xld_X6 missing subckt npode_lvt_mac in file HDN6PLVT08_LVLDBUFE0_IV4Y2_2.spf:7272 (ESPSPC-212)
Warning: SPICE instance Xld_X7 missing subckt npode_lvt_mac in file HDN6PLVT08_LVLDBUFE0_IV4Y2_2.spf:7283 (ESPSPC-212)
Warning: SPICE instance Xld_X8 missing subckt npode_lvt_mac in file HDN6PLVT08_LVLDBUFE0_IV4Y2_2.spf:7294 (ESPSPC-212)
Warning: SPICE instance Xld_X9 missing subckt npode_lvt_mac in file HDN6PLVT08_LVLDBUFE0_IV4Y2_2.spf:7301 (ESPSPC-212)
write_testbench level_shift_nopode_spf_real
Information: Generating testbench level shift nopode spf real.ltb (ESPUI-292)
verify -matched_designs HDN6PLVT08_LVLDBUFE0_IV4Y2_2
Error: Check design failed on testbench id tb0. (ESPUI-239)
Information: Use command "report_log" for more information. (ESPUI-078)
```

Figura 4.7: Mensajes de advertencia en referencia a los elementos PODE dentro del circuito.

Como sugiere el mensaje de información, se puede utilizar el comando *report_log* para tener más detalle respecto a los problemas que se presentaron, lo que arroja lo siguiente en la figura 4.8.

```
Error-[URMI] Module "npode_lvt_mac" is missing definition for instance Xld_X1 in level_shift_nopode_spf_real.gv at line number 14244
Error-[URMI] Module "npode_lvt_mac" is missing definition for instance Xld_X10 in level_shift_nopode_spf_real.gv at line number 14245
Error-[URMI] Module "npode_lvt_mac" is missing definition for instance Xld_X11 in level_shift_nopode_spf_real.gv at line number 14246
Error-[URMI] Module "npode_lvt_mac" is missing definition for instance Xld_X12 in level_shift_nopode_spf_real.gv at line number 14247
Error-[URMI] Module "npode_lvt_mac" is missing definition for instance Xld_X13 in level_shift_nopode_spf_real.gv at line number 14248
Error-[URMI] Module "npode_lvt_mac" is missing definition for instance Xld_X2 in level_shift_nopode_spf_real.gv at line number 14255
Error-[URMI] Module "npode_lvt_mac" is missing definition for instance Xld_X3 in level_shift_nopode_spf_real.gv at line number 14276
Error-[URMI] Module "npode_lvt_mac" is missing definition for instance Xld_X4 in level_shift_nopode_spf_real.gv at line number 14297
Error-[URMI] Module "npode_lvt_mac" is missing definition for instance Xld_X5 in level_shift_nopode_spf_real.gv at line number 14318
Error-[URMI] Module "npode_lvt_mac" is missing definition for instance Xld_X6 in level_shift_nopode_spf_real.gv at line number 14339
Error-[URMI] Module "npode_lvt_mac" is missing definition for instance Xld_X7 in level_shift_nopode_spf_real.gv at line number 14360
Error-[URMI] Module "npode_lvt_mac" is missing definition for instance Xld_X8 in level_shift_nopode_spf_real.gv at line number 14381
Error-[URMI] Module "npode_lvt_mac" is missing definition for instance Xld_X9 in level_shift_nopode_spf_real.gv at line number 14394
Error-[URMI] Module "ppode_lvt_mac" is missing definition for instance Xld_X14 in level_shift_nopode_spf_real.gv at line number 14249
Error-[URMI] Module "ppode_lvt_mac" is missing definition for instance Xld_X15 in level_shift_nopode_spf_real.gv at line number 14250
Error-[URMI] Module "ppode_lvt_mac" is missing definition for instance Xld_X16 in level_shift_nopode_spf_real.gv at line number 14251
Error-[URMI] Module "ppode_lvt_mac" is missing definition for instance Xld_X17 in level_shift_nopode_spf_real.gv at line number 14252
Error-[URMI] Module "ppode_lvt_mac" is missing definition for instance Xld_X18 in level_shift_nopode_spf_real.gv at line number 14253
Error-[URMI] Module "ppode_lvt_mac" is missing definition for instance Xld_X19 in level_shift_nopode_spf_real.gv at line number 14254
esp ERROR:  parsing error.
```

Figura 4.8: Detalle de los errores durante el proceso de verificación.

En definitiva, no se puede analizar de manera correcta el circuito debido a la presencia de los elementos que no cuentan con una definición funcional.

Para poder capear este problema, se pueden agregar características "default" a los elementos PODE mediante un comando de ESP. En este caso en particular el circuito tiene dos tipos distintos de PODE, uno representado como un **pmos** y otro como un **nmos**, con nombres **ppode_lvt_mac** y **npode_lvt_mac**, respectivamente. Para dar propiedades a uno de ellos, por ejemplo a **ppode_lvt_mac**, se debe utilizar el siguiente comando "default" *set_device_model -i -type pmos ppode_lvt_mac*. Lo anterior, se realiza para ambos elementos y nuevamente se inicia el proceso de verificación.

En esta ocasión sí se puede crear el archivo *.gv*, por lo que se inicia la simulación, pero

esta falla, ESP reporta nueve errores durante le proceso de verificación, como se muestra en la figura 4.9.

```
... Starting ESP simulation:     April  7, 2022     10:02:12
           INIT
    BINCYCLE1_1
    BINCYCLE1_2
    BINCYCLE2_1
    BINCYCLE2_2
    BINCYCLE3_1
    BINCYCLE3_2
    BINCYCLE4_1
    BINCYCLE4_2
    BINCYCLE5_1
    BINCYCLE5_2
    BINCYCLE6_1
    BINCYCLE6_2
    BINCYCLE7_1
    BINCYCLE7_2
    BINCYCLE8_1
    BINCYCLE8_2
    SYMCYCLE1_1
    SYMCYCLE1_2
    SYMCYCLE2_1
    SYMCYCLE2_2
    SYMCYCLE3_1
    SYMCYCLE3_2
    SYMCYCLE4_1
    SYMCYCLE4_2
    SYMCYCLE5_1
    SYMCYCLE5_2
9 error(s) are found
... Ending ESP simulation:     April  7, 2022     10:02:15
Testbench tb0 finished
Error: Verification failed. (ESPUI-031)
```

Figura 4.9: Se muestra la cantidad de ciclos realizados, la cantidad de errores encontrados y reporta que la verificación falló.

Al igual que en el caso anterior, se puede utilizar el comando *report_log* para revisar en más detalle la causa del error en la verificación. Ahora el error corresponde a una discrepancia entre los valores entregados por el circuito que representa la referencia y el de implementación; esto se puede ver en la figura 4.10:

```
Start of cycle            INIT
esp Info:  DC initialization complete.

Start of cycle      BINCYCLE1_1
============ Input Signal Values ==============
                A = 0
               EN = 0
              VDD = 1
             VDDI = 1
              VSS = 0
================================================
== Checking outputs                  355
Checking X
     ref  = 0
     imp  = x
```

Figura 4.10: Se muestra uno de los errores encontrados durante la simulación.

Con los vectores aplicados se obtiene un valor igual a cero en el circuito fijado como referencia, mientras que en la implementación se tiene una valor de X. Este comportamiento

se puede dar debido a que los transistores tipo PODE tiene su terminal *Gate* desconectado y, mediante la definición realizada utilizando el comando *set_device_model*, se le entregaron características funcionales. Este no tiene realmente una funcionalidad dentro del circuito, pero esto ESP no tiene forma de "saberlo". Entonces, se recurre a otro comando que entrega ESP, en esta ocasión para eliminar dichos elementos PODE del circuito, el que corresponde a *remove_instance*. Este comando se debe ejecutar para cada uno de los PODE presentes en el circuito, lo que se puede observar en la figura 4.11.

