Instituto Tecnológico de Costa Rica

Escuela de Ingeniería Electrónica

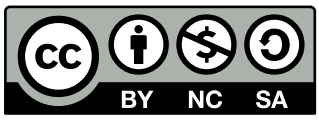
TEC | Tecnológico de Costa Rica

Diseño de un generador de bibliotecas de celdas estándar para flujo de síntesis física

Documento de tesis sometido a consideración para optar por el grado académico de Maestría en Electrónica con Énfasis en Microelectrónica

Luis Felipe Retana Corrales

Cartago, 16 de Febrero, 2023



Instituto Tecnológico de Costa Rica
Escuela de Ingeniería Electrónica
Maestría Académica en Electrónica
Trabajo Final de Graduación
Tribunal Evaluador
Acta de Aprobación de Tesis

Defensa del Trabajo Final de Graduación
Requisito para optar por el título de Máster en Ingeniería Electrónica
Grado Académico de Magister Scientiae

El Tribunal Evaluador aprueba la defensa del Trabajo Final de Graduación denominado **"Diseño de un generador de bibliotecas de celdas estándar para flujo de síntesis física"**, realizado por **Luis Felipe Retana Corrales** Carné: **201111782**, y hace constar que cumple con las normas establecidas por la Unidad Interna de Posgrados de la Escuela de Ingeniería Electrónica del Instituto Tecnológico de Costa Rica.

Miembros del Tribunal Evaluador

ROBERTO CARLOS MOLINA ROBLES (FIRMA)
Digitally signed by ROBERTO CARLOS MOLINA ROBLES (FIRMA)
Date: 2023.02.09 00:05:19 -06'00'

M.Sc Roberto Molina Robles
Profesor Lector

JUAN JOSE MONTERO RODRIGUEZ (FIRMA)
Digitally signed by JUAN JOSE MONTERO RODRIGUEZ (FIRMA)
Date: 2023.02.09 13:14:37 -06'00'

Dr. Juan José Rodríguez
Profesor Lector

PABLO DANIEL MENDOZA PONCE (FIRMA)
Firmado digitalmente por PABLO DANIEL MENDOZA PONCE (FIRMA)
Fecha: 2023.02.09 11:01:13 -06'00'

Dr. Pablo Mendoza Ponce
Evaluador Independiente

Firmado por RONNY GIOVANNI GARCIA RAMIREZ (FIRMA) PERSONA FISICA, CPF-01-1137-0229.
Fecha declarada: 10/02/2023 12:50 PM
Esta representación visual no es fuente de confianza. Valide siempre la firma.

Dr. Ronny García Ramírez
Director de Tesis

Cartago, lunes 6 de feb. de 2023

Resumen

La automatización de celdas estándar permite reducir la necesidad de dibujar manualmente los transistores para que componen las diferentes celdas de una biblioteca, esto permite agilizar el proceso de generación de trazados correctos desde construcción y reduce la posible inserción de errores debido a la intervención humana. La necesidad de automatizar la generación de las celdas estándar aumenta conforme la complejidad de los Circuitos Integrados incremente por lo que reducir el tiempo que toma desarrollar una biblioteca permite agilizar el proceso de diseño de los circuitos ya que las bibliotecas de celdas estándar son el bloque fundamental del diseño físico.

Este trabajo presenta una solución al problema de automatizar la generación de celdas para una biblioteca utilizando un acercamiento que unifica los dos acercamientos existentes en la literatura, los generadores con base en procedimientos y los generadores con base en optimizadores. Con el fin de realizar dicho acercamiento, se plantea el uso de una plantilla de diseño que contenga las locaciones de los transistores para luego utilizar algoritmos de enrutamiento como optimizadores con el fin de generar una herramienta que permita reducir la complejidad a la hora de manipular las restricciones de diseño y generar diseños correctos desde construcción.

El acercamiento planteado incursiona de una manera novedosa al abordar el problema mediante la generación de plantillas abstractas para modelar el diseño y utilizar algoritmos de enrutamiento y compactación sin comprometer la calidad y capacidad del usuario de controlar el flujo de diseño con la finalidad de traducir la intención del diseñador a una implementación correcta desde construcción. La herramienta fue capaz de crear trazados cuya área difiere en menos de 20% para las celdas combinacionales cuyos retardos y consumo energético son comparables a los vistos por celdas de calidad industrial y permitiendo escalar los diseños de manera simple a un nodo tecnológico de menor tamaño. Por tanto los resultados observados presentan un panorama positivo sobre el acercamiento utilizado y motiva a pensar que, en el largo plazo, se puede automatizar el proceso de generación de bibliotecas produciendo diseños de calidad industrial reduciendo la interacción humana

Palabras clave: Circuitos Integrados, Automatización, Celdas Estándar, Trazado, Transistor.

Abstract

Standard cell automation reduces the need to manually draw the transistors that make up the different cells of a library, thus speeding up the process of generating correct by construction layouts and reducing potential insertion errors due to human intervention. The need to automate the generation of standard cells increases as the complexity of Integrated Circuits grows therefore reducing the time it takes to develop a library allows to speed up the circuit design process as standard cell libraries are the fundamental building block of physical design.

This document presents a solution to the problem of automating cell generation for a library using an approach that unifies the two existing approaches in the literature, procedure-based generators and optimizer-based generators. In order to carry out this approach, the use of a design template that contains the locations of the transistors is proposed to then use routing algorithms as optimizers in order to generate an automation tool that allows for reducing the complexity when manipulating the design constraints and generating correct by construction designs.

The proposed approach ventures in a new way by addressing the problem by generating abstract templates to model the design and use routing and compaction algorithms without compromising the quality and ability of the user to control the design flow in order to translate the designer's intent to a fully realized design. The tool was able to create layouts whose area differs by less than 20% for combinational cells whose delays and power consumption are comparable to those seen by industrial-grade cells, and scale the designs easily to a smaller technology node. Therefore, the observed results present a positive outlook on the approach used and motivates to think that, in the long term, the library generation process can be automated, producing industrial-quality designs by reducing human interaction.

Keywords: Integrated Circuits, Automation, Standard Cells, Layout, Transistor.

a mis queridos padres

Agradecimientos

El resultado de este trabajo no hubiese sido posible sin el apoyo de el profesor asesor Ronny García Ramírez quien ejerció como guía durante el proceso de desarrollo del proyecto, mis padres y hermano quiere brindaron su apoyo incondicional durante los estudios de maestría, profesores lectores Juan José Montero y Roberto Molina Robles, quienes dispusieron de su tiempo para servir como lectores y, finalmente, a los profesores y personal de la maestría quienes ayudaron durante el transcurso de los estudios de maestría.

Luis Felipe Retana Corrales

Cartago, 16 de Febrero, 2023

Contenido

Índice de figuras

Índice de tablas

Chapter 1

Introducción

El proceso de diseño de un circuito integrado (CI) implica esfuerzo de varios actores como lo son arquitectos, diseñadores, diseñadores físicos, ingenieros de implementación y otros, este recurso humano se agrupa en equipos de trabajo orientados a una tarea en específico de acuerdo a la etapa de diseño en el cual se encuentre el proyecto. Las etapas se dividen acorde al flujo de diseño, el cual involucra a todos los equipos de trabajo, a este flujo de diseño, se le conoce como flujo de diseño para circuitos integrados a gran escala, también conocido como *Very Large Scale Integration* ó VLSI por sus siglas en inglés. [29].

La figura 1.1 muestra un diagrama de bloques con los pasos que conforman el flujo de diseño de circuitos integrados a gran escala. El flujo de diseño VLSI consta de dos etapas principales: *front-end* y *back-end*. Cada una de estas etapas se subdivide conforme a la función realizada.

El *front-end* inicia definiendo las características funcionales del producto, a partir de dichas especificaciones se crea una descripción funcional o comportamental del diseño mediante un lenguaje de alto nivel y culmina en la síntesis comportamental, se le conoce como síntesis comportamental, por que el funcionamiento se captura a nivel de lenguaje de transferencia de registros también llamado *register transfer language* ó RTL por sus siglas en inglés y sus elementos registros y memoria mediante un lenguaje de descripción de hardware (*hardware description language*) también simplificado como HDL por sus siglas en inglés, como lo es en lenguaje Verilog.

El segundo paso es la realización de la síntesis física, este proceso toma la descripción a nivel de RTL como entrada y tiene como salida un trazado del diseño completo, este trazado se compara contra la especificaciones físicas con el fin de validar el funcionamiento, si estas comparaciones son positivas el diseño se envía a fabricación.

La síntesis física, la cual se denomina como la transición que existe hacia el dominio geométrico. La representación geométrica es conocida como trazado (*layout*) y se crea a partir de tomar los elementos pertenecientes a un archivo con la definición de celdas físicas e interconexiones llamado *netlist* proveniente de la síntesis de RTL (transistores y compuertas lógicas) y llevarlos a formas y capas específicas [16]. Una parte importante

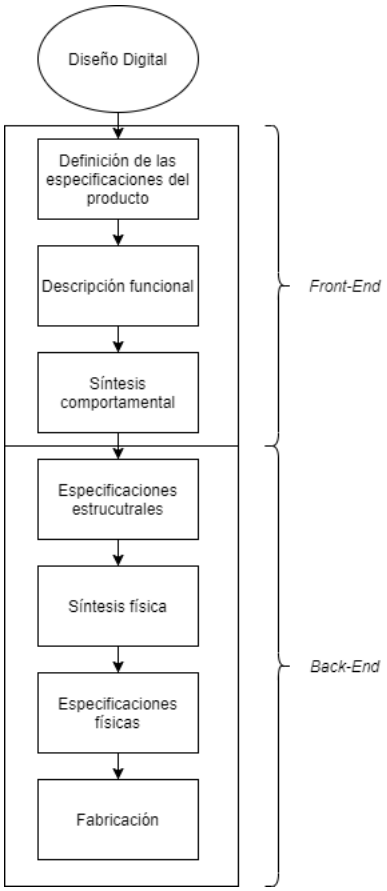


Figure 1.1: Diagrama de bloques que ejemplifica el flujo de diseño de Circuitos Integrados. Nótese como el flujo se divide en dos partes, el *front-end* y el *back-end*

de este esfuerzo se logra mediante la implementación de herramientas y metodologías de automatización conocidas como Automatización del diseño electrónico del inglés *Electronic Design Automation* también conocido como EDA, las cuales permiten recibir una descripción de alto nivel del diseño y traducirlo hasta llegar al trazado físico.

La síntesis de RTL toma el diseño a nivel RTL y lo mapea a diferentes componentes físicos agrupados en celdas combinacionales y celdas secuenciales. Estas celdas se entregan y se agrupan en conjuntos de altura estandarizada (misma altura entre los rieles de alimentación), emulando la modularidad de los bloques Lego®. En un armazón de bloques Lego®es posible reemplazar una misma pieza por otra de diferente color debido a su similitud dimensional. Este principio, es el mismo utilizado a la hora de implementar la síntesis física y al conjunto de celdas o compuertas lógicas se le conoce como una biblioteca de celdas estándar. Estas bibliotecas son generadas por diseñadores y entregadas a los diferente equipos de implementación para producir los CIs.

Una biblioteca contiene diferentes tipos de celdas asociadas a una función lógica, ANDs, ORs, Inversores, flops, latches y otras. Estas celdas se componen de una descripción a nivel de esquemático y diferentes tipos de vistas como la vista de diseño, vista física y una vista abstracta. Dichas vistas son necesarias para que las herramientas puedan leerlas

y utilizaras, por lo que se implementan diferentes estándares con el fin de facilitar la implementación.

Las bibliotecas, la fabricación y las herramientas son costosas e imponen una barrera de entrada al mercado por lo que una forma de mitigar dichas barreras de entrada es utilizar herramientas libres con el fin de desarrollar el flujo de diseño, se encuentran desde simuladores de HDL (icarus verilog y Verilator), hasta enrutadores como (ghdl y Fairly Good Router), pero pocas se centran en la capacidad de generar celdas hoja (*leaf cells*) de una biblioteca de celdas estándar.

Debido a los requisitos y barreras mencionadas que imponen una limitante a la capacidad de insertar productos tempranos al mercado, la automatización en la generación de celdas se ha convertido en una rama de investigación desde los años 90 [2], [10] e inicio de la década de los 2000 [22], creando dos vertientes de investigación: los generadores con base en procedimientos y los generadores con base en optimizadores. Los procedimientos utilizan plantillas de los diseños con el fin de agilizar la generación de las celdas y facilitar el escalamiento de los diseño sin comprometer la calidad del trazado. Herramientas como PyCell de Synopsys® y PCell de Cadence®, se ofrecen en la industria de EDA pero con la limitante de que los diseñadores ocupan amplio conocimiento de la tecnología sobre la cuál se trabaja. En contraste con los generadores procedimentales, se tienen los generadores con base en optimizadores los cuales buscan replicar las benevolencias del diseño digital mediante un acercamiento de divide y vencerás [6], [11], este enfoque ha sido predominante en las investigaciones académicas pero poco adoptadas por la industria debido a las limitaciones que presentan en la calidad de los trazados y a la complejidad de desarrollar dichas herramientas debido a la alta complejidad de los diseños modernos, los cuales ya no contienen únicamente celdas digitales si no que se espera que contengan elementos analógicos.

1.1 Motivación de la tesis

Diseñar, implementar y fabricar chips es un proceso complejo y de alta demanda creativa por lo que la experiencia de los diseñadores tiene un peso amplio en una implementación que pueda culminar en un chip funcional. Una de las áreas en dónde se refleja de primera mano la afirmación anterior, es en la generación de trazados de celdas estándar debido a que es un proceso iterativo en el cual la experiencia dicta, no sólo la calidad de la implementación, también condiciona la velocidad en la cual un trazado puede ser completado con una calidad alta de fabricación.

La industria utiliza acercamientos parametrizables con la finalidad de crear un repertorio de celdas las cuales puedan ser reutilizadas y escaladas para múltiples procesos, pero uno de los principales problemas del diseño de bibliotecas de celdas estándar es la realización de los diseños con la menor cantidad de violaciones a las reglas de diseño durante el proceso de migración a tecnologías más novedosas, en especial con la complejidad que las nuevas

tecnologías imponen [7]. Sin embargo, este acercamiento posee barreras de entrada debido a que grupos pequeños de diseño, diseñadores con poca experiencia en el área y grupos universitarios pueden encontrar bloqueos a la hora de diseñar, ejecutar e implementar las bibliotecas, un ejemplo mencionado con anterioridad es que las herramientas para generar trazados están sujetas a licencias que son provistas por las compañías las diseñaron esto implica un costo económico de entrada y en algunos casos, no aminoran la necesidad de conocer profundamente la tecnología. Otra barrera es la dificultad de utilizar herramientas de acceso libre, si bien existen contra partes de uso libre, la documentación, el soporte técnico o capacidad de soportar tecnologías modernas es limitada.

Este trabajo consiste en el desarrollo de un generador de celdas CMOS estándar utilizando un acercamiento con base en plantillas el cual permite la flexibilidad de interactuar en cualquier etapa de la implementación, esta plantilla pasa por un proceso de optimización automatizado produciendo una celda que puede utilizarse para fabricar CIs. La idea detrás de utilizar una plantilla es la capacidad de escalar hacia otras tecnologías el diseño implementado con el fin de reducir el recargo que existe sobre la pericia y experiencia del diseñador reduciendo así el tiempo para generar celdas estándar. Como se mencionó con anterioridad un área importante a la hora de escalar el diseño son las reglas de diseño. Esfuerzos en la predicción de las reglas de diseño se han propuesto [17] con la finalidad de contemplar el impacto de las reglas de diseño en etapas tempranas del proyecto con resultados positivos por lo que se plantea utilizar algoritmos de optimización para interconectar la celda con el fin de obtener un trazado que cumpla los requisitos de diseño, minimice la interacción humana y esté libre de violaciones a las reglas de diseño utilizando un punto intermedio entre los generadores de celdas procedimentales y los generadores de celdas con base en optimizadores con al finalidad de brindar al diseñador flexibilidad a la hora de implementar el diseño aminorando el compromiso entre la velocidad de implementación y la calidad de resultados.

1.1.1 Aporte de la tesis

Esta tesis constituye un acercamiento el cual trata de presentar una solución balanceada entre la interacción humana y la automatización del diseño electrónico con el fin de minimizar la dependencia que existe sobre la pericia y la experiencia del diseñador mediante el desarrollo de una herramienta que pueda ser implementada por el Laboratorio de Diseño de Circuitos Integrados del Instituto Tecnológico de Costa Rica con la finalidad de automatizar el proceso de generación de bibliotecas de celdas CMOS estándar.

La herramienta debe ser capaz de generar celdas que puedan ser implementadas en herramientas de calidad industrial por lo que las métricas a considerar son el rendimiento, el área y la potencia. Debido a la complejidad de implementar los diferentes formatos estandarizados por la industria, es necesario que la herramienta sea capaz de enlazarse con herramientas de edición de trazados como Custom Compiler de Synopsys con el fin de generar las vista requeridas y las simulaciones funcionales. Esto conlleva a la sigueintes

implicaciones:

- La herramienta debe ser capaz de leer el archivo correspondiente a la tecnología que se va a utilizar.
- La biblioteca resultante debe estar libres de violaciones a las reglas de diseño e implementar adecuadamente la función lógica.
- La herramienta debe poder crear bibliotecas en diferentes tecnologías.
- Las herramientas de caracterización como Custom Compiler y PrimeLib deben ser capaces de generar la vista de diseño de la biblioteca generada.
- La biblioteca debe ser utilizada en un diseño de una unidad pequeña mediante el proceso de síntesis lógica mediante la herramienta Custom Compiler.

1.1.2 Esquema de la tesis

Debido a los temas que deben ser abordados es necesario dividir la tésis en varios capítulos con la finalidad de mostrar la investigación y el aporte realizado por lo que el documento se divide de la sigueinte manera: en el capítulo 2, se presenta la teoría requerida para comprender el resto del texto, el capítulo 3 presenta la solución y la respectiva implementación. En el capítulo 4 se presentan los resultados y análisis y finalmente en el capítulo 5 y 6 se muestran las conclusiones y los anexos.

Chapter 2

Estudio Bibliográfico

Este capítulo provee el contexto necesario para comprender la solución propuesta en esta tesis, primeramente se presenta una introducción sobre el flujo semi personalizado, en la segunda sección se define el concepto de celdas estándar. Seguidamente, en la tercera sección, se introduce la definición de un generador de celdas y los diferentes acercamientos presentes en la literatura para generar celdas estándar y, finalmente, las técnicas de enrutamiento que se implementan para automatizar dicho proceso.

2.1 Automatización del Diseño Electrónico

Construir un Circuito Integrado (CI) es complejo, en especial considerando que la densidad de los chips se incrementa con cada generación implicando que los circuitos integrados modernos conllevan millones de transistores. Con el fin de lidiar con la complejidad de integrar los diseños, se desarrolló una técnica conocida como diseño semi-personalizado (*semi-custom design*), esta metodología parte de la premisa del uso de celdas pre-establecidas conocidas como celdas estándar.

Las celdas estándar se caracterizan por tener una altura definida conocida como *pitch* y permite una manipulación sencilla por las herramientas de automatización o EDA. Con la finalidad de facilitar la implementación de un diseño, las celdas estándar se agrupan de acuerdo a sus funciones y capacidad de manejar diferentes tamaños de carga (*drive strenght*), este agrupamiento se conoce como biblioteca de celdas estándar debido a la variedad de elementos que posee y la centralización de dichos componentes en un único conjunto.

Cada celda en una biblioteca de celdas estándar posee tres vista principales, la vista de diseño, la vista abstracta y la vista geométrica o física. Una vista, permite manipular la celda de acuerdo a la información que contiene, la vista de diseño contempla la información funcional o lógica (función lógica, diagramas de estado, entradas, salidas, definición de pines de alimentación y otros) y la información referente al temporizado de la celda (retardo combinacional, retardo secuencial, capacitancias en los pines de entrada, arcos

de tiempo entre los pines y demás), un ejemplo de la vista de diseño es la información recopilada por los archivos .lib, estos archivos están definidos en el estándar conocido como *liberty*. Por su parte, la vista abstracta contiene la información periférica de la celda como lo es la forma, el nombre de los pines y el metal asociado a dicho pin, vías e interconexiones internas, esta vista utiliza anotaciones debido a que no contiene información física de la celda, un ejemplo de la vista abstracta son los archivos .lef los cuales están definidos en el estándar LEF. Y finalmente, la vista física contiene la información geométrica referente a las capas de metal, como lo es densidad, código de colores, interconexiones, vías, capacitancias parásitas y demás, esta vista es conocida como ascii (debido a la encripción) y se puede citar como ejemplo los archivos OASIS, formato perteneciente a la compañía Cadence.

Cada una de las vistas pertenecientes a una celda son requeridas por diferentes herramientas de diseño con el fin de realizar alguna función específica, por ejemplo, la vista de diseño se utiliza durante el proceso de síntesis con el fin de generar un *netlist* optimizado en función de alguna arista como lo es área, potencia o rendimiento.

2.1.1 Síntesis Física

La síntesis física es el proceso en el que se toma un *netlist* a nivel de compuertas y se define el trazado del diseño, este trazado incluye las coordenadas de los transistores y de las líneas de metal [8].

El flujo de diseño de la síntesis física se descompone en los pasos visto en la figura 2.1 y toma como entrada el *netlist* a nivel de compuertas.

Diseño de planta

El diseño de planta o *floorplan* es la etapa de la síntesis física en la cual se asignan coordenadas fijas a los *macros* y se asignan pines y conexiones entre ellos. El *floorplan* debe garantizar que todos lo módulos del chip provenientes del *netlist* tengan una forma geométrica y una locación asignada con el fin de garantizar el colocado de compuertas y que cada pin que contenga una conexión externa tenga un espacio asignado de tal manera que se puedan generar interconexiones internas y externas.

Colocado

El colocado o *placement* consiste en asignar una locación fija a las celdas estándar, en esta etapa se realizan optimizaciones enfocadas a minimizar el largo de las interconexiones y mejorar el temporizado, estas optimizaciones se logran aplicando restricciones y directrices de diseño a las herramientas.

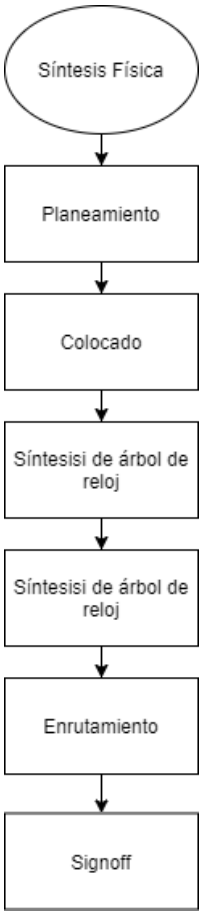


Figure 2.1: Diagrama de bloques que ejemplifica el flujo de síntesis física. Es de recalcar que el flujo se divide en varias etapas una de las cuales es el enrutamiento, área en que la tesis se enfoca.

Síntesis del árbol de reloj

La síntesis del árbol del reloj o *clock treee synthesis* es uno de los pasos más críticos a la hora de diseñar un IC debido a la delicadeza de la señal de reloj, en este paso, se interconecta la señal de reloj a lo largo del chip con la finalidad de optimizar el temporizado, este proceso es parcialmente guiado, indicando la distribución y características del árbol de buffer para los relojes, además de utilizarse un tipo especial de celdas para el reloj.

Enrutamiento

La etapa de enrutamiento o *routing* consiste en crear las interconexiones a lo largo del chip, las técnicas de enrutamiento más comunes son el enrutamiento global y el enrutamiento detallado. El enrutamiento global funciona como guía para determinar los caminos válidos a la hora de realizar las interconexiones, mientras que el enrutamiento detallado toma las guías determinadas por en enrutamiento global con la finalidad de crear las interconexiones con el menor largo posible entre ambos puntos. Este paso es completamente automatizado y tiene como finalidad crear las interconexiones sin provocar la violación de alguna regla

de diseño o afectación en el temporizado del chip.

Signoff

El *signoff* consiste en pruebas que garantizan que el diseño está listo para ser entregado para fabricación (*tapeout*). En esta etapa se verifican errores y se arreglan mediante el proceso de ECO (*Engineering Change Order*), estos arreglos son completamente manuales y dependen de la experiencia del ingeniero. En caso de un error grave, cabe la posibilidad de realizar una re-implementación con un *floorplan* completamente diferente.

2.2 Celdas CMOS Estándar

Una celda estándar es un diseño específico para cada una de las compuertas que se encuentran en una biblioteca y se utiliza la tecnología CMOS del inglés *Complementary Metal Oxide Semiconductor*. El diseño de las celdas estándar inicia con la definición de las pistas horizontales y verticales, una pista es el espacio requerido por una interconexión. Existen tres maneras de determinar el espaciado de una pista utilizando la distancia de centro a centro, estas son de línea a línea, vía a línea y de vía a vía [8].

$$d_1 = \frac{w}{2} + s + \frac{w}{2} \tag{2.1}$$

$$d_2 = \frac{w}{2} + s + Via_{overlap} + \frac{w}{2} \tag{2.2}$$

$$d_3 = \frac{Via_{size}}{2} + Via_{overlap} + s + Via_{overlap} + \frac{Via_{size}}{2} \tag{2.3}$$

Donde d es la distancia acorde a la ecuación, w es el ancho de la línea de metal, s es el espacio permitido entre las dos líneas de metal y $Via_{overlap}$ es la mínima distancia permitida entre una vía y el perímetro de metal que la rodea. La relación de las ecuaciones 2.1, 2.2 y 2.3 está dada por $d_3 > d_2 > d_1$.

Este acercamiento se ejemplifica en la figura 2.2, en el cual, se ordenan de izquierda a derecha como, línea a línea, línea a vía y vía a vía. El acercamiento de vía a vía permite manipular de manera más simple los diseños ya que la distancia de vía a vía es mayor que la distancia de pista a pista y pista a vía aunque aumenta el consumo de área.

Utilizando tecnología CMOS, una celda estándar se compone de un grupo transistores tipo P y un grupo de transistores tipo N. Las celdas en tecnologías CMOS se componen de ambos tipos de transistores que se alinean, usualmente, creando una fila de transistores P en la parte superior de la celda y una fila de transistores tipo N en la parte inferior de la celda, permitiendo conectar de manera simple los rieles de alimentación (VDD para el nivel de tensión lógico representado por un 1 y VSS para el nivel de tensión lógico

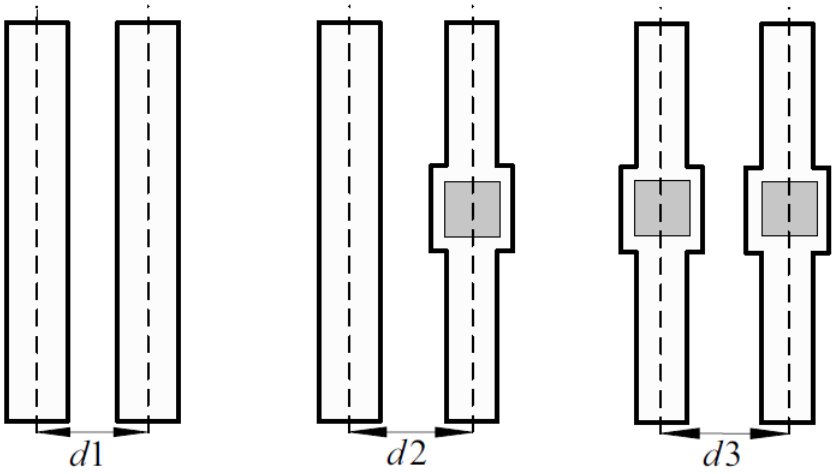


Figure 2.2: Ejemplificación de las reglas de diseño pista a pista, pista a vía y vía a vía. Nótese como la distancia varía dependiendo del acercamiento.

representado por 0). Estos rieles de alimentación son clave para determinar la altura de las celdas estándar debido a que dichos rieles deben ser lo suficientemente gruesos para mantener un flujo de corriente adecuado para los transistores.

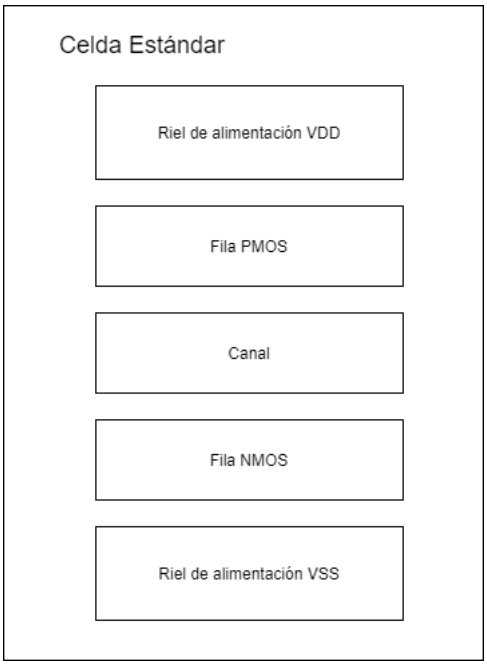


Figure 2.3: Plantilla genérica para construir celdas CMOS estándar. Se observa como se utiliza una estructura genérica con al finalidad de facilitar su uso por las herramientas de automatización.

En general, como se observa en la figura 2.3, una celda estándar se compone del riel de alimentación en la parte superior, seguido por una fila de transistores tipo P, un espacio de difusión entre los transistores P y los transistores N, la fila de transistores N y finalmente el riel de referencia.

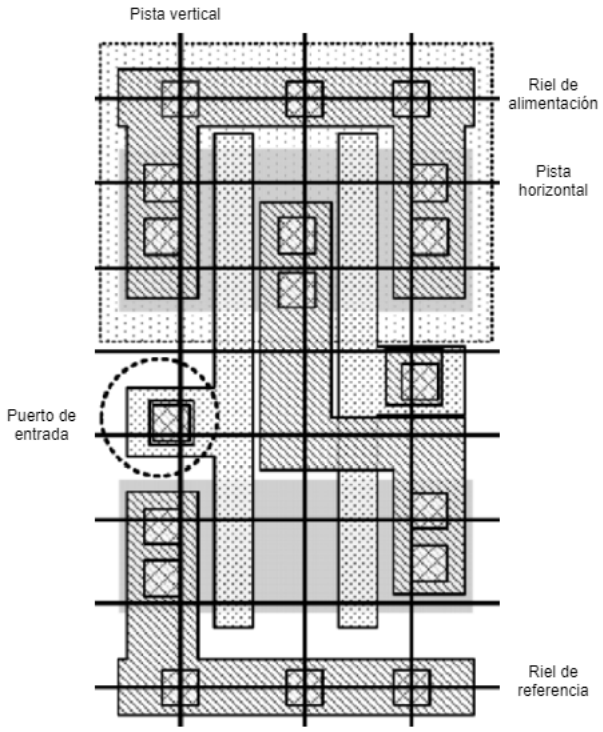


Figure 2.4: Ejemplificación del trazado de una celda CMOS estándar para una función lógica and, tomado de [8]. Obsérvese como los pines se alinean a las pistas

La figura 2.4 muestra una celda estándar con un mallado definido en el cual se demarcan las pistas horizontales y verticales de la celda. Estas pistas están alineadas siempre sobre los rieles de alimentación de una celda. También se observa que los rieles VDD y VSS poseen múltiples contactos, esta técnica se denomina como rieles completamente conectados o *fully strap* [8] y permite reducir la resistencia entre los rieles y las tinas N y P aumentando así la inmunidad a efectos como *latch-up*.

Es necesario saber la cantidad de capas de metal disponibles para realizar interconexiones en el diseño debido a que existen dos tipos de enrutamiento, enrutamiento local o interno y enrutamiento global. El enrutamiento local o interno son las interconexiones que existen de transistor a transistor mientras que el enrutamiento global son las interconexiones realizadas de celda a celda. Por lo que es deseable que las celdas estándar utilicen las capas de metales inferiores con el fin de liberar espacio para el enrutamiento global, el cual, consume la mayoría de las pistas de enrutamiento disponibles.

Otra característica es el acceso a los puertos, es deseable que los puertos utilicen metales bajos y estén alineados con las pistas para garantizar un acceso sencillo por las herramientas de enrutamiento automatizado con el fin de producir interconexiones de mayor calidad (menor cantidad de violaciones debidas a reglas de diseño).

Un conjunto minimalista de reglas de diseño se observa en [8] y se puede resumir en las siguiente reglas:

- Ancho mínimo de una tina tipo N.