```
remove_instance -i Xld_X1
remove_instance -i Xld_X10
remove_instance -i Xld_X11
remove_instance -i Xld_X12
remove_instance -i Xld_X13
remove_instance -i Xld_X14
remove_instance -i Xld_X15
remove_instance -i Xld_X16
remove_instance -i Xld_X17
remove_instance -i Xld_X18
remove_instance -i Xld_X19
remove_instance -i Xld_X2
remove_instance -i Xld_X3
remove_instance -i Xld_X4
remove_instance -i Xld_X5
remove_instance -i Xld_X6
remove_instance -i Xld_X7
remove_instance -i Xld_X8
remove_instance -i Xld_X9
```

Figura 4.11: Cada una de las conexiones eliminadas conectadas a los PODE del circuito.

Luego de estas modificaciones la simulación no presenta errores, por lo que se termina la verificación con éxito.

Por otra parte, en la sección anterior se mencionó la existencia de la **supply RC net**, la que se encuentra presente en el circuito actualmente estudiado, con lo que se presentan copias de las distintas fuentes de voltaje. Las fuentes originales para este circuito son **VCC**, **VSS** y **VDDI**.

Para lidiar con lo anterior se puede modificar la forma en que se definen las fuentes de voltaje, para que todas estas copias de las fuentes originales sean consideradas como fuentes reales. Con esto se evita que el valor de las fuentes reales tengan que propagarse por toda la *supply RC net*. A continuación, se presenta un esquema en la figura 4.12 para dejar en claro esto:

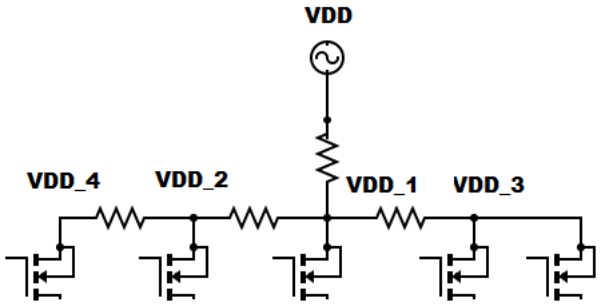


Figura 4.12: Cada uno de los nodos de las resistencias corresponde a una repetición de la fuente original.

Con la utilización del comando *set_supply_net -i -type virtual -logic 1 VDD.** todas esas copias pasarían a ser fuentes reales con la mismas características de la original, como se muestra en el siguiente esquema de la figura 4.13

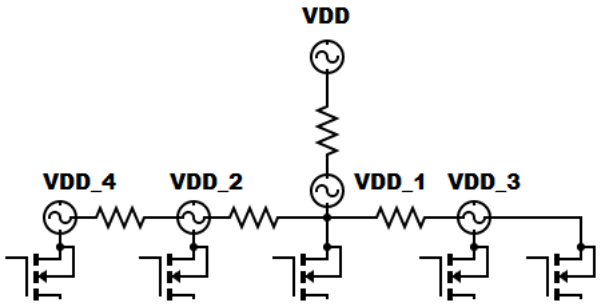


Figura 4.13: Todas las copias pasan a ser fuentes reales.

Que dichas señales no tengan que propagarse por la *supply RC net*, trae consigo una mejora en el tiempo de procesamiento.

En particular, para este circuito, el tiempo que toma la simulación, previo a la nueva definición de las fuentes de voltaje, es de $91[s]$. Mientras que con la modificación en la definición de las fuentes de voltaje, el tiempo que tarda es de solamente $70[s]$. Esta mejora podría parecer bastante pequeña, pero hay que tener en cuenta que se trata solamente de la verificación individual de un circuito, mientras que en general los clientes hacen múltiples verificaciones al mismo tiempo, por lo que esta leve mejora podría significar grandes cambios en los tiempos totales. Esto quedará en evidencia en experimentos que serán explicados posteriormente en este documento.

Cabe destacar que en particular esta lista de conexión pasa la verificación sin problemas. En otros casos, el valor que debe propagar la fuente de voltaje original a través de la *supply RC net* no logra mantener los niveles adecuados para que todos los transistores puedan percibirlos en su conectores, lo que directamente trae problemas de funcionamiento del circuito, dado que el transistor propaga una señal “X”, de un estado indeterminado. Esto provoca discrepancias

al momento de realizar las simulaciones. Por lo que la utilización del comando *set_supply_net -i -type virtual -logic 1 VDD.** se volvería una obligación más que una opción, lo que trae consigo sus propios problemas.

Definir a cada una de las copias de la fuente de voltaje como la original toma tiempo. Nuevamente es necesario mencionar que el caso de estudio anterior es muy particular, tiempo que se multiplica por cada copia de las fuentes de voltaje y por cada circuito que quisiera ser verificado.

Capítulo 5

Adaptación de herramienta de reducción externa para uso con ESP

Existe otro equipo dentro de la organización, llamado StarRC, que lleva años desarrollando, entre otras, su propia herramienta de reducción de listas de conexión y se mostraron dispuestos a prestar su herramienta para que se realizaran pruebas, con la intención de ver como la modificación y eliminación de elementos desde la lista de conexión pueden afectar el rendimiento en el proceso de verificación de ESP.

La cantidad de *netlists* utilizadas previamente no resulta lo suficientemente numerosa como para especificar de manera concreta cuáles son las opciones que podrían permitir el mejor de los desempeños posibles de ESP una vez el circuito es reducido. Por lo que se proveen dos librerías enteras para poder realizar las pruebas necesarias; la primera, desde ahora en adelante Librería A, cuenta con 187 elementos; la segunda, desde ahora en adelante Librería B, cuenta con 243.

Existe toda una política sobre propiedad intelectual, por lo que no se puede mencionar qué cliente es el propietario de los circuitos, tampoco se puede especificar la tecnología utilizada. Las conexiones internas que pueden ser compartidas no incluyen ninguna característica de los elementos que puedan violar las políticas de propiedad intelectual.

5.1. En búsqueda de las mejoras

Al iniciar las pruebas con las dos librerías mencionadas con anterioridad, se cuenta con cuatro comandos de la herramienta de reducción:

- REDUCTION_NETS
- REMOVE_NETS
- REMOVE_DEVICES
- MERGE_DEVICES

Los nombres de los cuatro comandos son bastante intuitivos; el primero, realiza todas las reducciones posibles dentro de la lista de conexión. Se puede especificar el nivel de reducción que se quiere, para ESP, dado que lo importante es la funcionalidad y no la precisión, se utiliza el mayor de estos niveles; el segundo, tiene la capacidad de eliminar redes enteras, en particular se puede eliminar la *supply RC net*; la tercera, elimina todos los elementos que especifiquen dentro de este comando; el cuarto, por su parte, combina los elementos que se repiten y cumplen la misma funcionalidad.

Antes de ejecutar la herramienta es necesario especificar cuáles son todos los comandos que se quieren utilizar en la reducción de la lista de conexión. Un ejemplo que incluye todos los comandos previamente mencionados se ve en la figura 5.1:

```
INPUT_FILENAME: filename.spf
OUTPUT_FILENAME: reduced_filename.spf
REDUCTION_NETS: * LEVEL HIGH MAX_DELAY_ERROR RONLY_KEEPC
REMOVE_NETS: ln_* VDD VSS VDDI VCC vss vssx vdd vcc vddi
REMOVE_DEVICES: *pode*
MERGE_DEVICES: PARALLEL
```

Figura 5.1: Comandos utilizados para la reducción del circuito.

Aquí se está utilizando la mayor de la reducciones de las redes internas, que representaría a la reducción de cada uno de los sub-circuitos como en la figura 3.2. Se eliminan todas las *supply RC net* relacionadas con las fuentes del circuito. Tambien se eliminan todos los elementos tipo PODE. Finalizando con la combinación los elementos repetidos que se encuentren en paralelo, en particular de los transistores.

Para la experimentación con ambas librerías, se utilizaron tres combinaciones distintas de los comandos presentes en la herramienta de reducción. Con el propósito de identificar el impacto que tiene una reducción de los circuitos, junto con la eliminación de los PODE, pero no eliminando las *supply RC net*, se usa la siguiente lista de comandos:

- REDUCTION_NETS
- REMOVE_DEVICES
- MERGE_DEVICES

Luego, para comprobar la mejora eliminando las *supply RC net* y los elementos PODE, se tiene la siguiente combinación:

- REMOVE_NETS
- REMOVE_DEVICES
- MERGE_DEVICES

Para finalizar, se hace el uso de todos los comando como en la lista que sigue.