- Distancia mínima entre dos tinas tipo N con el mismo potencial.
- Área mínima de una tina tipo N.
- Ancho mínimo de la difusión que define el ancho delos transistores NMOS y PMOS.
- Ancho mínimo de difusión en caso de interconexiones.
- Distancia mínima entre dos regiones de difusion.
- Mínima superposición tolerada entre una tina N y una región P+ dentro de la tina N.
- Espacio mínimo desde la tina N hasta la región P+ fuera de la tina N.
- Área mínima de la difusión.
- Ancho mínimo del poli para el canal de un transistor tipo NMOS.
- Ancho mínimo para interconexión realizada en poli.
- Espacio mínimo entre poli y la difusión en la capa de óxido.
- Espacio mínimo entre dos poli en el área de la capa de óxido.
- Extensión mínima de poli en la capa de óxido (*end-cap*).
- Área mínima del poli.
- Tamaño de los contactos.
- Espaciado de los contactos.
- Espacio mínimo de contacto difusión-poli.
- Ancho máximo de metal 1.
- Extensión mínima de metal 1 sobre contacto.
- Espacio mínimo para metal 1.
- Área mínima de metal 1.

2.3 Generadores de celdas

Prautsch [22], define un generador como: "*pieza de código procedimental que ejecuta comandos de manera secuencial*". A partir de la definición anterior, un generador de celdas se puede precisar como un programa que permite automatizar la generación de trazados de celdas mediante la ejecución de comandos de manera secuencial.

Los generadores de celdas se pueden categorizar de acuerdo a la implementación, Scheible [25] categoriza el acercamiento de acuerdo a dos vertientes, la primera de arriba hacia abajo (*buttom-up*) y una segunda de abajo hacia arriba (*top-down*). Las estrategias de arriba hacia abajo consideran un acercamiento de alto nivel, lo cual implica una independencia de la tecnología debido a que utilizan parámetros y condiciones durante la implementación con el fin de representar la intención del diseñador. Por su parte, el acercamiento de abajo hacia arriba inicia de una solución previamente concebida y el diseñador describe de manera procedimental el diseño. Ambos acercamientos requieren de validaciones físicas para garantizar la validez de la implementación ya que la intención del diseñador se encuentra implícita en ambos acercamientos.

2.3.1 Metodología de arriba hacia abajo

La metodología de abajo hacia arriba ataca el problema con un acercamiento de alto nivel. Este acercamiento considera que cada elemento es independiente del contexto en el cual se desenvuelve, como objetos vecinos o reglas de diseño dependientes del contexto en dónde se desenvuelve el diseño, debido a esta consideración, se utilizan parámetros definidos de manera explícita con el fin de plasmar la intención del diseñador. A esta metodología también se le llama utilizar acercamiento por optimización, este acercamiento utiliza optimizadores para automatizar el diseño. Un optimizador es una herramienta que, de manera similar al flujo de diseño digital, construye varios candidatos mediante la implementación de motores de optimización, estos candidatos son analizados con la finalidad de escoger una solución que se adapta de mejor manera a los especificaciones provistas por el diseñador, dichas especificaciones son requisitos de temporizado o un archivo de la tecnología que contiene las reglas de diseño (el motor de optimización se encarga de interpretar dichas especificaciones, abstraer la información necesaria y producir una solución adecuada).

La metodología de arriba hacia abajo trata de replicar el flujo de diseño digital [2], [6], [10], [11] en el cual se consideran una serie de pasos con la finalidad de generar un diseño que cumpla los requisitos necesarios, dichos pasos se pueden resumir similar a lo visto en [4]:

- Definición de la celda: Descripción de los elementos necesarios para construir una celda (transistores, interconexiones, contactos, vias y pines).
- *Clustering* y colocado global: Las regiones o tinas de los transistores se colocan de

manera adecuada.

- Colocado de los transistores: Los transistores se asignan una posición fija con coordenadas válidas.
- Enrutamiento: Las interconexiones entre las diferentes capas se definen por medio de un algoritmo que evalúa la viabilidad de dichas interconexiones.
- Compactación: La compactación se utiliza para lograr optimizaciones topológicas.
- Colocado de pines: Los pines se colocan de acuerdo a las especificaciones de diseño.
- Enrutamiento del metal: Los contactos (pines) previamente introducidos se interconectan por líneas de metal creando así el enrutamiento final del diseño.

El diseño inicia a nivel de esquemático del cual se captura un esquemático con el fin de construir una descripción estructural. El optimizador se encarga de concebir el diseño a partir de dicha descripción estructural (abstracción del esquemático) generando, finalmente, un trazado que cumple con las especificaciones de diseño, a este acercamiento también se le conoce como trazado a partir de esquemático.

El trazado resultante no necesariamente se considera óptimo debido a que las reglas de diseño afectan directamente el resultado y, al ser introducidas como parámetros de entrada (definidas explícitamente), son estáticas durante todo el proceso de diseño por lo que se requieren varias iteraciones con el fin de encontrar un diseño satisfactorio.

Otra característica de este acercamiento es el uso de algoritmos para la implementación de los trazados [13] [15], el enrutamiento global, erutamiento de canal y enrutamiento de caja cruzada (*switchbox*) [18] son comúnmente utilizados para generar trazados digitales. En la actualidad, se han utilizado acercamientos con base en la satifactibilidad booleana [24] el cuál busca generar trazados conscientes del proceso de manufactura.
Otra forma de optimización es la optimización topológica, en el caso de la lógica CMOS, utilizar los caminos de Euler, los cuales crean un grafo que permite crear una topología que minimiza el uso de las capas de metal. En el caso de la lógica secuencial debido a la complejidad agregada por los lazos de realimentación, otras técnicas son requeridas para generar un diseño, recientemente el uso de los grafos bipartitos [15] ha permitido mejoras substanciales en el acercamiento de arriba hacia abajo.

2.3.2 Metodología de abajo hacia arriba

Los trazados dependientes del contexto en el que se desarrollan se han introducido de manera estable en el flujo de diseño digital, especialmente en el área de los generadores de celdas, ejemplos comerciales de dicho acercamiento se pueden observar en herramientas como PCells ® y PyCells ®, generadores de celdas de las compañías Cadence y Synopsys respectivamente, estas herramientas son ambientes completos para la generación de celdas, normalmente parten de una descripción del diseño o plantilla (*template*) la cual produce

un diseño final mediante procedimientos ejecutados secuencialmente de tal forma que reproducen la intención del diseñador de manera directa (diseñador define los comandos o procedimientos que el diseño necesita).

Las entradas de esta metodología se pueden clasificar en dos grupos, parámetros e interfaces [22]. Los parámetros son entradas estáticas (la información de la tecnología) que se utiliza para mapear la tecnología mediante código procedimental, por su parte las interfaces son entradas dinámicas, esto significa que varían de acuerdo al contexto de las reglas de diseño conforme la tecnología varía [21]. En este acercamiento, las celdas son generadas de manera procedimental, siguiendo una serie de pasos con el fin de generar un diseño correcto desde construcción.

Los pasos básicos de este acercamiento se presentan en [27].

- Restricciones geométricas: Dependencias específicas de la tecnología son definidas.
- Instanciación del modelo: Se instancia el modelo que contiene una descripción abstracta del trazado con base en comandos estructurales (posiciones de las vías, pines, tinas y demás).
- Extracción de elementos parásitos: Se extraen los elementos parásitos (resistencias y capacitancias) del diseño.
- Evaluación del rendimiento: Se evalúa la viabilidad del diseño a partir de la información de la instancia.
- Optimización: Una herramienta que implementa alguna estrategia de modificación estructural a partir de las restricciones geométricas.

Estos pasos no necesariamente se ejecutan en el orden presentado y se suelen implementar en un bucle de iteraciones que permita garantizar la integridad del diseño una vez evaluado el rendimiento del trazado resultante.

Una de las desventajas principales es que la metodología utiliza considerablemente la intervención humana, en especial a la hora de definir las especificaciones y los modelos del trazado debido a la dependencia sobre la pericia del diseñador y a la dificultad de crear un modelo formal del diseño (plantilla) como consecuencia de las reglas de diseño dependientes del contexto en el cual la solución se va a desenvolver.

2.3.3 Acercamientos existentes: Optimizadores contra Procedimientos

Marolt [5] propone una categorización para los generadores de celdas con base en el manejo de restricciones o directrices de diseño (*design constraints*). Las restricciones de diseño se pueden categorizar de dos maneras, restricciones explícitas o formales e implícitas o no formales.

Las restricciones explícitas son aquellas que poseen una descripción no ambigua la cual no deja campo para interpretaciones debido a que establecen una relación o condición directa entre los elementos que restringe y puede ser verificada matemáticamente. Podemos considerar como ejemplo el tamaño mínimo entre dos vías de metal colocadas como interfaz entre metal 1 y difusión. Por su parte, las restricciones implícitas son ambiguas por lo que describen la interacción entre los objetos de manera no directa por lo que depende de la experiencia e interpretación del diseñador, un ejemplo de una restricción implícita es un comentario indicando que la vía V1 y la vía V2, no pueden estar muy cerca o podrían observarse problemas eléctricos. A partir de lo especificado con anterioridad se puede definir el manejo de las restricciones de acuerdo a los acercamientos existentes en la literatura, mientras los optimizadores utilizan exclusivamente restricciones formales, los generadores procedimentales utilizan restricciones no formales con el fin de generar el diseño final.

Optimizadores

Los optimizadores consisten en elementos de código que utilizan algoritmos de optimización con el fin de buscar una solución que cumpla con los requisitos del diseño. Normalmente, los optimizadores se dividen en dos etapas, exploración y evaluación [5], en la etapa exploratoria, el optimizador se mueve al espacio de la solución con el fin de refinar el candidato (trazado propuesta para solución), la etapa de evaluación consiste en validar el resultado de la etapa exploratoria esta etapa define si un trazado fue exitoso o no con respecto a algún criterio de parada.

Como se ha mencionado con anterioridad, los optimizadores son utilizados por el acercamiento de arriba hacia abajo para generar trazados, una de las principales ventajas es que al implementar las restricciones de diseño de manera explícita permiten una implementación de naturaleza abstracta, dicha capacidad de abstraer el problema garantiza flexibilidad a la hora de implementarse sobre diferentes diseños. Dicha flexibilidad también conlleva una desventaja la cual está ligada al proceso de abstracción, [4] se propone un acercamiento mediante símbolos, un símbolo es una abstracción que representa un elemento del trazado (transistores, vías y contactos), si bien los símbolos permite adjudicar independencia de la tecnología, la desventaja del acercamiento propuesto es que el proceso de síntesis no necesariamente produce arreglos regulares lo cual puede tener un impacto significativo en la topología del trazado candidato incrementado el tiempo de ejecución que se requiere para generar un diseño, o con la posibilidad de que el diseño no se satisfaga para las restricciones dadas.

En general, el proceso de abstracción consiste en tomar el problema y reducirlo en modelos más simples, estos modelos son, comúnmente, descripciones matemáticas (matrices, figuras geométricas, ecuaciones, expresiones algebraicas, entre otros) las cuales son optimizadas por algoritmos de optimización implementados por el optimizador, en otras palabras, el problema se reduce a resolver el modelo abstracto del problema planteado inicialmente.

La implicación directa es un incremento en la complejidad de implementar la herramienta debido a la necesidad de formalizar el conocimiento del experto con el fin de que la herramienta pueda producir un diseño adecuado pero no necesariamente óptimo. Con el fin de formalizar el conocimiento del experto, es necesario utilizar parámetros de entrada para que sean utilizados por los algoritmos los cuales se construyen a partir de conocimiento de automatización pero no necesariamente con conocimiento de diseño de trazados, además, como se mencionó con anterioridad, los parámetros se mantienen como elementos estáticos durante el diseño por lo que el optimizador se convierte en un elemento iterativo ya que se debe evaluar la interacción de dichos parámetros estáticos contra un criterio de parada para cada candidato generado por la etapa exploratoria.

En [2] se diseña una herramienta considera la síntesis (traduce de esquemático a transistores), el colocado de los transistores y el enrutamiento, pero posee dos limitantes, la primera limitación es que requiere que los transistores NMOS y PMOS sean duales, en otras palabras si se utilizan tecnologías que utilizan cantidades desiguales de NMOS y PMOS (Ej: Lógica Domino) se desperdicia área. La segunda limitante es, que es utilizada para transistores de tamaño idéntico, si esto no ocurre de dicha forma, el algoritmo no es eficiente para garantizar un área mínima y finalmente, si el diseño es complejo, el tiempo de ejecución es significativamente alto.

Una forma de mitigar el problema de el tiempo de ejecución es dividir las herramientas en varias fases, [11] se centra en lidiar con dos problemas el colocado y el enrutamiento, para el colocado se considera el área disponible y la capacidad de enrutamiento que el diseño poseer y para el enrutamiento, la consideración principal es encontrar el árbol de Steiner que conecta dos puntos, en general el algoritmo de Steiner permite encontrar una solución óptima dados dos puntos y ciertas restricciones de diseño. Cuando se presenta una violación de las reglas de diseño, se vuelve a lanzar una corrida sobre la estructura anterior, esto implica que si la corrida anterior tenía un enrutamiento relativamente malo, este se mantendrá a lo largo de la corrida hasta que se modelen las reglas de diseño y se arreglen afectando el tiempo de ejecución, también, la complejidad de implementar la herramienta aumentan considerablemente con el fin de minimizar la intervención humana durante el proceso de desarrollo lo cual implica una limitante a la flexibilidad de la herramienta.

A nivel de optimizadores para generación de celdas, se presenta un resumen de algunas implementaciones encontradas en la literatura.

La tabla 2.1, presenta una comparativa realizada para diferentes acercamiento encontrados en la literatura, en dicha tabla se observa que las herramientas poseen un compromiso de diseño, si las herramientas se diseñan con el fin de atacar el problema de una manera genérica, problemas complejos reducen la capacidad de optimizar el diseño por su parte, aumentar la capacidad de optimizar el diseño implica limitar los pasos que la herramienta puede actuar.

Referencia	Año	Acercamiento	Limitante
[2]	1991	Síntesis, colocado	Solo lógica complementaria
[4]	1992	Colocado, enrutamiento	Diseños con pocos transistores
[11]	2013	Colocado, enrutamiento	Diseños complejos comprometen la optimización
[24]	2012	Enrutamiento	Diseño con menos de 60 transistores
[15]	2018	Colocado, enrutamiento	Uso metal 2 es mayor para las celdas generadas

Tabla 2.1: Comparativa para diferentes de optimizadores presentes en la literatura, el acercamiento utilizado y la limitante. Nótese como los optimizadores poseen problemas de optimización si los diseños aumentan la complejidad de implementación.

Generadores procedimentales

Un generador procedimental se caracteriza por dos elementos principales, una interfaz de entrada que permite definir los parámetros de entrada para la entidades del diseño y una secuencia de comandos que ejecutan funciones que permiten generar trazados de manera similar a un *script*. Este acercamiento permite lidiar con problemas comunes que las bibliotecas personalizadas poseen, por ejemplo, el trazado final de una celda puede ser de aplicación específica, puede variar la tecnología, la forma del trazado no es capaz de utilizarse debido al área que consume, en otras palabras, no existen grados de libertad para un diseño establecido.

Los generadores procedimentales generalizan los diseño mediante la parametrización, aumentando la eficiencia y también brindado automatizaciones que permiten reutilizar diseños previamente establecidos aunque limitados por las capacidades que las herramientas ofrezcan como soporte para doble riel de alimentación o la capacidad de definir la orientación de las capas de metal a utilizar. Si bien los generadores imitan prácticas adecuadas de los diseñadores, representa una manera eficiente de capturar el conocimiento de los expertos con la desventaja de que requiere una preparación previa significativa antes de producir el diseño pues dichos expertos deben familiarizarse con la herramienta previa a su uso a un nivel más profundo comparado con los algoritmos de optimización cuya intención es encender y ejecutar (*plug and play*).

Es costumbre que a los generadores procedimentales se les llama celdas, y se pueden instanciar varias veces a lo largo del diseño o de manera jerárquica con el fin de crear diseños mas complejos, a este acercamiento también se les llama celdas parametrizadas [14] debido a que la descripción de la celda se genera mediante los parámetros de entrada y la subsecuente ejecución de los comandos se utiliza para obtener un diseño a partir de los parámetros de entrada del trazado, por ejemplo, el diseño de un inductor [19], consiste en la definición de los parámetros del trazado los cuales son: diámetro exterior, el número de vueltas, número de segmentos en un semicírculo), *pitch*, la anchura a las relaciones de espaciado en la parte más interna y vueltas más externas. Después de la definición de los parámetros, se realizan computaciones de acuerdo a las relaciones definidas entre

los parámetros para obtener la implementación adecuada, como lo son cálculo de radios, relaciones entre anchos y largos y demás y, finalmente, se ejecutan comandos específicos del generador procedimental para crear formas geométricas las cuales se les asocia un material del diseño como metales o difusiones.

Si bien, un generador procedimental replica el diseño generado por un experto, permite crear un portafolio de celdas que pueden ser reutilizados múltiples veces por el mismo equipo con el fin de mejorar el tiempo hacia el mercado, y garantizando que el portafolio contiene soluciones que fueron validadas en silicio aumentado la validez de los diseños concebidos por los generadores. Como se ha mencionado con anterioridad, los generadores procedimentales utilizan restricciones de diseño no formales debido a que el diseño ya ha sido concebido antes de implementarse, pero, este acercamiento también garantiza una mayor flexibilidad ya que permite manipular los comandos de diseño con el fin de garantizar que el diseño se puede re-ajustar de acuerdo a los requisitos del nodo tecnológico en el cual se desenvuelve sin necesidad de formalizar las restricciones mediante parámetros estáticos como si ocurre con los optimizadores, lo cual implica una menor complejidad para desarrollar la herramienta.

Referencia	Año	Acercamiento	Manejo de DRCs
[28]	2013	Colocado, enrutamiento	Depende del usuario (no optimiza)
[21]	2017	Colocado, enrutameiento	Depende del algoritmo de optimización
[20]	2016	Colocado, enrutamiento	Depende de un algoritmo de ordenamiento
[22]	2018	Colocado, enrutamiento	Depende de las estrategias disponibles

Tabla 2.2: Análisis de generadores procedimentales y la interacción con las reglas de diseño. Obsérvese como los generadores procedimentales depende de las estrategias disponibles y de los usuarios para generar diseños de alta calidad.

La tabla 2.2 presenta una síntesis de los diferentes acercamientos utilizados por los generadores procedimentales, también evidencia que la dependencia que existe sobre el conocimiento de la tecnología y la experiencia del diseñador al no poseer etapas de optimización y compactado con el fin de minimizar los DRCs o escalar el diseño hacia otras tecnologías debido a la dependencia que mantienen sobre ya sea, los algoritmos de optimización utilizados o de las estrategias implementadas y disponibles. También muestran dependencias de acuerdo al usuario que utiliza la herramienta, debido a que en las estrategias disponibles deben ser provistas por el usuario [28] o requieren un entendimiento de el tipo de optimización disponible lo cual complica el uso de la herramienta [21] [20] [22].

Los generadores procedimentales tienen uso mayoritariamente en la industria por encima de la academia. Se observan implementaciones en la literatura enfocados al desarrollo y automatización de bibliotecas pero con una finalidad de manipular elementos analógicos.

En el caso de los acercamientos vistos en la industria, no todos depende de restricciones no formales, se busca un acercamiento intermedio que permita cierto grado de independencia

del diseñador, debido a que entre más intervención humana para realizar las optimizaciones, el tiempo que se toma para desarrollar un diseño se incrementa.

2.3.4 Generadores de celdas: Conclusiones

A manera de síntesis, la comparación principal entre ambos acercamientos proviene del manejo de las restricciones de diseño y de la flexibilidad que estos ofrezcan. Los generadores de celdas con base en algoritmos optimizadores brindan un acercamiento que minimiza la interacción humana y son relativamente independientes de la tecnología lo cual garantiza una mayor flexibilidad, pero implica un aumento en la complejidad del diseño y desarrollo de dichas herramientas generando un compromiso entre la optimización del resultado y la flexibilidad.

Por su parte los generadores de celdas procedimentales aseguran diseños de calidad personalizada debido a la capacidad de replicar el diseño producido por un diseñador, pero implica que requiere una mayor preparación de antemano con el fin de generar el diseño adecuado ya que se necesitan herramienta especializadas que soporte dicho acercamiento, muchas veces propietarias lo cual dificulta el traslado de los diseños entre diferentes herramientas. Otro problema es la variabilidad de los diseños pues las diferencias entre nodos tecnológicos, especialmente lo más modernos es continua lo cual puede imposibilitar el uso de diseños anteriormente realizados.

Se propone un acercamiento mixto, en el cual se define de manera abstracta la plantilla de la topología del trazado con el fin de evitar limitar el uso de las restricciones de diseño y se utiliza un optimizador con el fin de buscar las interconexiones de manera automática aumentando la flexibilidad de la herramienta para varias tecnologías.

2.4 Enrutamiento

El enrutamiento es el proceso por el cual se realizan interconexiones de metal entre pines y puertos, estas interconexiones son globales o internas [26]. Una interconexión global ocurre cuando se interconectan un punto inicial perteneciente y un punto final que no pertenecen a la misma celda, por ejemplo, una conexión de celda a celda a nivel de chip, por su parte, una interconexión interna ocurren cuando se conecta un punto inicial y un punto final pertenecientes al mismo elemento, por ejemplo, en una compuerta AND, se realiza una conexión interna entre la salida de una compuerta NAND y la entrada de un inversor.

Los dos principales acercamientos propuestos para resolver el problema del enrutamiento se clasifican de acuerdo a si existe o no abstracción. En [16] se presentan dos categorías para los acercamientos utilizados en el proceso de enrutamiento, el enrutamiento con base en mallas (*grid based router*) y enrutamiento geométrico (como ejemplo se mencionan los enrutadores con base en segmentos *gridless tile-based router*).

El enrutaminto con base en mallas consiste en abstraer el problema a una matriz de puntos los cuales se les asigna una coordenada en el eje x y eje y con el fin de definir cuales puntos ocupan ser conectados y realizar dicha interconexión entre punto y punto [16], uno de los ejemplos mas representativos de dicho acercamiento es el algoritmo de Lee el busca el camino más corto entre dos puntos definidos en el mallado, una de las principales ventajas de este algoritmo es la simplicidad de implementación aunque uno de los problemas que conlleva es el tiempo de ejecución, ya que este se incrementa substancialmente de acuerdo a la topología que se requiera implementar debido a los bucles que utiliza. Para solventar dicha limitante algoritmos que mejoran el tiempo de ejecución se han propuesto, un ejemplo es el algoritmo de Dijkstra el cual introduce peso a los caminos o ejes existentes entre los puntos o nodos de un mallado, el concepto de peso permite discriminar varios caminos sin la necesidad de visitarlos pues la idea principal es encontrar el camino con menor peso pero puede producir resultados no necesariamente óptimos, como por ejemplo, violaciones a las reglas de diseño o problemas eléctricos además de aumentar el consumo computacional debido a las estructuras de datos utilizadas.

El enrutmamiento geométrico consiste recorrer un mapa de polígonos o segmentos y en definir dos tipos de segmentos, un segmento vacío y un segmento utilizado, el algoritmo recorre el plano y devuelve una colección de segmentos que representa una interconexión punto a punto, este acercamiento tiende a generar resultados de alta calidad debido a que las restricciones espaciales se toman en cuenta durante el proceso de generar las interconexiones, pero requieren gran capacidad computacional debido a que es necesario consultar el área constantemente y la asignación de costo a los caminos se realiza mediante esquemas complejos lo cual imposibilita su uso para diseño grandes.

Para ambos tipos de enrutamineto, los algoritmos se pueden extender y aplicar dependiendo de las necesidades, estos algoritmos se pueden clasificar de acuerdo al acercamiento utilizado. En [26] los autores presentan una clasificación básica de los algoritmos: algoritmos con base en grafos y algoritmos con base en la computación geométrica.

2.4.1 Algoritmos con base en grafos

En la industria VLSI, muchos problemas se pueden reducir a una construcción de un grafo, los diagramas de Euler son ejemplos de dichas reducciones. Los algoritmos con base en grafos se pueden subdividir de acuerdo a la implementación: algoritmos de búsqueda, algoritmos de expansión de árbol, algoritmos del camino más corto, algoritmos de pareo [1].

Algoritmos de búsqueda

Los algoritmos con base en grafos, poseen tres métodos básicos de búsqueda los cuales son: búsqueda en profundidad, búsqueda en amplitud y búsqueda topológica.

Búsqueda en profundidad La idea detrás de la búsqueda en profundidad es tomar un vértice del grafo y profundizar la búsqueda de dicho vértice hasta encontrar todas las aristas que se originan de dicho nodo o vértice, una vez que se recorrieron todas las aristas el algoritmo se devuelve al siguiente vértice que contiene aristas sin explorar, si asumimos que los vértices y los bordes poseen una constante de tiempo constante, el algoritmo posee una relación lineal con el tiempo de acceso a los vértices y bordes [26].

Búsqueda en amplitud La búsqueda en amplitud consiste en visitar todos los vértices adyacentes antes de explorar otro vértice. El algoritmo comienza en un vértice v, se buscan todas asa aristas adyacentes al vértice v y se colocan dichos vértices en una cola de prioridad, el vértice v se asigna como explorado y la cola continúa hasta explorar todos los vértices. Similar a la búsqueda en profundidad, este algoritmo depende del tiempo de acceso a los vértices de manera lineal por lo que si la cantidad de vértices tiende hacia el infinito, el tiempo tiende a crecer hacia el infinito.

Búsqueda topológica La búsqueda topológica se implementa de manera similar a la búsqueda de profundidad, la principal diferencia es que primero se visitan los vértices padres antes de los hijos. Por ejemplo, se tiene un vértice A, este vértice no posee padres, por lo cual se explora, seguidamente se observan las aristas y estas se comunican con los vértices B y C, pero B y C se comunican con E, F, G y H, debido a la lista de prioridad, los siguientes nodos a visitar son B y C, por lo que una posible búsqueda topológica podría ser A, B, C, D, E, F, G y H, otro ejemplo mas simple sería el ordenar el grafo de manera descendiente de acuerdo a los tiempos de acceso de los vértices.

Algoritmos de expansión de árbol

Los algoritmos de expansión de árbol consideran que los problemas de grafos se pueden dividir en un subconjunto de tal forma que el vértice que pertenece al subconjunto posee cierta propiedad P. La definición formal del problema es como sigue: Partiendo un grafo $G = (V,E)$, seleccione un subconjunto $V' \subseteq V$ de tal forma que V' posee una propiedad P. Una implementación común es el árbol recubridor mínimo o MNT, y se define como sigue: Dado un grafo con peso $G = (V,E)$, selecciones un subconjunto de aristas E de tal forma que $E' \subseteq E$ y E' induce a un árbol y el costo o peso total de las aristas sobre dicho árbol es mínimo. Tres métodos básicos de solución son el algoritmo de Prim, el algoritmo de Boruvka y el algoritmo de Kruskal. Este acercamiento busca optimizar la topología del árbol, por lo que si se utiliza en un problema donde la expectativa es encontrar un camino válido puede ser ineficiente ya que depende de la topología, para aumentar el rendimiento del algoritmo, es necesario utilizar estructuras de datos más complejas debido a que es un algoritmo recursivo y transita por todos los nodos (al menos una vez).

Algoritmos de camino más corto

Los algoritmos que bucan el camino más corto, son la base de las técnicas de ruteo global y ruteo detallado para un caso especial de grafo. Estos algoritmos se definen a partir de los vértices del grafo como sigue: Dado un grafo con peso $G = (V,E)$, selecciones un subconjunto de vértices $V' = \{v_i,v_j\}$, considerando que $V' \subseteq V$, de tal forma que V' sea el camino de menor peso o costo entre los vértices, el algoritmo de lee o *maze router* es un ejemplo de dicha implementación. Estos algoritmos poseen un alto costo computacional debido a que ocupan minimizar una función de costo, por lo que la estructura de datos debe ser capaz de almacenar toda la información que se requiere, debido a esto, utilizar estos algoritmos para implementaciones simples aumenta el costo computacional con un beneficio no muy grande comparado con otros algoritmos.

2.4.2 Algoritmos de geometría computacional

Los algoritmos con base en geometría computacional consisten en computar las intersecciones de dos o más segmentos en el dominio Euclidiano, uno de los más conocidos es el barrido de líneas.

El barrido de líneas [23] permite identificar si dos segmentos se se intersecan y el funcionamiento es como sigue: Se tiene una cantidad n de segmentos en un plano, dichos segmentos se ordenan aumentado el valor de la coordenada x. Un barrido es una representación de los puntos pertenecientes a las líneas encontradas en la coordenada x ordenadas por su coordenada y y una intersección ocurre si se tienen dos puntos consecutivos pertenecientes a diferentes segmentos (poseen misma coordenada x y y pero son diferentes segmentos). Si el punto detectado es un punto de inicio el segmento se agrega a una estructura de datos que permite realizar un seguimiento ordenando los puntos por sus coordenadas x y y [26].

La tabla 2.3 presenta una comparativa inicial de acuerdo a la complejidad de implementación y al costo computacional, cada uno de los algoritmos posee ventajas y desventajas de acuerdo al uso y a los requisitos de implementación, para el caso de implementación simple, significa que el código se puede implementar en un único bucle sin necesidad de utilizar estructuras de datos mas complejas que la definición del árbol, una complejidad media implica que se requieren estructuras de datos paralelas con el fin de realizar los cálculos requeridos, por ejemplo, en el algoritmo de camino más corto, normalmente se utiliza una lista de prioridad para guardar los resultados al expandir el frente de onda [1].

2.4.3 Enrutamiento en generadores de celdas

Los generadores de celdas con base en optimizadores utilizan algoritmos de enrutamiento con la finalidad de realizar las interconexiones internas y externas de la celda (pines y puertos).

Algoritmo	Ventaja	Desventaja
Búsqueda	Fácil implementación	Recorre todos los nodos de manera recursiva
Expansión de arbol	Produce un grafo mínimo	Depende del atributo a minimizar
Camino mas corto	Encuentra un camino mínimo	Complejo de implementar
Barrido de líneas	Produce resultados de alta calidad	Alto costo computacional

Tabla 2.3: Comparativa de las ventajas y desventajas paras los diferentes algoritmos utilizados en el proceso de enrutamiento. Es de recalcar que existe un compromiso entre resultados de alta calidad, el costo computacional y el costo de implementación.

En la literatura, los generadores de celdas con base en optimizadores, por lo general utilizan las técnicas convencionales de colocado y enrutamineto, en otras palabras, abstraen y modelan el problema con el fin de reducir el recurso computacional utilizado. Acercamientos modernos utilizan el algoritmo de Lee [15] pero con variaciones como el uso de árboles de Steiner virtuales, el árbol de Steiner busca encontrar la línea más corta entre dos nodos. Otro acercamiento es el uso de una abstracción booleana del problema, en [24] el problema se modela como una ecuación booleana y se enruta la celda de acuerdo a la solución de la ecuación, el problema es el tiempo en ejecución, pues la satisfactibilidad booleana es un problema complejo de resolver en tiempo polinomial y puede requerir alta capacidad computacional.

Los autores de [3] proponen ciertas especificaciones que un enrutador de celdas estándar debe seguir, de los cuales los principales se pueden resumir en:

- El algoritmo de enrutamiento debe ser independiente del trazado o la plantilla utilizada con el fin de parametrizar los recursos que dispone cada tecnología.
- El algoritmo de enrutamiento debe ser independiente de las reglas de diseño y debe aceptar parámetros de entrada que representen las reglas de diseño.
- Tamaño de las interconexiones debe ser un parámetro de optimización con el fin de minimizar la longitud de las líneas de metal en diseños grandes.

El acercamiento más utilizado por las herramientas EDA es el de dividir el problema en dos partes [16], generar una guía de enrutamiento y realizar las interconexiones con el fin de reducir la capacidad computacional requerida. Normalmente, la guía se utiliza para

definir posibles caminos desde los puntos iniciales hasta los puntos finales, la siguiente etapa consiste en el use de un enrutador detallado el cual devuelve la interconexión con la mayor calidad permitida por la guía de enrutamiento [9]. Este acercamiento se utiliza desde FPGAs (*Field Programable Gate Arrays* hasta herramientas de calidad industrial como IC Compiler de Synopsys o Innovus de Cadence.