- REDUCTION_NETS
- REMOVE_NETS
- REMOVE_DEVICES
- MERGE_DEVICES

En la figura 5.1 se puede ver que es necesario especificar el nombre del circuito que se quiere reducir y el nombre que se quiere para el circuito reducido. Dado que se trata de una librería con una gran cantidad de elementos y que la herramienta no cuenta con la capacidad para realizar múltiples reducciones al mismo tiempo. Es necesario crear una pequeña macro, para que este proceso de reducción pueda ser realizado a toda la librería.

Una vez se aplican los tres niveles de reducción a ambas librerías, llega el proceso de verificarlas. Los resultados de esta verificación se comparan con los tiempos obtenidos al realizar el mismo proceso pero sin el proceso de reducción a las redes de conexión.

El resumen de los tiempos de verificación sobre todos los elementos de la Librería A se encuentra en la tabla 5.1

Tabla 5.1: Resumen de los tiempos de verificación para la Librería A, que contiene 189 elementos.

	Lista de conexión original	Reducción interna	Eliminación de red supply RC	Reducción completa
Tiempo de verificación [h:mm:ss]	3:40:13	3:34:48	2:14:35	0:11:33
Porcentaje de tiempo reducido		2.46 %	38.89 %	94.76 %

Similar para los tiempos de verificación de la Librería B en la tabla 5.2

Tabla 5.2: Resumen de los tiempos de verificación de la Librería B, que contiene 274 elementos

	Lista de conexión original	Reducción interna	Eliminación de red supply RC	Reducción completa
Tiempo de verificación [h:mm:ss]	5:30:57	4:38:44	2:16:01	0:02:19
Porcentaje de tiempo reducido		15.78 %	58.90 %	99.30 %

Con lo anterior, se puede notar que la sola eliminación de la *supply RC net* causa un impacto no menor a los tiempos de verificación. Lo anterior, además, elimina la necesidad

definir múltiples fuentes de voltaje, sumado a que ya no se presentarían problemas ligados a la integridad de la señal que va desde la fuente hasta los transistores.

Como era de esperar, la utilización de todos los comandos de reducción sobre las listas de conexión trae consigo la mayor mejora, presentando una disminución tremenda en los tiempos que toma llevar a cabo el proceso de verificación.

Estos resultados fueron presentados tanto al equipo de ESP como al equipo de StarRC. Dado lo impactante de las mejoras presentadas, el equipo de StarRC incorporará dentro de su herramienta una nueva opción para los usuarios.

Todos los clientes que cuenten con la licencia para utilizar ambos productos, ahora podrán hacer uso de la opción *-esp* dentro de la herramienta de reducción producida por el equipo de StarRC, por lo que la reducción de la lista de conexión se realizará utilizando la siguiente lista de comandos:

- REDUCTION_NETS
- REMOVE_NETS
- REMOVE_DEVICES
- MERGE_DEVICES

Dado esto, desde ahora en adelante, la herramienta de ESP tiene incorporada un herramienta externa a su flujo de verificación que permite, sin problemas, trabajar con las listas de conexión en formato *.spf*, además de contar con una mejora importante en los tiempos relacionados con la verificación, gracias a la reducción en la cantidad de conexiones.

Capítulo 6

Propuesta y desarrollo de solución para ESP

En este capítulo se abordará, de la mejor manera posible, la forma en que se presenta la propuesta de solución al resto del equipo, lo que es primordial para que se pueda llevar a cabo el proyecto, sumado a la forma en que esta solución es desarrolla dentro del código de ESP.

6.1. Functional Spect

Una de las partes fundamentales dentro del desarrollo de un proyecto dentro de ESP, corresponde a la propuesta que se hace a los distintos miembros del equipo. La forma de hacer el a través del llamado *Functional Specification* (o *Functional Spect*), que corresponde a un documento que contiene la mayor cantidad de información que se tiene respecto al proyecto, desde cuáles son las motivaciones para realizarlo, pasando por una breve descripción de lo que se quiere realizar, donde se incluyen los cambios que serán vistos o percibidos por el usuario.

La mayor relevancia de este proceso es justamente el impacto que tendrá la implementación en el usuario, por lo que las propuestas y explicaciones de la solución se limitan a lo que se pretende hacer, el "cómo" se hace se presenta más adelante en este informe.

Para comenzar a trabajar en el proyecto, es necesario que el *Functional Spect* sea aprobado por los miembros del equipo más cercano a los clientes, los que dentro de la organización son denominados *Produtc Application Engineer* (PAE). A pesar de lo anterior, el documento puede ser revisado por cualquier miembro del equipo, quienes pueden hacer cualquier pregunta o comentario que les parezca prudente.

Con todos los resultados obtenidos del trabajo realizado con el equipo de StarRC, los cuales muestran una mejora bastante buena en los tiempos de verificación, la propuesta de desarrollo de un proceso de reducción parasitaria propio de ESP surge casi de manera natural.

Por lo que se muestra en las tablas 5.1 y 5.2, la mejora que genera por si sola la eliminación de la *supply RC net* en los tiempos de verificación es significativa, además, esto demuestra que

la eliminación de estas conexiones soluciona los problemas que se presentaban antes durante el proceso de verificación. Esto trae consecuencias no menores, ya que antes de realizar la colaboración con el equipo de StarRC, la idea general que se tenía era de realizar un proceso de reducción de elementos dentro de ESP, no de eliminación de elementos. Es por lo anterior que dentro de la investigación bibliográfica se incluyen los métodos de reducción de orden. Pero estos resultados cambian el enfoque a la eliminación de elementos, pero no de todas las redes, solamente de aquellas que pertenecen a la *supply RC net*.

Esta eliminación se puede realizar dentro del flujo de verificación del circuito por lo que, a diferencia de lo que hace la herramienta de StarRC, no se pretende modificar directamente la lista de conexión entregada por el cliente; esto mediante el cambio en la forma en que se crea el archivo *.spf*.

Al realizar una verificación de una netlist en formato *.spf*, el usuario debe ser consciente si esta contiene o no transistores tipo PODE. De ser así, se debe utilizar el comando *set_net_device* para que ESP lo reconozca como un transistor común y corriente. Esto evita que falle el comando *read_spice* por no reconocer el comportamiento de dicho elemento.

6.1.1. Presentación del *Functional Spect*

Quienes asisten a esta presentación son todos los PAE, a quienes se les explica la problemática que existe y se recorre sección por sección el documento. Los miembros del equipo de desarrollo pueden asistir a esta reunión si les parece apropiado.

En general, se hace necesaria más de una presentación para poder aprobar el *Functional Spect*, en vista de que cada uno de los asistentes puede sugerir cambios en el documento, lo que puede influir directamente en el desarrollo de la solución. Por ese motivo resulta prudente esperar la aprobación antes de iniciarlo.

La importancia de estas reuniones recaen en los distintos puntos de vista que los miembros del equipo poseen, debido a su conocimiento de la herramienta a nivel usuario, por lo que para un desarrollador puede parecer prudente llevar a cabo los cambios de cierta manera, pero se podría afectar de manera negativa la experiencia a nivel usuario.

Un ejemplo de lo anterior, ocurrió durante la primera de las presentaciones al equipo. Dentro de la solución, el usuario debía informar que realizaría una verificación con una lista de conexión en formato *.spf* agregando una opción dentro del comando *read_spice*. El archivo *.tcl* quedaría como se muestra en la figura 6.1:

```
set sh_continue_on_error true
# Set style which generates single testbenches
set testbench_style symbolic

# Prepare environment to check design
read_verilog -r test.v
read_spice -spf -i inv.sp
set_supply_net -i -type real -logic 1 VDD
set_supply_net -i -type real -logic 0 VSS

verify

quit
```

Figura 6.1: Proceso de verificación de un inversor con modificación al comando *read_spice* .