2.4.4 Enrutadores: Conclusiones

Los enrutadores modernos optimizan los recursos computacionales mediante guía de enrutamiento sin comprometer la calidad del resultado de las interconexiones, si bien, acercamientos modernos plantean el uso de solucionadores booleanos o incorporando mejoras a los algoritmos de enrutamiento detallado [12], incrementan la complejidad de la implementación y, por consiguiente, el consumo de recursos computacionales.

Es posible utilizar enrutadores con base en computación geométrica, pero la complejidad de implementar la herramienta es mayor que abstraer el problema y formular un modelo a partir de la teoría de grafos, por lo que se propone combinar un acercamiento booleano para definir los estados de los nodos en el mallado (1 para nodo viable como interconexión y 0 como nodo no viable) y utilizar un algoritmo de enrutamiento detallado el cual visite los nodos de la malla y se guíe por cuales están definidos como viables por la guía de enrutamiento, este acercamiento permitiría reducir el consumo de recursos computacionales debido a que el enrutador detallado ya no ocuparía utilizar estructuras de datos complejas para generar las interconexiones y permitiría modelar las reglas de diseño en una etapa temprana (durante la generación de las guía de enrutamiento) permitiendo a la herramienta completar diseños minimizando el riesgo de violentar las reglas de diseño.

Chapter 3

Propuesta de Solución

La solución propuesta consiste en la implementación de un enrutador el cual toma como entradas un modelo de una celda y lo traduce a un modelo matemático (matriz) y finalmente general el elemento físico a partir del mallado definido.

La razón por la cuál la solución se enfoca en un enrutador, es debido a que se desea analizar la capacidad de los enrutadores para resolver el problema del enrutamiento considerando las reglas de diseño durante la generación de las interconexiones, por lo que implementar algoritmos de colocado se escapa del alcance de la tesis, debido a esto, el uso de una plantilla de diseño de una topología conocida, permite generar un marco de referencia con el cual comparar los diseños finales. De acuerdo a lo visto en el capítulo 2.3, los acercamientos con base en procedimiento dependen del la experiencia y pericia del diseñador, por su parte los optimizadores dependen de los algoritmos involucrados. Para efectos de la solución propuesta, se analiza un compromiso entre flexibilidad, simplicidad de implementación y calidad de resultados. Considerando estas tres aristas se escoge un acercamiento mixto el cual permite introducir abstracciones y optimizadores sin perder la flexibilidad de establecer control sobre los parámetros de diseño comprometiendo la diversidad de geometrías que se pueden implementar.

La idea de utilizar un modelo abstracto de la celda parte del hecho de que utilizar algoritmos de optimización permite universalizar el uso de la herramienta y no depender de expertos con el fin de generar trazados a diferencia de utilizar celdas parametrizadas. El acercamiento propuesto permite a el usuario definir restricciones de diseño como lo son bloqueos de enrutamiento de manera directa con el fin de guiar a la herramienta a alcanzar el diseño deseado sin necesidad de considerar la geometría de los polígonos como normalmente se hace con los generadores de celdas con base en procedimientos, pero también tiene la capacidad de utilizar restricciones de diseño no formales debido a que la plantilla del diseño final es provista por el usuario, esta plantilla consta de el colocado de los transistores y requiere que el usuario defina los puntos de inicio y final de cada interconexión de manera explícita.

La capacidad de recibir restricciones explícitas e implícitas por parte del usuario, permite

mitigar ciertas deficiencias que los acercamientos tradicionales poseen como se analizó en el capítulo anterior, debido a esto se utiliza un acercamiento mixto en el cual se parte de una plantilla de diseño, la cual el diseñador plasmó la intención en dicha plantilla y luego puede invocar optimizadores para transformar y optimizar el diseño con el fin de crear un diseño libre de violaciones a las reglas de diseño con la ventaja de poseer mayor control según lo visto en la sección 2.3.4 además de poder escalar el diseño de manera simple entre tecnologías mediante la configuración de los parámetros adecuados. Una desventaja de utilizar este acercamiento es que los diseños resultantes no están a la altura de la calidad de un diseño personalizado ya que las optimizaciones son dependientes de los algoritmos de optimización y no de una descripción detallada del diseñador.

El flujo de diseño consta de tres etapas: Configuración de la herramienta, Enrutamiento y Traducción.

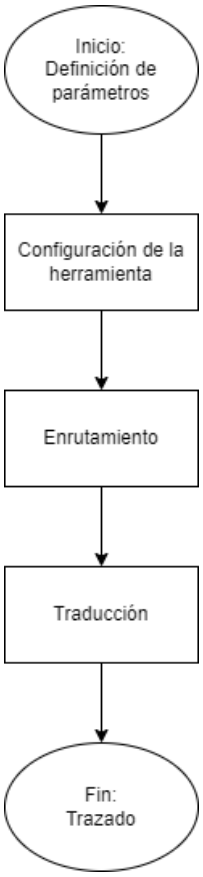


Figure 3.1: Flujo implementado para generar celdas automatizadas a partir de una plantilla. Nótese que las etapas de enrutamiento y traducción se implementan mediante algoritmos optimizadores minimizando la interacción humana.

La figura 3.1 presenta el flujo para generar una celda, este flujo consiste en varios pasos cuyo primer paso es la configuración de la herramienta. Este paso consiste en la lectura de la plantilla, la lectura del archivo de configuración de las reglas de diseño y la generación de un modelo formal por el cual en enrutador puede implementarse y crear el trazado final. Cabe resaltar que el usuario requiere conocimiento de la tecnología para desarrollar

la plantilla y definir las reglas de diseño que necesitan utilizarse.

Después de de configurar la herramienta, se prosigue con el proceso de enrutamiento, en este paso se realizan las intervenciones de los cables a vías o vías a cables utilizando algoritmos optimizadores para realizar dichas interconexiones por lo que no requiere interacción con el usuario. Finalmente se culmina con el paso de la traducción en donde se traducen los elementos colocados en la malla a elementos físicos en otras palabras se convierte de una malla a una geometría, este proceso se realiza de manera automática a partir de los parámetros provistos por el usuario durante la configuración.

3.1 Configuración de la herramienta y Mallado

Como se menciona con anterioridad, la etapa de configuración consta de tres pasos: la lectura de la plantilla de diseño, la lectura del archivo que contiene las reglas de diseño y, finalmente, la creación de la matriz.

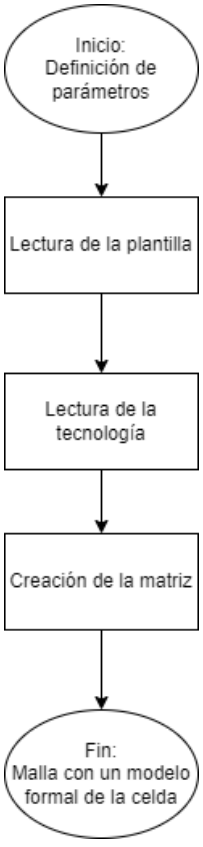


Figure 3.2: Diagrama que bloques que muestra la implementación utilizada para la etapa de configuración la herramienta la cual se constituye por la lectura de la plantilla del diseño, las reglas de diseño de la tecnología y la definición del mallado. Nótese que es requerido que el usuario configure adecuadamente la herramienta para evitar problemas durante los flujos posteriores por lo que requiere conocimiento de la herramienta y de las reglas de diseño básicas.

La figura 3.2 presenta el flujo que conlleva crear la malla sobre la cual se realiza el enrutamiento. El primer paso consiste en la lectura de la plantilla de diseño la cual contiene la definición de los elementos básicos como lo son los transistores y las vías. El segundo paso consiste en la lectura de las reglas de diseño concernientes a la tecnología y se define en un archivo llamado *techfile* y es provisto por el fabricante por lo que en la mayoría de los casos posee una estructura personalizada. Con el fin de permitir la compatibilidad de la herramienta con varias tecnologías, se opta por un acercamiento en el cual las reglas de diseño se especifican en un archivo de entrada con el fin permitir configurar diferentes tecnologías de manera simple, se decidió este acercamiento debido a la facilidad de escalar los diseños sin necesidad de considerar modelados complejos de reglas de diseño similar a lo visto en [28]. El archivo de la tecnología debe traducirse a un formato más simple para su lectura por lo que se escoge el formato JSON y mediante un analizador de sintaxis se toma el *techfile* y se crea un objeto JSON de la tecnología. Se utiliza el formato JSON debido a la facilidad de importar la bibliotecas existentes en C++, formatos como YAML fueron analizados pero se descartaron debido a la falta de soporte nativo del lenguaje. La biblioteca JSON utilizada se llama *JSON for Modern C++ version 3.9.1*, creada por el usuario nlohmann en github. Este analizador sintáctico permite leer como un objeto el archivo de la tecnología, este objeto se recorre utilizando las llaves y los atributos (similar a un diccionario), por lo que facilita la implementación de las reglas de diseño. La biblioteca utiliza una estructura similar a una tabla *hash* la cual se maneja por asociatividad entre llaves y atributos, facilitando la asignación de parámetros competentes a la tecnología. El uso de un formato genérico como JSON facilita la migración entre procesos debido a que los archivos de la tecnología poseen formatos propietarios, lo cual dificulta la migración entre tecnologías por lo que manejar un formato común facilita dicha implementación y JSON posee soporte nativo en C++ mientras que otros formatos no.

Finalmente se genera el modelado formal del mallado el cual tiene como base el uso de una matriz booleana debido a la simplicidad de implementación, en el cual cada punto de la matriz posee su propio par de coordenadas (desplazamiento horizontal y vertical) y un estado asociado que define si está bloqueado, si no hay ninguna interconexión presente o si es parte de alguna interconexión.

La idea de utilizar una malla parte del hecho que los algoritmos de enrutamiento con base en grafos requieren menos capacidad computacional que los algoritmos con base en geometría computacional [16] lo cual es una limitante a la hora de implementar el algoritmo en máquinas convencionales de recursos limitados.

El modelo de la matriz considera que la distancia entre punto y punto de la matriz equivale al área de una vía perteneciente a la capa de metal más inferior (metal 1 normalmente), por lo que para definir la cantidad de puntos de la matriz es necesario considerar que existe una resolución mínima requerida para representar la celda, esta resolución mínima es una relación entre el área y el tamaño del nodo.

$$P_{(x,y)} = \frac{A_C}{A_V} \tag{3.1}$$

La ecuación 3.1 representa la resolución mínima de la matriz, donde $P_{(x,y)}$ representa la cantidad de puntos de la matriz para el eje x y para el eje y, A_C es el área de la celda y A_V es el área de la vía. Esta transformación se realiza de manera automática en caso de no indicarse lo contrario, si el usuario especifica el tamaño de la matriz, entonces se toman los puntos definidos para las coordenadas x y coordenadas y definidos por el usuario como punto de partida.

Esta relación posee un tamaño mínimo también llamado resolución mínima y determina un tamaño de acuerdo a las reglas de diseño utilizadas, en este caso se utiliza una regla de diseño para espaciamiento de vía a vía, de acuerdo a lo visto en la sección 2.2, utilizar una regla de diseño vía a vía si bien produce diseños menos óptimos, permite simplificar la implementación debido a que implícitamente las reglas de vía a vía cumplen las demás, por ende la relación 3.1 debe definirse en términos de vías, por lo que la cantidad de puntos realmente significa cuantos puntos (tamaño de una vía) permiten representar adecuadamente el trazado final.

3.2 Enrutador

Un enrutador es el elemento que permite crear interconexiones entre dos puntos uno inicial y uno final a estos puntos se les conoce comúnmente como vías, dichas vías son puntos de acceso a capas inferiores o superiores de metal permitiendo así la transmisión de señales eléctricas a lo largo del trazado del circuito.

Se realizaron pruebas heurísticas para corroborar un acercamiento que permitiese correr el enrutador en máquinas de poco recursos, se probaron tres acercamientos, el algoritmo de Dijkstra, el algoritmo de Lee y un acercamiento con base en guías de enrutamiento y un enrutador detallado. La principal limitante del algoritmo de Dijkstra y el algoritmo de Lee es que necesitan una capacidad computacional alta, en especial si la resolución de la malla o la cantidad de nodos es alta debido a la necesidad de utilizar colas de prioridad y frentes de onda con el fin de encontrar el camino más corto entre dos puntos con el fin de seleccionar el camino de menor costo, el cual se coloca al frente de la cola de prioridad y banderas o estados de los nodos que deben ser guardadas con el fin de validar el resultado obtenido al final por lo que es posible exceder la capacidad computacional de la máquina, en algunos casos, el algoritmo no completaba correctamente.

El acercamiento menos costoso fue dividir el problema en dos partes similar a como los enrutadores modernos atacan el problema [1]. La primera parte consisten en la generación de una guía de enrutamiento, de acuerdo a los algoritmos vistos en la sección 2.4. El algoritmo que se ajusta de mejor manera son los de camino más corto debido a que la idea es encontrar un camino válido que minimice la distancia entre dos puntossiendo esta solución correcta y mínima. La segunda parte consiste en crear un grafo de interconexiones

mediante el uso de el algoritmo búsqueda de profundidad, en este caso las mallas son relativamente pequeñas por lo que el consumo computacional no es un factor a considerar siempre y cuando el problema se defina de manera simple. La solución propuesta plantea que el mallado se defina de manera booleana en el cual el estado de algún nodo de la malla es acertado por un algoritmo que define si los puntos de la malla son válidos (Verdadero) o inválidos (Falso), este acercamiento permite reducir el costo computacional mencionado en la sección 2.4 debido a las estructuras de datos que se requirieren utilizar lo cual permite que el algoritmo de Lee (algoritmo de distancia mínima) se desplace sobre los nodos marcando generando las aserciones necesarias. Con el fin de mejorar el rendimiento del algoritmo (perfil de memoria y velocidad), se crea una restricción de enrutamiento la cual define una caja de tamaño AxB, donde A es el punto de inicio y B es el punto de llegada en el cual el algoritmo de Lee se ejecuta (menor distancia que recorrer). Una vez encontrado un camino válido, se guardan los estados de los nodos en la malla y el proceso se reinicia una vez mas por cada interconexión a realizar sin necesidad de guardar estados intermedios de la malla minimizando así la cantidad de recurso computacional utilizado. Una vez generadas las guías de entrutamiento, las interconexiones se generan mediante un algoritmo de búsqueda de profundidad, la idea es recorrer los puntos de la malla que se encontraron por el algoritmo de Lee y asignarlos a una estructura de datos que guarda las interconexiones y los puntos se deben convertirse en polígonos.

El enrutador ejecuta dos etapas, la primera etapa toma la matriz como entrada con la finalidad de crear una guía para que el enrutador realice el proceso de conectar los puntos, esta guía de enrutamiento se realiza en mallas asociadas a diferentes capas de material, por ejemplo, existe una malla para metal 1 y otra malla para metal 2 sobre las cuales el algoritmo actúa aunque dependen de la definición del usuario (el algoritmo actúa sobre una malla especificada, el usuario necesita darle a entender al algoritmo que es necesario moverse a capas superiores). La segunda etapa permite definir las conexiones punto a punto entre el punto de partida y el punto de llegada, por ejemplo, se decide implementar una interconexión llamada A, la herramienta toma dicha interconexión y genera una guía de enrutamiento (primera etapa), seguidamente el algoritmo toma el resultado de la primera etapa y recorre la malla encontrando los puntos válidos pertenecientes a la interconexión A y marcando cono punto bloqueado o inválido el nodo para subsecuentes interconexiones. El proceso culmina en la generación de un mallado con las interconexiones definidas y traducidas al mudo físico almacenadas mediante un grafo con la definición de las interconexiones de acuerdo a los materiales utilizados.

La figura 3.3 representa el flujo que utiliza la herramienta para crear las interconexiones. Se observan dos etapas, la primera llamada aserción de los nodos la cual consiste en colocarse sobre un nodo de la matriz y validar si es posible utilizar dicho nodo como parte del camino válido entre el punto de partida y el punto de llegada, esta aserción utiliza valores binarios para identificar si el espacio de trabajo es válido o no. La segunda etapa es la generación de interconexiones, en esta etapa se toman los puntos de inicio y llegada y se interconectan mediante la guía de enrutamiento creada por las aserciones. Un punto puede tener varias condiciones asociadas, estas condiciones se pueden asignar de acuerdo

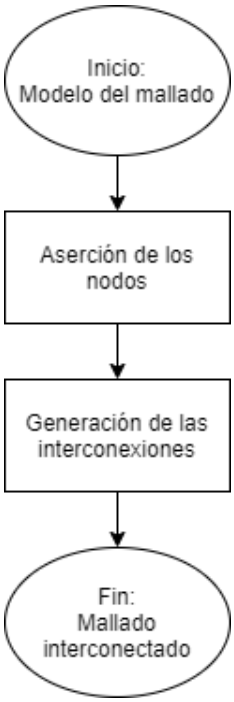


Figure 3.3: Diagrama de bloques que muestra el acercamiento utilizado para implementar el enrutador. Obsérvese como si divide en dos etapas las cuales implementan una guía de enrutamiento (aserción de nodos) y enrutamiento detallado (generación de interconexiones y traducción al mundo físico).

a la posición de los elementos en la malla. Estas condiciones se dividen en: punto sin bloqueo, punto bloqueado, punto válido y punto inválido. El punto sin bloqueo y el punto bloqueado dependen del estado anterior de la matriz mientras el punto válido y el punto inválido se asignan en el estado actual.

Un punto se denomina bloqueado si su valor durante el proceso de aserción sobre la malla es diferente de 0 mientras que un punto sin bloqueo posee un valor de 0. Un punto se define como válido si posee un valor de 1 en la malla durante el proceso de enrutamiento, si su valor es de 0 se define como inválido.

3.2.1 Aserciones

Las aserciones de realizan de manera sectorizada, se toma el punto de inicio y el punto final y se define un espacio válido de aserciones que comprende todos los puntos que se encuentran dentro del punto de inicio y el punto final, en general el área válida para realizar aserciones comprende lo siguiente:

$$\delta_X = X_{final} + \alpha_{cmp} - X_{inicio} + \alpha_{cmp} \tag{3.2}$$

$$\delta_Y = Y_{final} + \alpha_{cmp} - Y_{inicio} + \alpha_{cmp} \tag{3.3}$$

La ecuación 3.2 está asociada a los desplazamiento en el eje X, donde X_{final} representa el punto final, X_{inicio} representa el punto inicial y α_{cmp} es un factor que aumenta el área donde las aserciones pueden realizarse dado por un entero positivo. Por su parte, la ecuación 3.3 limita el desplazamiento en el eje Y en donde Y_{final} representa el punto final, Y_{inicio} representa el punto inicial y α_{cmp} es un entero positivo compartido con la ecuación que limita el desplazamiento en X.

Las ecuaciones 3.2 y 3.3 representan un área permitida por el cual se restringe al enrutador, a este bloqueo se le llama guía de enrutamiento similar al concepto de enrutamiento global. Si no es posible encontrar una solución válida, dicha diferencia se escala por un factor de compensación α_{cmp}, este factor es un número entero definido por el usuario durante la configuración, tiene como valor predeterminado 1. Este incremento en área se controla por un factor de iteraciones definido por el usuario, la idea es desplazar el punto de inicio una cantidad de puntos α y de igual manera se utiliza sobre el punto final, esto permite expandir el área de enrutamiento con la finalidad de crear nuevos caminos desde el punto de inicio hasta el punto final.

Algoritmo 1 Aserción del nodo

```
PARA TODO punto(x,y) HAGA
  SI vlido(punto(x,y)) ENTONCES
    punto(x,y) 1
  Ó
    punto(x,y) 0
  FIN SI
FIN PARA TODO
```

El pseudo código para la implementación se observa en el algoritmo [1], este algoritmo verifica si un punto es válido o no. El punto se define como válido si cumple los siguientes requisitos:

- Se encuentra entre el punto de inicio (0,0) y el punto final de la matriz (X,Y).
- No existe ningún bloqueo en el punto de la malla.
- Se encuentra dentro de la sección limitada para realizar las aserciones.

Este acercamiento se denomina una aserción. La aserción define un estado de válido o inválido de acuerdo al contexto del elemento sobre el que se realiza, en este caso el nodo. Cada nodo que se encuentra dentro del área restringida por el segmento debe ser visitado con el fin de definir un estado, válido o inválido. Se utiliza un algoritmo de búsqueda de amplitud, este acercamiento permite reducir la cantidad de líneas de código a costa del rendimiento.

La implementación del algoritmo de búsqueda se realiza mediante la recursividad. Se define una función que permite realizar las aserciones en los nodos y dentro de esta función se implementa la recursión como medio para recorrer la matriz.

Algoritmo 2 Desplazamiento sobre la matriz para realizar las aserciones

PARA TODO $punto_{(x,y)}$ **HAGA**
  SI $punto_{(x,y)} = Fin(x,y)$ **ENTONCES**
    retornar 1
  FIN SI
  SI $vlido(punto_{(x,y)})$ **ENTONCES**
    Asignar $punto_{(x,y)}$ 1
    SI $punto_x, < puntofin_X$ **ENTONCES**
      assertar($punto_{(x+1,y)}$) ? retornar 1 : retornar 0
    FIN SI
    SI $punto_{y,<puntofin_Y}$ **ENTONCES**
      assertar($punto_{(x,y+1)}$) ? retornar 1 : retornar 0
    FIN SI
    SI $punto_{x,>puntofin_X}$ **ENTONCES**
      assertar($punto_{(x-1,y)}$) ? retornar 1 : retornar 0
    FIN SI
    SI $punto_{y,>puntofin_Y}$ **ENTONCES**
      assertar($punto_{(x,y-1)}$) ? retornar 1 : retornar 0
    FIN SI
  Ó
    retornar 0
  FIN SI
FIN PARA TODO

EL algoritmo [2] presenta el pseudo código implementado para recorrer la matriz, este algoritmo toma un nodo, valida el resultado e inmediatamente se desplaza hacia el siguiente nodo más cercano en el eje X, si dicho punto falla se prueba el siguiente nodo moviéndose en el eje Y, si todos los movimientos posibles fallan, se regresa un falso indicando que no hay un camino válido y por ende el segmento de enrutamiento válido no existe.

Enrutamiento

Una vez que la malla ha sido acertada, se procede con el enrutamiento, el algoritmo de enrutamiento utilizado es el algoritmo de Lee conocido como *maze router*. El algoritmo de Lee busca encontrar un camino válido entre 2 puntos. El solucionador se desplaza por los nodos con valor 1 e ignora los nodos que no tengan dicho valor.

Un punto se define como válido si cumple los siguientes requisitos:

- Se encuentra dentro de la malla o la matriz.
- Fue definido como válido por las aserciones.

Los puntos identificados como válidos se almacenan en una estructura de tipo mapa cuyos

Algoritmo 3 Solucionador de el mallado

PARA TODO $punto_{(x,y)}$ **HAGA**
  SI $punto_{(x,y)} == 1$ **ENTONCES**
    guardar $punto_{(x,y)}$
  Ó
    ignorar $punto_{(x,y)}$
  FIN SI
FIN PARA TODO

valores contemplan una llave designada por el nombre de la *net* y una tupla que contiene las coordenadas X y Y de los puntos válidos de la matriz, el almacenamiento permite identificar cuales puntos y cuales elementos deben de ser mapeados a un elemento físico.

El desplazamiento sobre la matriz para rutear las *nets*, funciona de manera similar a el proceso de restringir mediante aserciones las *nets*, la diferencia principal, es que a la hora de rutear una *net*, la *net* se guarda en una estructura de datos.

Una de las principales limitaciones de este acercamiento es que permite trabajar únicamente con ángulos rectos, esta limitación ocurre debido a la simplicidad del algoritmo utilizado por lo que limita las topologías a utilizar, en otras palabras, las topologías que el algoritmo soporta son topologías simples a costa de incrementar el área o las capas superiores para realizar interconexiones.

3.2.2 Conversión física

La conversión física se realiza mediante el uso de las reglas vía-vía la razón de utilizar vía a vía es debido a la simplicidad de implementación como fue descrito en el capítulo 2.2 pues el uso de las reglas vía a vía implica que las demás reglas no se violentan debido a la distancia utilizada.

En este paso se asignan dimensiones físicas las diferentes capas (poly, diff, metales y otras) y la conversión cosiste en tomar las capas predefinidas por los elementos colocados y traducirlas a las capas definidas en el archivo de la tecnología con sus respectivos tamaños, una limitación de este acercamiento, es que el tamaño es consistente para todos los elementos, en otras palabras es una operación global y no selectiva (todos los elementos van a ser del mismo tamaño). Se utiliza un algoritmo de búsqueda a profundidad para cada capa de material con el fin de asignar una dimensión física a los polígonos.

Para traducir del dominio de los algoritmos al dominio físico, es necesario tomar en cuenta que el algoritmo asume que el punto de inicio es (0,0) y el punto final es (X,Y), esta consideración inicial implica que todos los puntos a traducir deben ser enteros positivos. La traducción ocurre en dos etapas, en la primera etapa se traducen los elementos previamente colocados (poly, difusión y demás), estos elementos tienen posiciones conocidas para el enrutador, de segundo se traducen las inteconexiones de metal las cuales son los

Algoritmo 4 Desplazamiento sobre la matriz para resolver el enrutamiento

PARA TODO $punto_{(x,y)}$ **HAGA**
  SI $punto_{(x,y)} = fin(X,Y)$ $valido(punto_{(x,y)})$ **ENTONCES**
    retornar 1
  FIN SI
  SI $valid(punto_{(x,y)})$ **ENTONCES**
    guardar($punto_{(x,y)}$)
    SI $punto_x, < puntofin_X$ **ENTONCES**
      validar($punto_{(x+1,y)}$) ? retornar 1 : retornar 0
    FIN SI
    SI $punto_y, < puntofin_Y$ **ENTONCES**
      validar($punto_{(x,y+1)}$) ? retornar 1 : retornar 0
    FIN SI
    SI $punto_x, > puntofin_X$ **ENTONCES**
      validar($punto_{(x-1,y)}$) ? retornar 1 : retornar 0
    FIN SI
    SI $punto_y, > puntofin_Y$ **ENTONCES**
      validar($punto_{(x,y-1)}$) ? retornar 1 : retornar 0
    FIN SI
  Ó
    retornar 0
  FIN SI
FIN PARA TODO

elementos que se conectaron durante el proceso de enrutamiento.

El algoritmo de traducción consiste en determinar el tamaño mínimo de la interconexión y la capa de metal asociada y colocar la interconexión en el centro del nodo, después se asigna el metal a la interconexión, hecho esto se crea un polígono cuyo ancho es de tamaño mínimo y largo equivale a una vía. Se debe encontrar un límite superior y un limite inferior que cubra el ancho mínimo de una *net* de metal. El límite superior se puede encontrar de la siguiente manera:

$$IntrcnxN_{(X,Y)} = \frac{puntopartida_{(x,y)} \times Via_L + puntollegada_{(x,y)} \times Via_L}{2} + \frac{Met_{NW}}{2} \quad (3.4)$$

En dónde $puntopartida_{(x,y)}$ representa el nodo inicial de la malla, $puntollegada_{(x,y)}$ el punto final de la malla, Via_L es el largo de la vía del metal respectivo y Met_{NW} representa el ancho mínimo permitido para una interconexión del metal determinado.

Por su parte el límite inferior se puede encontrar de la siguiente forma:

$$IntrcnxnN_{(X,Y)} = \frac{puntollegada_{(x,y)} \times Via_L + puntopartida_{(x,y)} \times Via_L}{2} - \frac{Met_{NW}}{2} \quad (3.5)$$

En dónde $puntopartida_{(x,y)}$ representa el nodo inicial de la malla, $puntollegada_{(x,y)}$ el punto final de la malla, Via_L es el largo de la vía del metal respectivo y Met_{NW} representa el ancho mínimo permitido para una interconexión del metal determinado.

Las ecuaciones 3.4 y 3.5 permiten asignar un polígono relleno de metal a las coordenadas en el espacio creando así pequeños segmentos de metal hasta construir la *net* en su totalidad. Estas ecuaciones funcionan de la siguiente manera: se encuentra el nodo al cual la *net* de metal pertenece, este nodo proviene de la matriz solución del problema del enrutamiento, el punto de inicio del nodo $startpoint_{(x,y)}$ está dado por el punto (x,y) de la matriz solución en el cual se determinó que hay un segmento válido de la *net* multiplicado por la dimensión mínima del nodo y el punto final $endpoint_{(x,y)}$, se obtiene al sumar el ancho del nodo, en este caso, el ancho de una vía, de esta manera se asigna un valor geométrico al nodo, este se divide entre dos para obtener el punto medio de la distancia entre ambos puntos en el plano y se toma el ancho mínimo de la línea de metal y se divide entre dos para sumar y restar de acuerdo al límite que se está buscando. Este acercamiento también se utiliza para definir reglas de diseño entre bloques de IP y celdas estándar durante el proceso de colocado de los transistores.

Algoritmo 5 Algoritmo para traducir las interconexiones

```
1: PARA TODO nombre_int EN intrcnxns(nombres_int,puntos_(x,y)).llaves() HAGA
2:   PARA TODO punto_(x,y) EN intrcnxns(nombre_int,puntos_(x,y))[nombre_int] HAGA
3:     asignar_dimensiones(punto_(x,y))
4:   FIN PARA TODO
5: FIN PARA TODO
```

El algoritmo [5] se utiliza para identificar los movimientos en el eje X y los movimientos en el eje Y. Durante el proceso de enrutamiento si el punto es válido se almacena con un identificador de la *net* pertinente con la finalidad de traducir adecuadamente las *nets* hacia el trazado. Los datos se guardan en una estructura de diccionario cuyas llaves son los nombres de las *nets* y los valores asociados a las llaves son tuplas con los puntos pertenecientes a la *net*. Los valores deben ser únicos por lo que se eliminan duplicados y se ordenan de manera ascendente de tal forma que las *nets* se descomponen en líneas rectas que se desplazan en el eje x y en el eje x, las intersecciones se identifican cuando existen líneas que comparten alguna coordenada pero difieren en la otra. Identificar intersecciones es relevante debido a que el algoritmo crea cruces en las *nets* que podrían violentar alguna regla de diseño.

Finalmente se asignan las dimensiones a cada *net* dividiéndolas en segmentos, la división se realiza utilizando líneas rectas sobre el eje X y sobre el eje Y, debido a que las *nets* fueron previamente ordenadas la división en líneas rectas es un acercamiento que facilita la optimización de las conexiones.

Optimización del trazado

La etapa de optimización busca generar un diseño correcto, entiéndase correcto como sin violentar reglas de diseño a nivel del trazado, esto se realiza mediante un acercamiento llamado algoritmos de compactación implementados por herramientas llamadas compactadores. Con este fin se diseña un compactador que utiliza un algoritmo relativamente simple que se compone de dos etapas.

La primera etapa consiste en construir un grafo el cual contiene la interacción de un polígono con el resto de los elementos del trazado, en este grafo se guardan las magnitudes que ocupen que resulte de analizar todas las interacciones posibles entre el objeto y el entorno. Con el fin de reducir la complejidad del algoritmo se toma en cuenta dos situaciones, la vecindad en el eje X y en el eje Y y guardar en el grafo únicamente el nuevo desplazamiento con referencia a la celda, por ejemplo, si dos rectángulos de material metal estás distanciados a una magnitud X y existe una regla de diseño la cual implica que DR mayor que X, el algoritmo va a asignar al nodo (polígono actual) del grafo el valor de la regla de diseño DR y el objeto a desplazar.

Algoritmo 6 Grafo de interacciones

```
PARA TODO polígono EN polígonosmaterial HAGA
  SI polígono - polígono_vecino MENOR QUE Ancho_Via ENTONCES
    dr_viol = dr_check(polgono)
    SI dr_viol = True ENTONCES
      guardar_distancia(polígono, polígono_vecino.ancho() + dr)
    FIN SI
  FIN SI
FIN PARA TODO
```

El algoritmo 6 presenta el pseudo código de la primera etapa del algoritmo, el análisis se realiza para ambos ejes (X y Y).

La segunda etapa consiste en actualizar recursivamente los polígonos que se encuentran en el trazado, en esta etapa se utiliza una estructura del tipo mapa con la finalidad de replicar el funcionamiento de un diccionario el cual se indexa por diferentes capas de material. El algoritmo consiste en recorrer cada una de las capas de material y buscar los objetos asociados, para cada uno de estos objetos se analizan las interacciones guardadas en el grafo de restricciones, una vez encontrada una interacción válida, se cambia la posición del elemento en el trazado considerando que la nueva posición se referencia al objeto en la vecindad (punto inicial del objeto en el eje x o y, ancho del polígono y la regla de diseño) si se encuentra una distancia que produce un desplazamiento mayor, este se implementa debido a que el mayor desplazamiento cumple implícitamente otras distancias mínimas, este algoritmo tiene una limitante debido a que posee dirección, el trazado se actualiza de izquierda a derecha y de abajo hacia arriba de manera incremental.