Dentro de los comentarios recibidos por parte del resto del equipo, es que cuando a un usuario se le da la posibilidad de agregar una nueva opción a un comando, ellos esperan nuevas funcionalidades para dicho comando, lo cual en este caso no se cumple, ya que solamente se está agrandando el rango de tipos de listas de conexiones que puede procesar ESP.

No solamente se debe incluir la nueva forma en que se debe realizar la verificación de los circuitos, también es importante agregar los nuevos mensajes de información, advertencia o errores que serán desplegados en pantalla de ser necesarios.

Luego de cada reunión se deben aplicar todos los cambios sugeridos por el equipo y se debe realizar una nueva presentación. Lo anterior se repite tantas veces sea necesario hasta que el documento sea aprobados por todos los miembros del equipo incluidos en este proceso.

En particular, para este proyecto, se realizaron solamente dos presentaciones antes de ser aprobado. Una vez aprobado inicia el desarrollo del proyecto, que para asuntos internos es llamado *SPF Removal*.

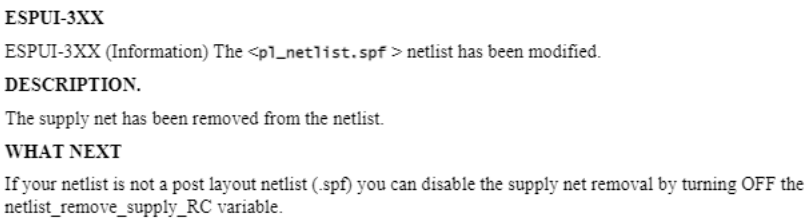
6.2. Descripción de la solución vista en el *funtional spect*

Desde ahora en adelante, la búsqueda y posterior eliminación de la *supply RC net* se realizará siempre durante el proceso de verificación. Lo anterior se logra mediante una variable de ambiente que permanecerá activada hasta que el usuario diga lo contrario. Esta variable tiene por nombre *netlist_remove_supply_RC*.

Si el usuario está consciente de que la lista de conexiones cuenta con elementos de tipo PODE, este debe utilizar el comando *set_device_model* previo a que se utilice el comando *read_spice*, debido a los problemas que traen consigo estos elementos, los cuales fueron comentados en capítulos anteriores.

Si bien puede que la red de conexiones no tenga una *supply RC net*, es necesario informar al usuario que se realizarán ciertas modificaciones durante el proceso de verificación, por lo que es necesario crear un nuevo mensaje de información, donde se le explica de manera breve

lo que ocurre y qué es lo que puede hacer para que estos cambios no se apliquen. Esto se puede ver en la siguiente figura 6.2:



ESPUI-3XX

ESPUI-3XX (Information) The <pl_netlist.spf > netlist has been modified.

DESCRIPTION.

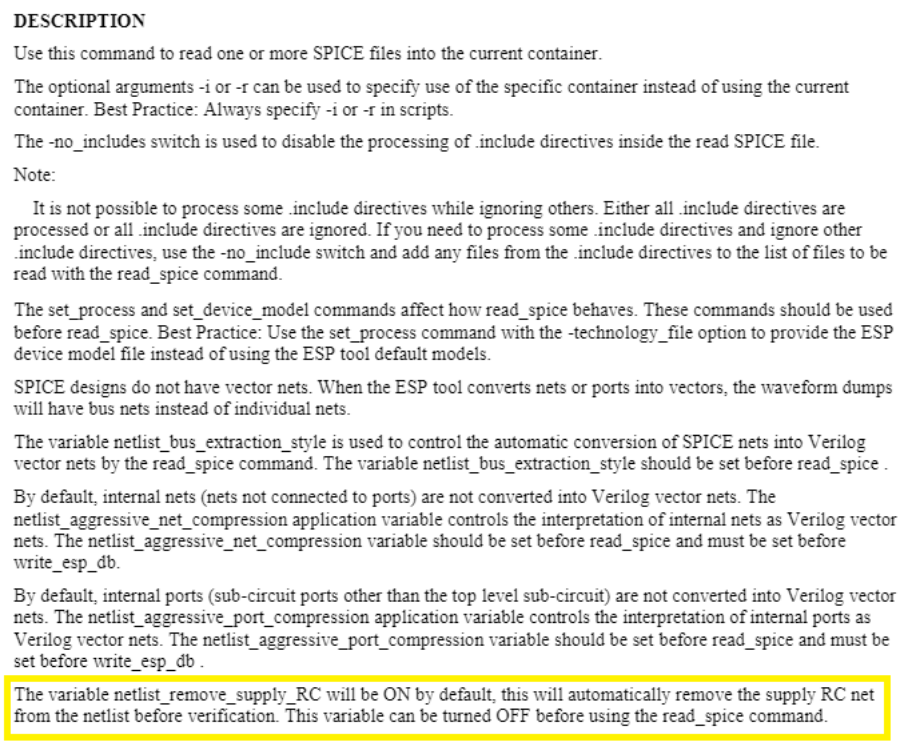
The supply net has been removed from the netlist.

WHAT NEXT

If your netlist is not a post layout netlist (.spf) you can disable the supply net removal by turning OFF the netlist_remove_supply_RC variable.

Figura 6.2: Nuevo mensaje de información, su explicación y sus indicaciones.

Todos los comandos a los que tienen acceso los usuarios, cuentan con su propia explicación de uso, lo cual se incluye dentro del manual de usuario de ESP. Si bien la forma en que se utiliza el comando *read_spice* no cambia, sí cambia su descripción, la cual se puede leer a continuación en la figura 6.3:



DESCRIPTION

Use this command to read one or more SPICE files into the current container.

The optional arguments -i or -r can be used to specify use of the specific container instead of using the current container. Best Practice: Always specify -i or -r in scripts.

The -no_includes switch is used to disable the processing of .include directives inside the read SPICE file.

Note:

It is not possible to process some .include directives while ignoring others. Either all .include directives are processed or all .include directives are ignored. If you need to process some .include directives and ignore other .include directives, use the -no_include switch and add any files from the .include directives to the list of files to be read with the read_spice command.

The set_process and set_device_model commands affect how read_spice behaves. These commands should be used before read_spice. Best Practice: Use the set_process command with the -technology_file option to provide the ESP device model file instead of using the ESP tool default models.

SPICE designs do not have vector nets. When the ESP tool converts nets or ports into vectors, the waveform dumps will have bus nets instead of individual nets.

The variable netlist_bus_extraction_style is used to control the automatic conversion of SPICE nets into Verilog vector nets by the read_spice command. The variable netlist_bus_extraction_style should be set before read_spice .

By default, internal nets (nets not connected to ports) are not converted into Verilog vector nets. The netlist_aggressive_net_compression application variable controls the interpretation of internal nets as Verilog vector nets. The netlist_aggressive_net_compression variable should be set before read_spice and must be set before write_esp_db.

By default, internal ports (sub-circuit ports other than the top level sub-circuit) are not converted into Verilog vector nets. The netlist_aggressive_port_compression application variable controls the interpretation of internal ports as Verilog vector nets. The netlist_aggressive_port_compression variable should be set before read_spice and must be set before write_esp_db .

The variable netlist_remove_supply_RC will be ON by default, this will automatically remove the supply RC net from the netlist before verification. This variable can be turned OFF before using the read_spice command.

Figura 6.3: Descripción del comando en el manual, incluye modificaciones.

El nuevo párrafo incorporado a esta descripción corresponde al encerrado en el recuadro amarillo de la figura 6.3.

Si bien los elementos de tipo PODE siguen presentado un problema para la verificación utilizando archivos *.spf*, existe una forma de eliminarlos, como se vio en el capitulo 5 (ver figura 4.11). Por lo que el enfoque de este proyecto es la eliminación de la *supply RC net*.

6.3. Desarrollo de Solución Interna

Como fue mencionado previamente, durante el proceso de verificación se crea un archivo en formato *.gv*. Dado que las *netlists* que fueron reducidas, utilizando la herramienta de reducción provista por el equipo de StarRC, fueron verificadas sin problemas. Desde este archivo se puede obtener información muy importante para guiar el trabajo de desarrollar una solución propia de ESP.

El trabajo inicial consta de la comparación entre dos archivos *.gv* que provienen del mismo circuito, ¿como es posible que, verificando el mismo circuito, se obtengan dos *.gv* distintos? Si bien se está estudiando el comportamiento del mismo circuito, la representación de este puede variar; el primero, se obtiene del proceso de verificación utilizando la lista de conexiones sin modificar; el segundo, es el obtenido de este mismo proceso pero reduciendo la lista de conexiones previamente con la herramienta de StarRC.

Ambos archivos *.gv* derivan en una verificación correcta. Es importante recordar que no todas las *.spf netlist* fallan durante este proceso, pero sí presentan ciertas dificultades y diferencia en el tiempo que esto toma, lo que se puede ver claramente en las tablas 5.1 y 5.2.

En resumen, no se busca realizar modificaciones directamente sobre la lista de conexiones, ya que dentro del proceso de ESP esta se mantendrá sin cambio alguno. Más bien, se quiere modificar el proceso tras la generación del archivo *.gv*, que es, finalmente, sobre el cual se realizan las simulaciones necesarias para comprobar si su funcionalidad es la correcta.

6.3.1. Elementos a eliminar

Como se mencionó previamente, los elementos que deben ser eliminados corresponden a aquellos que forman parte de la *supply RC net*, es decir, todos los elementos resistivos y capacitivos que van desde un nodo definido como fuente de voltaje hacia los transistores. Los elementos no "saben" si pertenecen o no a dicha red. Por lo que lo primero que se debe implementar, es una forma de determinar si forman o no parte de ella.

Para lo anterior, primero se debe tener en cuenta cuál es la información que se tiene a disposición. Esto está directamente relacionado con la forma en que ESP maneja la información obtenida desde la lista de conexión que fue ingresada mediante el comando *read_spice*.

Dentro de las muchas cosas que se pueden saber de cada elemento, al momento de ser procesado, algunas destacan por su importancia para el desarrollo de la solución propia para la eliminación de la red de conexiones, las cuales corresponden a las siguientes:

- Nombre del elemento.
- Tipo de elemento.
- Conexiones de los terminales de los elementos.

El código base de ESP está desarrollado en los lenguajes de programación **C** y **C++**, y cada uno de los elementos circuitales especificados en las listas de conexiones es almacenado

como un **objeto** dentro de alguna **clase** determinada. La forma en que dichas clases y objetos están definidas no pueden ser detalladas por razones de derecho de autor, pero es importante mencionarlo dado que esta característica permite realizar modificaciones mucho más intuitivas para la búsqueda de los elementos que conforman la *supply RC net*.

En este caso se trata de de una propiedad de nombre *_removeSPF* representado por un valor booleano, es decir puede ser **verdadera** o **falsa**. El valor inicial de esta propiedad para todos los elementos del circuito, previo a realizar la determinación de la red, es **falso**.

El primer intento por identificar todos los elementos relacionados con la *supply RC net* es bastante simple: se recorre la lista de elementos presente, se identifica si corresponde o no a un elemento resistivo o capacitivo, luego se verifica si este elemento está conectado o no a una fuente de voltaje.

Tanto los elementos capacitivos como resistivos cuentan solamente con dos terminales o nodos. Por simplicidad, para explicar lo que se quiere realizar, se puede definir una resistencia como $R_{nombre}(Nodo\ 1, Nodo\ 2)$; lo mismo para una capacitancia $C_{nombre}(Nodo\ 1, Nodo\ 2)$.

Entonces, al recorrer la lista de conexiones y encontrar un elemento capacitivo o resistivo, se accede al primer nodo y se verifica si ese nodo está conectado a una fuente de voltaje, si lo está, el valor de *_removeSPF* cambia a **verdadero**.

Con el estudio de las lista de conexión en formato *.spf* se pudo notar que, en particular para los elementos que conforman la red que se busca eliminar, las resistencias y capacitancias tienen una diferencia importante; las primeras, por lo general, son más en número conectándose en serie hasta alcanzar el transistor; las segundas se encuentran siempre conectadas en paralelo a alguna resistencia, por lo que basta con identificar este elemento único.

Por lo anterior, si se encuentra con una resistencia conectada a una fuente de voltaje, es necesario revisar el elemento conectado a su segundo nodo. Si este corresponde a una resistencia este igual puede ser "marcado" como parte de los elementos a eliminar. Lo anterior se puede llevar a cabo hasta que el elemento conectado al segundo nodo sea un transistor, lo que marcaría el final de la *supply RC net*, luego se sigue recorriendo la lista de conexiones. El proceso descrito anteriormente se puede ver en la figura 6.4.

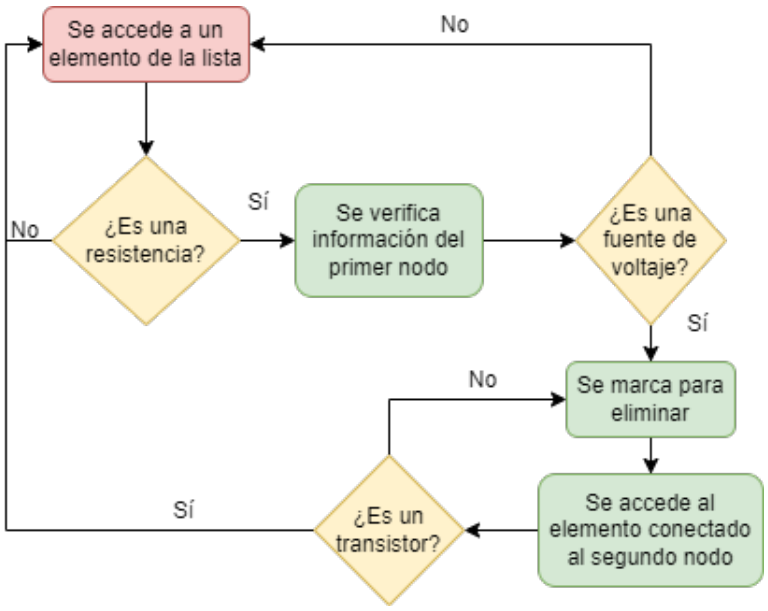


Figura 6.4: Diagrama de proceso de determinación de elementos resistivos pertenecientes a la *supply RC net*.

Para las capacitancias es mucho más simple: si esta tiene una conexión a una fuente de voltaje, simplemente se identifica para ser eliminado y se continúa la revisión de los elementos restantes de la lista de conexión. Similar a las resistencias, se agrega un diagrama para mejor entendimiento (figura 6.5).

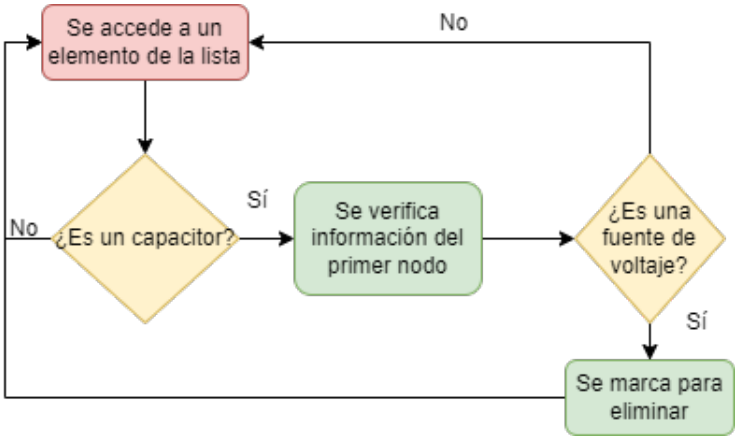


Figura 6.5: Diagrama de proceso de determinación de elementos capacitivos pertenecientes a las *supply RC net*.

6.3.2. Escritura de la *.gv file*

Una vez que se tiene identificada la *supply RC net*, es necesario modificar el proceso de escritura de la *.gv file* para que esta no se vea reflejada al momento de realizar la simulación. Por la forma en que se encuentra implementada la escritura de este archivo dentro de ESP, la eliminación de esta red resulta bastante simple.