El algoritmo [7] muestra una representación en pseudo código de la segunda etapa del

Algoritmo 7 Re-alineamiento de polígonos

PARA TODO polígono **EN** polígonos(material) **HAGA**
SI derecha(polígono,polígono_vecino) **Ó** debajo(polígono,polígono_vecino) **ENTONCES**
$polgono = polgono\_vecino.valor\_punto() + polgono\_vecino.ancho() + dr$
actualizar_polígono(polígono, polígonos$material$)
FIN SI
FIN PARA TODO

compactador, este algoritmo tiene una desventaja si bien los diseños terminan libres de reglas de diseño, no es una solución fácilmente escalable a diseños de mayor complejidad. Otro problema de este acercamiento es que no realiza mejoras topológicas, por ejemplo, si la topología escogida por el diseñador introduce problemas de congestión y por ende corto circuitos o circuitos abiertos, estos no podrán ser arreglados por la herramienta por lo que experiencia con la misma es requerida.

3.2.3 Validación

Para validar el diseño, se utiliza una biblioteca mínima, las herramientas de Synopsys, en específico Design Compiler, necesita que una biblioteca contengan 5 compuertas como mínimo, un flip flop D, una compuerta nand, una compuerta and, una compuerta nor y una compuerta or debido a que todo el resto de la lógica se puede construir a partir de dichas celdas para validar la implementación de la herramienta.

Con el fin de garantizar la funcionalidad inicial de la herramienta, se utilizan validaciones físicas de Custom Compiler (trazado contra esquemático (LVS) y reglas de diseño (DRC)) , primeramente se requiere que las celdas generadas por la herramienta cumplan la función lógica asociada, por lo que se realiza un esquemático de la celda con el fin corroborar la implementación LVS. Como segundo paso se realizan las pruebas de DRC con el fin de verificar que el diseño no incumple reglas fundamentales de la tecnología y puede ser fabricado finalizando así el ciclo de diseño de la herramienta, y, cómo último paso se realizan las simulaciones adecuadas en conjunto con la extracción de elementos parásitos para visualizar la calidad de los trazados generados.

Una vez validada la implementación básica de la herramienta, las celdas generadas deben ser capaces de ser interpretadas como una biblioteca por la herramienta de PrimeLib de Synopsys con el fin de validar si los resultados obtenidos pueden ser utilizados por las herramientas de síntesis para generar un *netlist* logrando implementar un flujo de diseño digital con bibliotecas de autoría propia. Finalmente, la vista de diseño debe ser generada con la finalidad de completar las vista mínima requerida para una síntesis lógica en otras palabras, los trazados generados deben ser de fácil comprensión para herramientas de caracterización utilizadas en la industria manteniendo así la capacidad de integrar diseños personalizados.

Con el fin de validar la implementación de la biblioteca se compara las características de área, potencia y velocidad de las celdas generadas contra las bibliotecas disponibles en el laboratorio donde se realiza la investigación. Se plantea el uso de 2 bibliotecas como referencia, una con tecnología de 180nm de XFab y la otra una tecnología de 90nm de Synopsys como bibliotecas de referencias con el fin de generar las métricas comparativas contra bibliotecas de calidad industrial, la comparativa es realizada sobre las celdas hoja de la biblioteca, en este caso, celdas AND, NAND, FLOP, NOR y OR. Se debe considerar que la tecnología de 90nm es una producto gratuito y no ha recibido actualizaciones desde el año 2013, por lo que existen problemas de compatibilidad con las versiones más recientes de las herramientas a tal punto de que la extracción de elementos parásitos en dicha tecnología no es posible por lo que la comparación de velocidad (rendimiento) y consumo de energía no es posible realizarla pero si la comparación a nivel de trazados (área).

Chapter 4

Resultados

La metodología propuesta para verificar las celdas generadas por el generador de celdas consisten en validar la implementación de los transistores mediante equivalencias entre la descripción de los transistores y un *netlist* el cual describe los transistores a validar.

Con el fin de realizar dicha validación se utiliza la herramienta de Custom Compiler de Synopsys, la cual permite realizar dicho tipo de validaciones a partir de un *netlist* de HSPICE. Se utilizan 5 celdas para realizar dicho diseño, AND, OR, NAND, NOR e INV, estas celdas se consideran como la unidad mínima para generar una biblioteca compatible con Design Compiler y se utilizan transistores de tamaño mínimo (420 de ancho y 180 de largo). Esta biblioteca se utiliza para validar la funcionalidad de la herramienta.

Para realizar la validación, es necesario crear un archivo compatible con Custom Compiler, este archivo se genera en el paso de la traducción del mallado al trazado físico y consiste en un *script* en el lenguaje TCL, el cual es el lenguaje utilizado por varias herramientas para construcción de ICs, también cabe mencionar que Custom Compiler trabaja en formato ASCII, por lo que es necesario obtener el contexto de diseño utilizando el identificador ASCII del diseño actual.

4.0.1 Generación de los trazados de las celdas

Las celdas utilizan dos vistas principales para su construcción, la vista de esquemático y la vista geométrica, la vista de esquemático se crea utilizando la herramienta de Custom Compiler mientras que la vista geométrica utiliza la herramienta propuesta en esta tesis para su generación. Con el fin de validar las celdas creadas por el generador, se utiliza una metodología que permite comprobar equivalencias entre el *netlist* que posee una descripción de los transistores y sus conexiones y el trazado, en un flujo de diseño, el trazado debe cumplir los con la descripción impuesta por el *netlist* (mismas conexiones, nombres de las interconexiones, nombres de los transistores y dimensiones de los mismos).

Estas validaciones se realizan sobre una biblioteca mínima, esta biblioteca mínima que contiene las siguientes celdas, AND, OR, NAND, NOR y un Flip Flop D.

4.0.2 Biblioteca personalizada generada en la Tecnología 180nm

Resultados Generales

Al comparar la biblioteca generada contra una referencia de calidad industrial. Las celdas hoja escogidas posee un *drive strength* unitario al ser celdas de tamaño mínimo (dentro del contexto de la biblioteca son la variante X0 las más pequeñas) similar al escogido por la biblioteca personalizada ya que el principal objetivo es probar que el generador de celdas puede producir resultados correctos desde construcción y observar la calidad de los algoritmos utilizados.

Para efectos de 180nm el pitch utilizado fue de 4,78nm mientras que las celdas estándar utilizan 5,28nm, el pitch utilizado esta considerado en función de el tamaño de las vías, de acuerdo a la figura 2.2, el acercamiento utilizado es el menos eficiente pero con menor complejidad para implementar la optimización.

A nivel de área los resultados se presentan en la siguiente tabla:

Celda	Personalizada (μm^2)	Industrial (μm^2)	Diferencia (%)
AND21	14,80296	12,544	-18
OR21	11,1	10,0352	-11
NAND21	14,80296	12,544	-18
NOR21	11,1	10,0352	-11
DFLOP	111,438	57,7024	-93

Tabla 4.1: Comparativa de área a nivel de trazado entre la biblioteca generada contra una biblioteca de calidad industrial. Nótese que las celdas generadas por la herramienta consumen mayor área debido a las restricciones de diseño y al acercamiento planteado.

La tabla 4.1, muestra que los valores de área son relativamente cercanos entre los diseños a excepción de la celda flip flop dónde se degrada substancialmente. También se debe considerar que el *pitch* de las celdas personalizadas es más pequeño (aunque por menos de 4 nm). El incremento en área se debe a que el algoritmo no permite trabajar con ángulos que no sean rectos por lo que las interconexiones se componen únicamente de elementos con direcciones verticales y horizontales, lo cual implica que el uso de área es mayor expandiéndose de manera horizontal. En el caso del flip flop, la topología utilizada produjo efectos de congestión lo cual creaba violaciones a las reglas de diseño por lo que se ocupo aumentar la resolución del mallado debido a la necesidad de utiliza la capa de metal 2. Un diseño complejo como lo es un flip flop puede obtener resultados correctos comprometiendo la optimización por ende se observa como el área se incrementa substancialmente.

Energía y rendimiento

Se utiliza un banco de pruebas sencillo con una carga de salida y un generador de señales a la entrada, la carga es estándar para todas las celdas en este caso se utiliza 1 fF debido a que es un valor aproximado al F04 para efectos de ambas bibliotecas (el apéndice A contiene una explicación mas detallada de como se calculó y que es el retardo F04). Todas las pruebas se realizaron utilizando la esquina típica la cual posee como tensión 1.8 V y temperatura de 25° C.

Al comparar la velocidad, se observa una dependencia sobre la topología utilizada en el caso de la celda personalizada se utilizaron transistores del mismo tamaño por lo que la transición no es simétrica, al utilizar transistores de las mismas dimensiones es esperable que tenga mejores transiciones de alto a bajo y en los tiempos de bajada, la razón es que los transistores tipo N son aproximadamente 2 veces más veloces que los tipo P por lo que los transistores simétricos cumplen una relación de tamaños 2 a 1 (el transistor P es dos veces más grande) esta relación es teórica pues en la realidad no siempre se satisface con la relación mencionada.

A nivel de celdas, se busca comparar el rendimiento por lo que hay 4 tiempos que interesan, el tiempo de subida, el tiempo de bajada, el retardo de propagación de alto a bajo y el retardo de propagación de bajo a alto. El tiempo de subida y de bajada se define como el tiempo que dura la transición de la señal de salida alcanzar valor 1 comenzando con un valor de 0 o alcanzar un valor de 0 iniciando con un valor de 1 respectivamente, a su vez el tiempo de transición se define como el tiempo que la señal dura en alcanzar el 90% del valor de la señal comenzando la medición cuando la señal está al 10% o vice versa.

El tiempo de subida se muestra en la siguiente tabla4.2.

Celda	Personalizada ns	Industrial ns	Diferencia (%)
AND21	0,135	0,135	0
NAND21	0,183	0,145	-26
OR21	0,115	0,101	-14
NOR21	0,273	0,3	9
DFLOP	0,166	0,208	20

Tabla 4.2: Comparativa de tiempo de subida entre la biblioteca generada contra una biblioteca de calidad industrial con elementos parásitos. Nótese como los resultados son relativamente cercanos a las celdas de calidad industrial.

También se captura el tiempo de bajada en la siguiente tabla 4.3.

Celda	Personalizada ns	Industrial ns	Diferencia (%)
AND21	0,0738	0,07	-5
NAND21	0,105	0,103	-2
OR21	0,134	0,114	-18
NOR21	0,0781	0,069	-13
DFLOP	0,103	0,205	50

Tabla 4.3: Comparativa de tiempo de bajada entre la biblioteca generada contra una biblioteca de calidad industrial con elementos parásitos. Es de recalcar que los resultados son comparables con las celdas de calidad industrial.

A partir de los resultados vistos en 4.3 y 4.2 se observa que la celda personalizada posee mejores tiempos de bajada que de subida, esto se debe a que las celdas están construidas con un sesgo hacia la transición de alto a bajo (por la similitud en dimensiones de los transistores) al comparar los resultados contra la celda estándar de calidad industrial, podemos observar como los resultados se asemejan, esta similitud implica que una topología simple produce resultados que satisfacen la mayoría de los escenarios aunque la topología no esté optimizada para la tecnología.

Por su parte, el retardo de propagación de alto a bajo y el retardo de propagación de bajo a alto se puede definir como el tiempo que le toma a la celda reflejar a la salida el cambio visto en la entrada, normalmente se define como el tiempo la señal de salida dura en alcanzar el 50% del valor de la señal medido desde el 50% del valor de la señal de entrada, y el tiempo puede diferir de acuerdo al tipo de transición (de bajo a alto o de alto a bajo).

El retardo de propagación de bajo a alto se muestra a continuación en al tabla 4.4.

Celda	Personalizada ns	Industrial ns	Diferencia (%)
AND21	0,179	0,202	11
NAND21	0,095	0,069	-38
OR21	0,138	0,103	-34
NOR21	0,144	0,123	-17
DFLOP	5,425	5,504	1

Tabla 4.4: Comparativa de retardo de propagación bajo a alto entre la biblioteca generada contra una biblioteca de calidad industrial con elementos parásitos. Nótese como la familia de las compuertas AND y NAND son más lentas y cómo las compuertas con una etapa de inversión (AND y OR) poseen un retardo menor.

Por su parte el retardo de propagación de alto a bajo se muestra en la tabla 4.5.

Celda	Personalizada ns	Industrial ns	Diferencia (%)
AND21	0,149	0,144	-3
NAND21	0,059	0,089	34
OR21	0,284	0,233	-22
NOR21	0,051	0,049	-4
DFLOP	5,318	5,785	8

Tabla 4.5: Comparativa de retardo de propagación alto a bajo entre la biblioteca generada contra una biblioteca de calidad industrial con elementos parásitos. Es de recalcar que los tiempos de bajada son más competitivos entre las bibliotecas.

Los resultados de las tablas 4.5 y 4.4 muestran que los diseños poseen retardos relativamente similares y la tendencia es similar en ambos casos, la celda personalizada está más sesgada hacia la transición de alto a bajo como se había mencionado inicialmente. Para efectos del Flip Flop el tiempo de Dato a Reloj es de 5,01ns (mismo banco de pruebas para ambas tecnologías) por ello el tiempo se ve alto.

A nivel de consumo de energía los datos capturados se pueden observar en la tabla 4.6.

Celda	Personalizada μW	Industrial μW	Diferencia (%)
AND21	1,84	2,1	12
NAND2	1,42	1,55	8
OR211	1,86	2,21	16
NOR21	1,4	1,68	17
DFLOP	8,11	10,08	20

Tabla 4.6: Comparativa de consumo de energía a nivel entre la biblioteca generada contra una biblioteca de calidad industrial a nivel de trazado con extracción de elementos parásitos en una tecnología de 180nm. Obsérvese como el consumo de energía es comparable entre la biblioteca personalizada y la referencia de calidad industrial.

Mediante la tabla 4.6 se puede ver como el consumo energético permanece en el mismo rango de valores, teniendo la celda personalizada una mejora marginal en la familia de las compuertas AND, NAND, NOR, OR y el Flop D este resultado se debe a que las celdas son de un canal más pequeño lo cual implica que la corriente del canal debe ser menor y por ende se observa menor consumo de potencia.

Hay que considerar que para hacer la comparación se utilizó la celda mínima perteneciente a las celdas estándar, la cual posee una relación de 2 a 1, también que la herramienta se configuró para utilizar tamaños simétricos por lo que se posee una relación de 1 a 1 a la hora de optimizar el tamaño del canal de los diferentes transistores, por lo que diferencias en el diseño son esperables. La razón de por que los transistores se diseñaron de tal

forma que posean las mismas dimensiones es para facilidad de implementar el generador de celdas, por lo que mejorar la topología y los algoritmos para obtener resultados de mayor calidad es parte del trabajo futuro.

Para validar los resultados se realiza una simulación funcional después de haber extraído elementos parásitos del trazado con el fin de validar el funcionamiento lógico y generar una biblioteca personalizada.

4.0.3 Simulación con elementos parásitos AND

Con el fin de realizar una simulación de mayor calidad se utiliza un *netlist* de SPICE con elementos parásitos incluidos.