Gracias a que se agregaron y se modificaron las carácteristicas de cada objeto que representa a los elementos circuitales, simplemente se busca a cada uno de los objetos que fueron marcados como miembros de la *supply RC net*. Entonces, todo objeto que cumpla con que la propiedad *_removeSPF* es **verdadera**, simplemente no será incorporado en la *.gv file*.

6.3.3. Problemáticas encontradas

En un inicio todo el proceso descrito anteriormente parece bastante simple de llevar a cabo, por lo que se tendría una solución al problema del procesamiento de las *.spf netlist* por parte de ESP, pero nada en la vida es realmente tan simple y la cantidad de problemáticas que fueron manifestándose durante el desarrollo no es menor.

El orden en que se describen a continuación los inconvenientes y sus soluciones, es el mismo orden en el que se presentaron durante el desarrollo de la solución interna.

6.3.3.1. Fuentes de voltajes multiplicadas

Como se pudo ver previamente con el análisis del formato *.spf*, es posible encontrar las fuentes de voltaje multiplicadas, siendo estas una representación de la fuente original. Por ejemplo, si la fuente original es **VDD** se podría encontrar con **VDD_1**, **VDD_2**, lo que se puede extender cuantas veces sea necesario.

El problema con estas fuentes de voltaje multiplicadas es que, para ESP, no están definidas como tales, es decir, si una resistencia $R_{cualquiera}(Nodo\ 1, Nodo\ 2)$ en su primer nodo tiene conectada una de estas fuentes, al momento de verificar si esta resistencia corresponde o no a un elemento de la *supply RC net*, esta no será considerada como una, pues **VDD_N** no es una fuente de voltaje.

Lo anterior ocurre debido a que el comando *set_supply_net* solamente asigna como fuente de voltaje a la original, en este caso **VDD**. Debido al largo tiempo de desarrollo de ESP, problemas similares fueron afrontados con anterioridad, por lo que existe la posibilidad para el usuario de utilizar una especie de "comodín" en este comando para que todas las fuentes que inicien con ese nombre sean consideradas como fuentes de voltaje. Para lo anterior, el usuario debería agregar un asterisco al final del nombre de la fuente, por ejemplo *set_supply_net -i -type real -logic 1 VDD**.

Esto último resulta bastante útil dependiendo de la cantidad de fuentes que se quieran definir. En el caso particular de las redes definidas en formato *.spf*, dadas las veces que se necesita realizar esto, no resulta para nada conveniente en términos de tiempo. Cabe destacar, que si bien se puede realizar el proceso de verificación para un circuito en particular, en ocasiones los clientes quieres verificar librerías completas de circuitos, como ambas librerías que fueron utilizadas durante la colaboración con StarRC, por lo que un aumento de tiempo considerable en solamente un circuito, que puede llegar a tener más de cien replicas de la misma fuente, de entre cientos de circuitos resulta, por lo bajo, poco práctico.

Para dar solución a este problema, previo a que se realice la identificación de la *supply RC net*, se identifican dentro de la lista de conexiones cuales son los las fuentes de voltaje que fueron definidas por el usuario, luego se recorre la lista buscando cuáles son sus correspondientes

"copias", las cuáles son reemplazadas por la fuente de voltaje original.

La búsqueda es bastante simple: se almacenan en un vector todos los nombres de las fuentes originales, luego los nombres de los elementos son comparados con los almacenados en dicho vector. Si llegan a coincidir, se elimina y en su lugar se conecta la fuente de voltaje original. Lo anterior se puede ver en la figura 6.6.

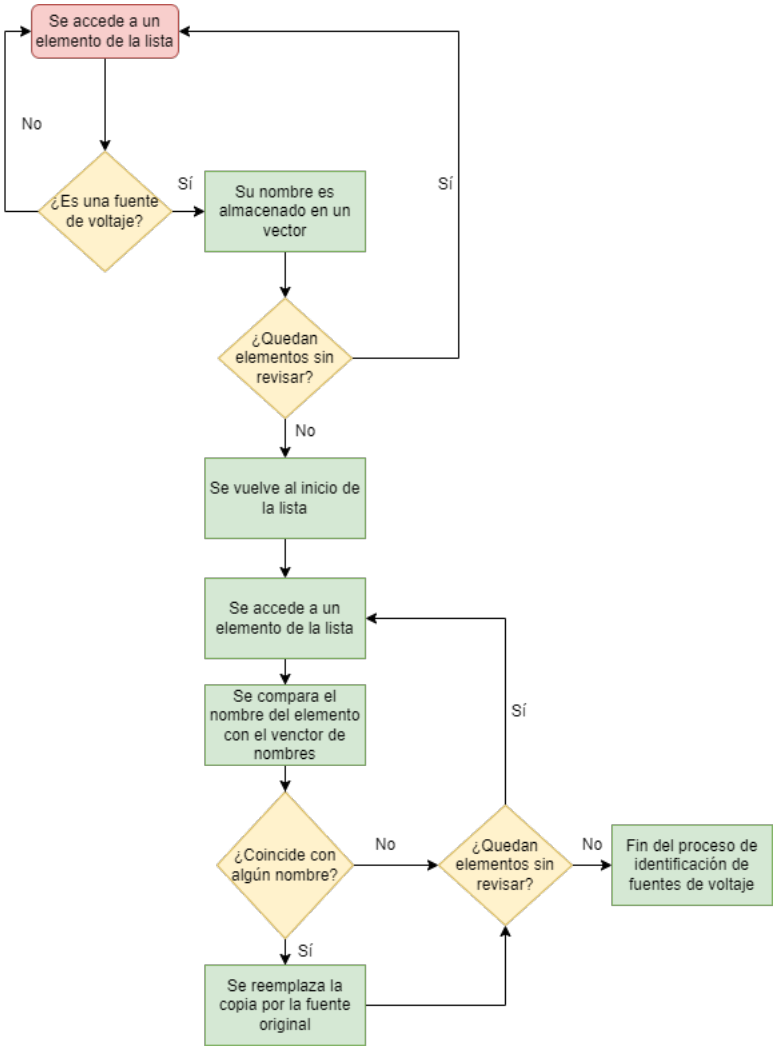


Figura 6.6: Búsqueda y reemplazo de fuentes de voltaje copiadas.

6.3.3.2. Propagación de las fuentes de voltaje

A pesar de que los elementos están siendo eliminados de manera correcta, al momento de realizar la verificación del circuito esta falla. Esto ocurre debido a que al eliminar el "camino" que sigue el voltaje hasta el transistor, dichos voltajes no se ven reflejados en los nodos correspondientes de los diversos transistores.

Entonces es necesario que las fuentes de voltaje sean correctamente propagadas hasta llegar al transistor antes de que todos los elementos de la red sean eliminados. Para esto, si se tiene una resistencia conectada a cualquiera de las fuentes de voltaje identificadas a su primer nodo $R_1(Nodo\ 1, Nodo\ 2)$, que a su vez tiene conectado a su segundo nodo otra resistencia

$R_2(Nodo\ 1, Nodo\ 2)$, todas las propiedades del primer nodo de R_1 serán traspasadas al segundo nodo de esta misma resistencia, entonces ahora el primer nodo de R_2 corresponde a una fuente de voltaje, pues $Nodo\ 2_{R1} = Nodo\ 1_{R2}$. De manera gráfica si se tiene una sucesión de tres resistencias conectadas a un transistor, pasaría lo siguiente (figura 6.7):

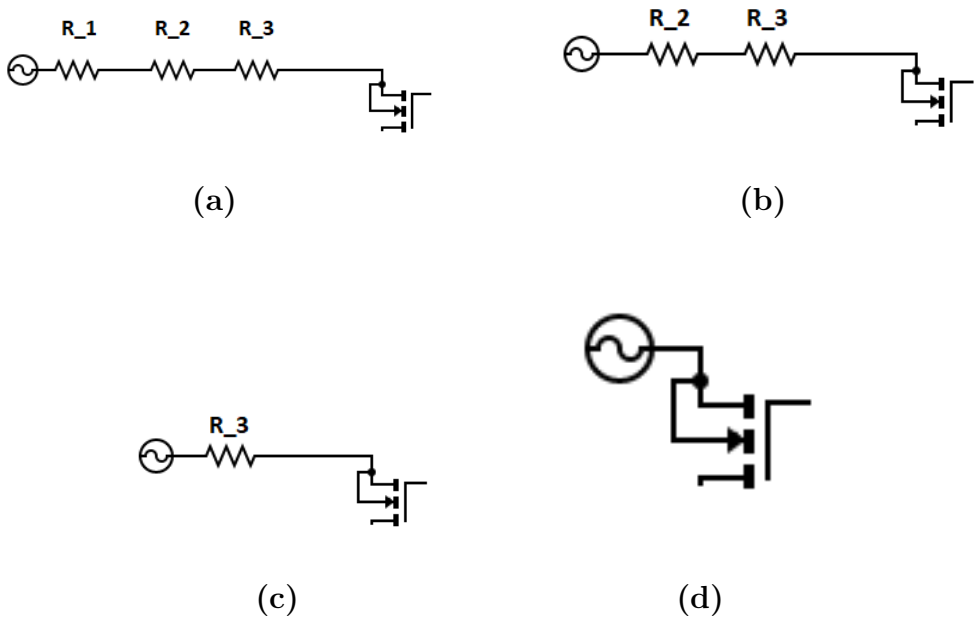


Figura 6.7: **(a)** Antes de propagar **(b)** Se propaga por la primera resistencia **(c)** Se propaga por la segunda resistencia **(d)** La fuente queda conectada directamente al transistor

6.3.3.3. Eliminación errónea de elementos

Otro problema que se presentó es la incorrecta eliminación de elementos, particularmente de resistencias cuyo segundo terminal se encontraba conectado a algún nodo de un transistor, lo que ocurría debido a una mala identificación de los límites de la *supply RC net*.

El final de la red a eliminar está determinado por la primera conexión que tiene una resistencia con un transistor. La idea inicial para identificar esto era bastante simple, cada vez que se accede a una resistencia que es parte de la red en cuestión, se accede a la información almacenada en su segundo nodo y se verifica que el tipo de elemento al que se está conectado sea o no un transistor. Lamentablemente, la información que se puede obtener desde los nodos de la resistencia no permite determinar de manera simple si es o no un transistor, esto debido a como se encuentran implementadas las distintas clases y objetos dentro del código.

Para solucionar lo anterior, simplemente se aprovechan las características de los transistores, el que cuenta con cuatro terminales *Bulk*, *Drain*, *Source* y *Gate*. En general el terminal *Bulk* no es muy utilizado, pero en un futuro los distintos fabricantes podrían darle un mayor uso, ya que el voltaje aplicado a este terminal puede modificar los niveles de voltajes necesarios para la conducción del transistor. Por lo anterior, se fija el interés solamente en *Drain*, *Source* y *Gate*.

Si es que la resistencia cuenta con una conexión en su primer nodo a una fuente de voltaje, se accede la información en su segundo nodo y se verifica que el nombre del elemento conectado a este concuerda con el *Drain*, *Source* o *Gate*. Si es que llega a coincidir el proceso de propagación de la fuente de voltaje, termina ahí para esa rama y se continúa marcando otras *supply RC net* si es que existen.

Incorporando esta modificación y la descrita en el punto anterior, el diagrama que describe la identificación de las resistencias pertenecientes a la *supply RC net*, incluyendo su propagación, queda como se muestra en la figura 6.8.

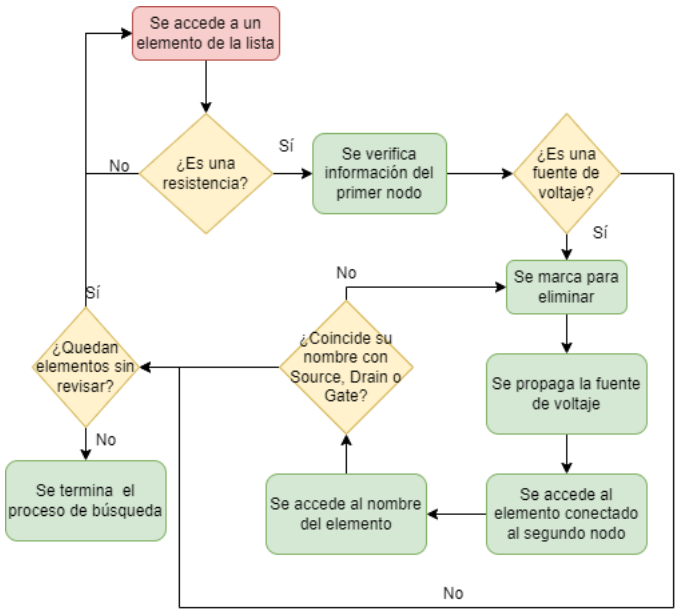


Figura 6.8: Diagrama de determinación de resistencias pertenecientes a la *supply RC net*.

6.3.4. Diagrama final del algoritmo para la determinación de la *supply RC net*

Si se juntan todas las etapas descritas, incluyendo las modificaciones necesarias dadas las problemáticas encontradas durante el proceso de desarrollo el diagrama final que describe la determinación de la *supply RC net*, el que se puede ver en la figura 6.9.

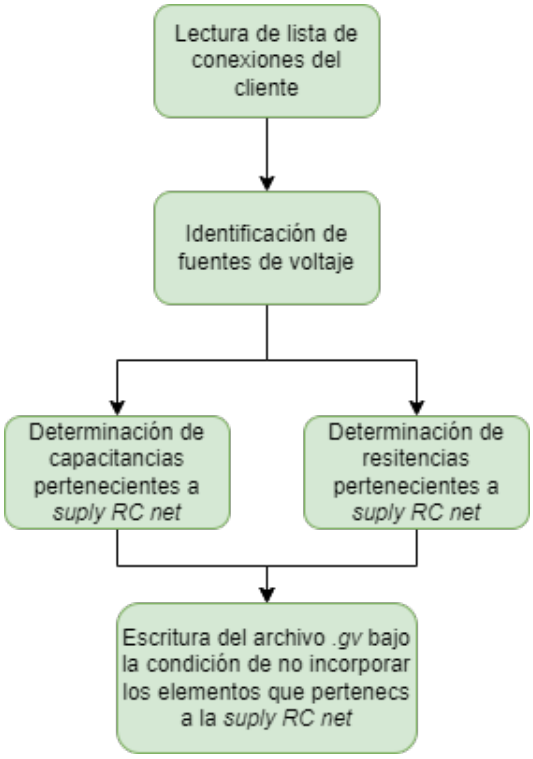


Figura 6.9: Diagrama incorporando cada uno de los procesos involucrados en el desarrollo de la solución.

Capítulo 7

Conclusión y Trabajos a Futuro

7.1. Conclusiones

En relación al objetivo planteado, gracias a todo el trabajo realizado en conjunto entre los equipos de ESP y StarRC, los clientes que cuenten con las licencias de ambos productos cuentan con una nueva funcionalidad en la herramienta de reducción, la cual permite, solamente utilizando el comando *reduction -esp netlist_name.spf*, modificar las listas de conexiones *.spf* de tal manera que de que el proceso de verificación mediante la utilización de ESP pueda realizarse sin problema alguno.

Esta nueva funcionalidad realiza diversos cambios dentro de la lista de conexiones, los cuales favorecen a la rápida propagación de señales durante las simulaciones, permitiendo la comprobación de la funcionalidad del circuito. Los cambios que se realizan a la lista de conexiones: eliminación completa de la *supply RC net*, lo que beneficia a la propagación de las fuentes de voltaje a los elementos del circuito; reducción de red RC interna, esto disminuye la cantidad de resistencias y capacitancias presentes entre las conexiones internas del circuito; eliminación de componentes PODE, dado que una de las problemáticas estaba ligada a que estos elementos no eran relevantes para la funcionalidad del circuito su eliminación permite que no sea necesario utilizar el comando *set_net_device* previo a leer los circuitos; la unión de elementos repetidos, en las *.spf netlists* es común encontrar la representación del mismo transistor varias veces esto aumenta las conexiones internas lo que afecta directamente los tiempos de simulación.

Si bien todos estos cambios son relevantes para que el proceso de verificación se pueda llevar a cabo sin problema, los dos más importantes son la reducción de red RC interna y la eliminación de la *supply RC net*, ya que la primera reduce los tiempos hasta en un 15 % y la segunda hasta en un 58 %. Pero el mayor beneficio se obtiene de que todas estas modificaciones se realicen en conjunto, llegando a un 99 % en la reducción de los tiempos necesarios para realizar el proceso de verificación en ESP.

Por lo que ahora es posible, mediante el uso de una herramienta externa a ESP, realizar la verificación sin problemas de *post layout netlist*, las cuales corresponden al formato de extensión *.spf*. Dado lo anterior es posible decir que el objetivo general se da por cumplido.

Es importante mencionar que el trabajo realizado en conjunto con el equipo de StarRC

tomó un total de tres meses, esto dada la gran cantidad de simulaciones distintas que fueron necesarias para obtener los resultados óptimos dentro de las verificaciones.

Dado que es posible realizar el proceso de verificación sobre listas de conexión con formato *.spf* sin problemas, a pesar de que sea con una herramienta externa, el proyecto a ojos del equipo se encuentra completado con éxito.

Por otra parte se tiene el desarrollo e implementación de la solución interna de ESP a la verificación de *post layout netlists*. Con la ayuda de los resultados obtenidos del trabajo con el la herramienta de StarRC, se puede notar que la eliminación de las *supply RC netlist* es suficiente para que el proceso se pueda llevar a cabo sin mayor inconveniente, por lo que se modifica el proceso de generación del archivo *.gv* en búsqueda de borrar todos los elementos que forman parte de dicha lista de conexión.

Todo el flujo que sigue un archivo desde que es "leído" por ESP hasta que se genera el archivo en formato *.gv* es un mundo aparte dentro del código base de la herramienta, conocido por el equipo como *Spice Parser*. Aquí se reconocen y se caracterizan cada uno de los elementos que están presentes en la *netlist* ingresada.

El principal problema es buscar la forma en que se pueda propagar una de las principales características para el funcionamiento correcto del circuito: las fuentes de voltaje. Con lo anterior y con la capacidad de poder recorrer la *supply RC net*, la eliminación de dichos elementos se vuelve posible de tal manera en que la funcionalidad del circuito no se ve comprometida.

Las pruebas iniciales realizadas sobre circuitos básicos, como lo es un inversor, comprueban que la solución es funcional y el proceso de verificación es posible realizarlo sin problema. Las listas de conexión provistas por el cliente son mucho más complejas en términos de cantidad de elementos y conexiones, lo que trae consigo una dificultad agregada para la revisión de los elementos presentes en el archivo *.gv*. Por lo anterior no es posible que se realice un proceso de validación para la solución propuesta.

El aprendizaje durante el desarrollo de este proyecto es impresionante, no solamente relacionado con el uso y el funcionamiento de ESP, sino también a niveles de programación en C/C++ y mejora en la comunicación oral y escrita en inglés.

7.2. Trabajos a futuro

A pesar de todo el trabajo realizado no fue posible que la solución fuera incorporada al código base. Un continuo estudio y aún más trabajo en la familiarización con el funcionamiento de ESP traerá consigo la posibilidad de mejorar las falencias que presenta hasta el momento el algoritmo encargado de la eliminación de la *supply RC net*.

Se seguirá mejorando el código actual con la meta de que, en un siguiente ciclo de lanzamiento del producto, se pueda aprobar con éxito la validación del proyecto y finalmente incorporarlo como una mejora funcional.

En estos momentos el usuario cuenta con una opción para poder eliminar los elementos PODE de sus listas, pero es necesario que este identifique de forma previa dichos elementos. Se pretende buscar una forma de que este proceso se pueda realizar de manera interna en ESP en búsqueda de mejorar aún más la experiencia como usuario de la herramienta.

Dadas las mejoras en el tiempo de verificación que aporta la combinación de eliminar la *supply RC net* con la reducción de elementos en las redes internas que fueron posibles gracias a la herramienta provista por el equipo de StarRC, se planea estudiar la factibilidad de implementar un método de reducción propio de ESP y que se realice de manera interna durante el proceso de verificación.

Es importante mencionar que el trabajo dentro de la empresa continúa, por lo que lo mencionado previamente puede se puede lograr para futuras actualizaciones del producto.

Bibliografía

[1] Marc T. Thomson, "Intuitive analog circuit design", Newnes 2006.

[2] International Roadmap for Devices and Systems, "International doadmap for devices and dystems 2021 Edition" <https://irds.ieee.org/editions>. 2021.

[3] Semiconductor Industry Association, "State of the U.S. semiconductor industry", <https://www.semiconductors.org/state-of-the-u-s-semiconductor-industry/>. 2021.

[4] Clayton B. McDonald, "Symbolic functional and timing verification of transistor level circuits", <https://www.researchgate.net/publication/221626969_Symbolic_functional_and_timing_verification_of_transistor-level_circuits>. 2001

[5] Paul Hoxey, Clayton B. McDonald, David Guinther, "An introduction to symbolic simulation", <https://www.design-reuse.com/articles/12192/an-introduction-to-symbolic-simulation.html>. 2005.

[6] Peter Benner, Michael Hinze, E. Jan W. ter Maten, "Model reduction for circuit simulation", Springer, 2011.

[7] Sheldon Tan, Lei Hen, "Advanced model order reduction techniques in VLSI design", Cambridge University Press, 2007,

[8] Wilhelmus H.A Schilders, Henk A. van der Vorst, Joost Rommes, "Model order reduction: theory, research aspects and applications" Springer, 2008

[9] "MOR Wiki", <https://morwiki.mpi-magdeburg.mpg.de/morwiki/index.php/Main_Page>

[10] "European Network for Model Reduction", <http://www.eu-mor.net/>

[11] Gordon Moore, "The future of integrated electronics", Electronics Magazine, 1965.

[12] International Roadmap for Devices and Systems, "More Moore", <https://irds.ieee.org/editions/2021/more-moore>, 2021.

[13] John Shalf, "The future of computing beyond Moore's Law", <https://royalsocietypublishing.org/doi/10.1098/rsta.2019.0061>, 2020.

[14] Wolfgang Arden, Michel Brillouët, Patrick Cogez, Mart Graef, Bert Huizing, Reinhard Mahnkopf, "More than Moore", <http://www.itrs2.net/uploads/4/9/7/7/49775221/irc-itrs-mtm-v2_3.pdf>

[15] Synopsys, "What is Sysmoore", <https://www.synopsys.com/glossary/what-is-sysmoore.html>, 2021.

[16] C. W. Ho, A. E. Ruehli, P. A. Brennan, "The modified nodal approach to network analysis" IEEE Transactions on Circuits and Systems, vol. 22, no. 6, pp. 504-509, 1975.

[17] L. T. Pillage, R. Rohrer, “Asymptotic waveform evaluation for timing analysis”, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 9, no. 4, pp. 352-366, 1990.

[18] L. Pillage, “Electronic Circuit & System Simulation Methods (SRE).” McGraw-Hill, Inc, 1998.

[19] E. J. Grimme, “Krylov projection method for model reduction”, Ph.D. dissertation, University of Illinois, 1997.

[20] A. Odabasioglu, M. Celik, L. T. Pileggi, “PRIMA: passive reduced-order interconnect macromodeling algorithm”, Proceedings of the 1997 IEEE/ACM international conference on Computer-aided design. IEEE Computer Society, pp. 58-65, 1997.

[21] W. E. Arnoldi, “The principle of minimized iterations in the solution of the matrix eigvalue problem”, Quarterly of Applied Mathematics, vol. 9, no. 1, 17–29, 1951.

[22] B. C. Moore, “Principal component analysis in linear systems: Controllability, observability, and model reduction”, IEEE Transactions on Automatic Control, vol. 26, no. 1, pp. 17-32, 1981.

[23] J. R. Phillips, L. M. Silveira, “Poor man’s TBR: a simple model reduction scheme”, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 24, no. 1, pp. 43-55, 2005.