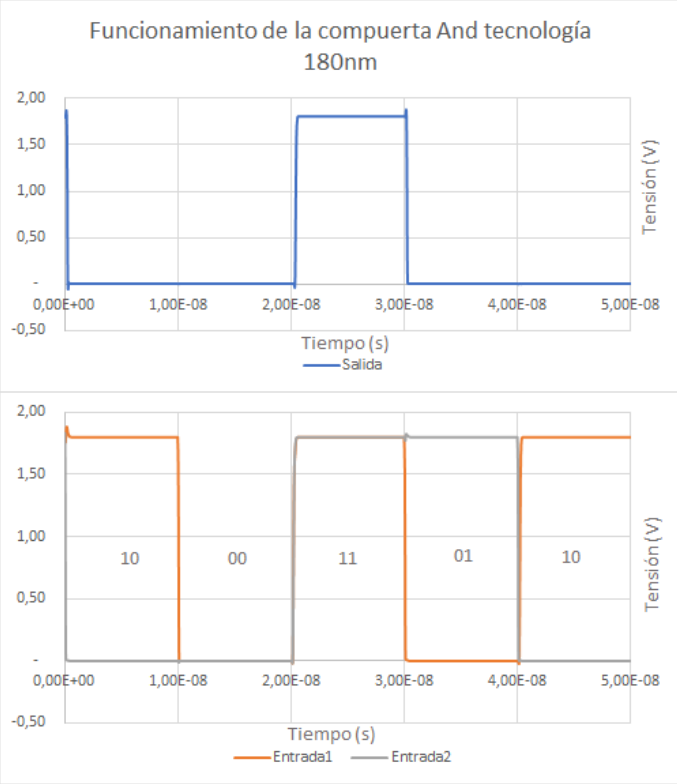


Figure 4.1: Validación del comportamiento lógico mediante una simulación de trazado completo con extracción de elementos parásitos en una tecnología de 180nm para la compuerta AND. Nótese como existe un punto de resonancia a los 30ns.

La figura 4.1 muestra los resultados de la simulación de la compuerta and la cual cumple la función lógica implementada. A los 30ns se observa un punto de resonancia en donde la tensión sube por encima de la fuente, esto se debe a los polos y ceros que existen debido a las cargas parásitas y la resistencia del canal de los transistores, por lo que el dimensionamiento juega un papel importante a la hora de implementar la plantillad e diseño.

4.0.4 Simulación con elementos parásitos NAND

Se presentan los resultados de simulación para la compuerta nand.

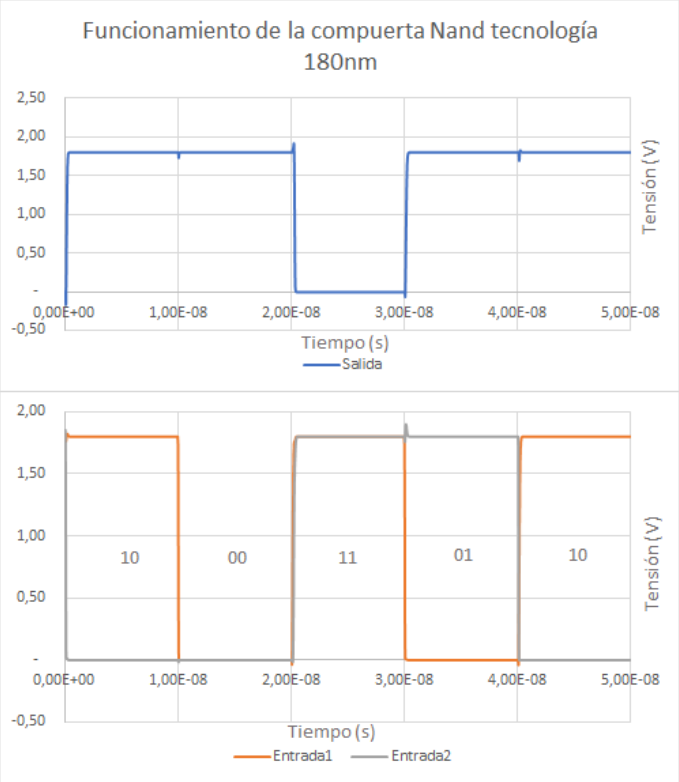


Figure 4.2: Validación del comportamiento lógico mediante una simulación de trazado completo con extracción de elementos parásitos en una tecnología de 180nm para la compuerta NAND. Obsérvese como a los 40ns existe una transición indeseada.

Como se observa en la figura 4.2 la simulación cumple el funcionamiento lógico deseado y de igual manera que la compuerta and existe un *glitch* cuando una de las entradas transicionan de alto a bajo y la otra entrada se encuentra en un estado alto, en el caso de la compuerta NAND implementada, el *glitch* no incurre en fallo funcional debido a que es una transición muy pequeña por lo que se enmascara eléctricamente (transición no va a incurrir en un cambio de estado de la lógica en cascada). Con el fin de mitigar estos efectos es requerido un dimensionamiento de los transistores que permitan minimizar los efectos debidos a la contención de los nodos internos durante las transiciones mencionadas con anterioridad.

4.0.5 Simulación con elementos parásitos OR

Para la celda OR, la simulación utiliza un trazado con elementos parásitos extraídos.

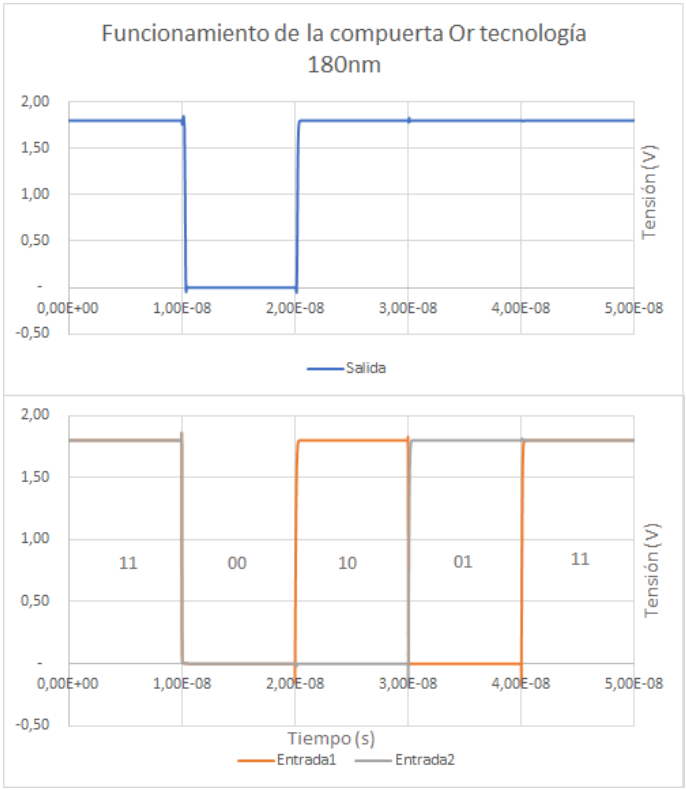


Figure 4.3: Validación del comportamiento lógico mediante una simulación de trazado completo con extracción de elementos parásitos en una tecnología de 180nm para la compuerta OR. Nótese como la celda posee puntos resonancia durante las transiciones.

La figura 4.3 muestra el resultado de la simulación funcional se observa que el comportamiento es el esperado pero también existe un punto en la transición dónde la señal resuena debido a las capacitancias parásitas lo cual hace que la carga de la compuerta aumente reflejándose como un aumento y disminución de la tensión por encima del la tensión de alimentación o de la tensión de referencia, con la finalidad de disminuir este efecto, una mejora el diseño definido en la plantilla la cual dimensione los transistores con el fin de suavizar el punto de resonancia.

4.0.6 Simulación con elementos parásitos NOR

El funcionamiento de la compuerta nor se realiza mediante el uso de una simulación de trazado completo con elementos parásitos.

La figura 4.4 muestra como la implementación cumple el funcionamiento lógico correcto para el diseño implementado, también se observa puntos de resonancia similar a la com-

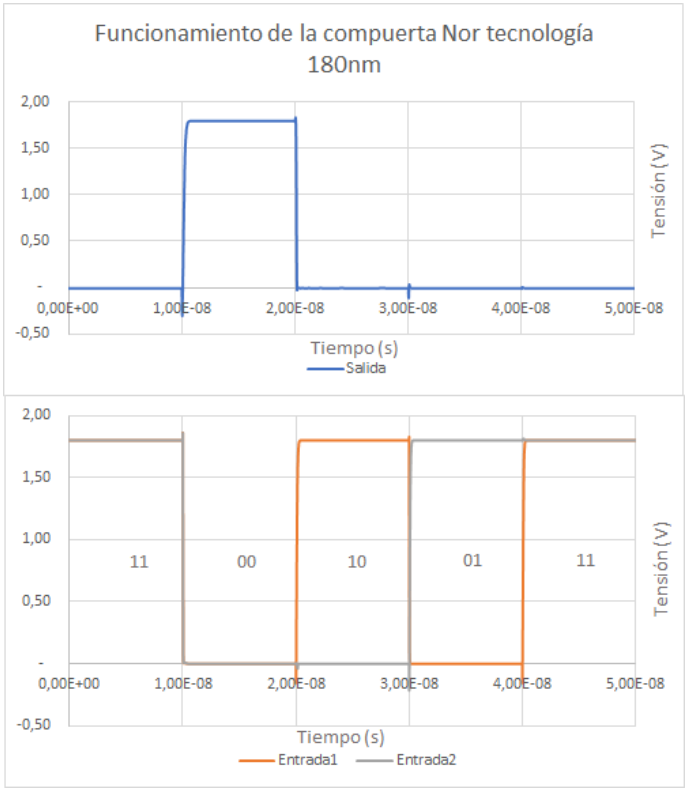


Figure 4.4: Validación del comportamiento lógico mediante una simulación de trazado completo con extracción de elementos parásitos en una tecnología de 180nm para la compuerta NOR. Obsérvese como las transiciones poseen resonancia similar a la compuerta OR.

puerta or, lo cual implica que los diseños deben dimensionarse de manera adecuada con la finalidad de generar una implementación de alta calidad.

4.0.7 Simulación con elementos parásitos FLOP

En el caso del flop se implementa una celda tipo D con reinicio asíncrono activo en 0.

La figura 4.5 muestra el comportamiento de la celda implementada, se observa que cuando la señal de reloj (naranja) transiciona de bajo a alto el flip flop captura y propaga el dato hacia la salida por lo que es un flip flop de flanco positivo, también se observa que posee una señal de reinicio asíncrono activo en bajo, pues si la señal de reinicio (naranja) está en alto, se permite el paso entre la entrada y la salida la topología de este flop coincide con el comportamiento de un flip flop tipo D validando la implementación realizada.

Una vez validada la implementación de las celdas, se procede a caracterizar la biblioteca mediante la herramienta herramienta PrimeLib de Synopsys, este proceso caracteriza la celda en múltiples esquinas generando tablas de transición por cada una de las condiciones de operación, por lo que es un análisis más exhaustivo que la simulación con elementos parásitos ya que la simulación se realiza en la esquina típica cuya tensión es 1.8 V y una

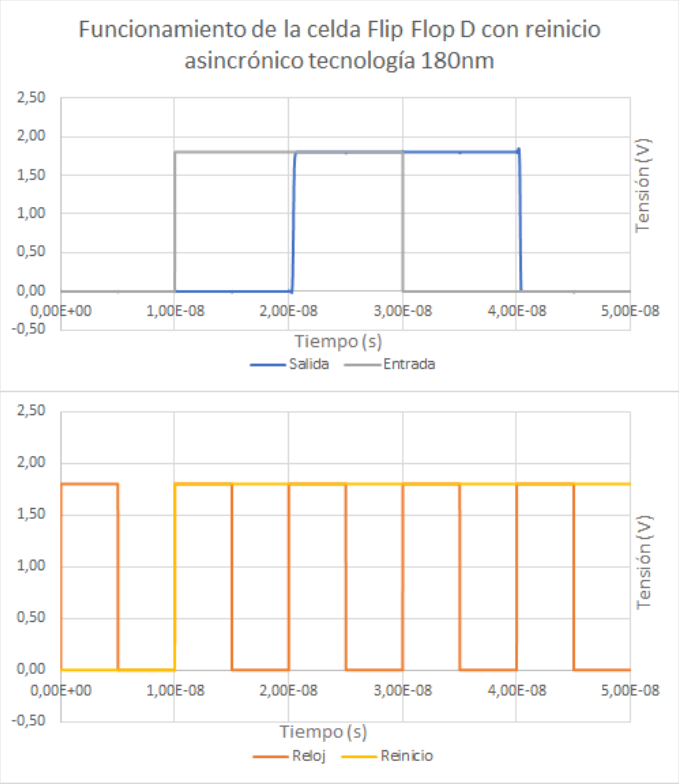


Figure 4.5: Validación del comportamiento secuencial mediante una simulación de trazado completo con extracción de elementos parásitos en una tecnología de 180nm para la celda FLOP. Nótese que el comportamiento del flop coincide con un flop tipo D.

temperatura de 25° C.

Seguidamente se realiza un proceso de síntesis lógica utilizando la biblioteca personalizada. El diseño a implementar es un contador de 4 bits con el fin de utilizar todas las celdas de la biblioteca y validar si la biblioteca generada es sintetizable, en general, el interés está en corroborar la validez de la biblioteca y no comparar los resultados obtenidos debido a que la biblioteca no está lista para fabricación pero la intención es generar diseños listos para fabricación en iteraciones futuras de la herramienta.

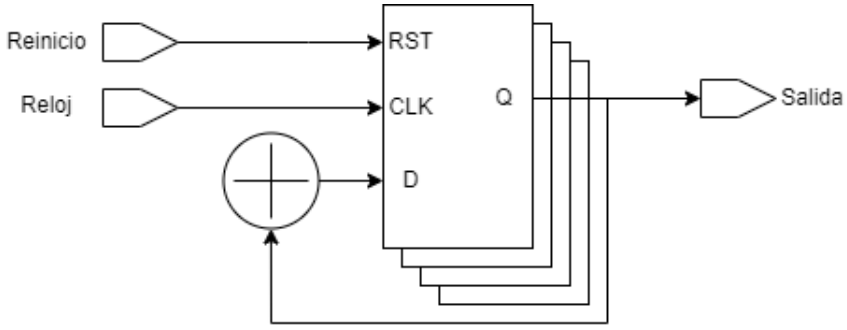


Figure 4.6: Contador de cuatro bits utilizado para validar la biblioteca durante el proceso de síntesis. Nótese que el diseño instancia varias celdas secuenciales y deja a libertad del sintetizador implementar la lógica combinacional.

La figura 4.6 muestra el diseño del contador a implementar, se utiliza una descripción funcional del circuito con la finalidad de dar libertad a la herramienta de síntesis con intención de que se utilicen diferentes combinaciones de lógica combinacional para confirmar si la biblioteca posee diferentes celdas combinacionales cuyo funcionamiento puede ser inferido por la herramienta compilación durante el proceso de síntesis.

```
module counter ( clk, rstn, out );
  output [3:0] out;
  input clk, rstn;
  wire   N7, N8, N9, N10, n16, n17, n18, n19, n20, n21, n22, n23, n24, n25;

  dflop_async_rst \out_reg[0]  ( .d(N7), .clk(clk), .rstb(1'b1), .q(out[0]) );
  dflop_async_rst \out_reg[1]  ( .d(N8), .clk(clk), .rstb(1'b1), .q(out[1]) );
  dflop_async_rst \out_reg[2]  ( .d(N9), .clk(clk), .rstb(1'b1), .q(out[2]) );
  dflop_async_rst \out_reg[3]  ( .d(N10), .clk(clk), .rstb(1'b1), .q(out[3])
        );
  inv U22 ( .v2_implicit(rstn), .o(n25) );
  nor21 U23 ( .v2_implicit(n25), .v3_implicit(out[0]), .o(N7) );
  nor21 U24 ( .v2_implicit(out[1]), .v3_implicit(out[0]), .o(n17) );
  and21 U25 ( .v2_implicit(out[0]), .v3_implicit(out[1]), .o(n18) );
  or21 U26 ( .v2_implicit(n18), .v3_implicit(n25), .o(n16) );
  nor21 U27 ( .v2_implicit(n17), .v3_implicit(n16), .o(N8) );
  nor21 U28 ( .v2_implicit(n18), .v3_implicit(out[2]), .o(n20) );
  nand21 U29 ( .v2_implicit(out[2]), .v3_implicit(n18), .o(n21) );
  nand21 U30 ( .v2_implicit(n21), .v3_implicit(rstn), .o(n19) );
  nor21 U31 ( .v2_implicit(n20), .v3_implicit(n19), .o(N9) );
  or21 U32 ( .v2_implicit(n21), .v3_implicit(out[3]), .o(n23) );
  nand21 U33 ( .v2_implicit(out[3]), .v3_implicit(n21), .o(n22) );
  and21 U34 ( .v2_implicit(n23), .v3_implicit(n22), .o(n24) );
  nor21 U35 ( .v2_implicit(n25), .v3_implicit(n24), .o(N10) );
endmodule
```

Figure 4.7: Netlist resultante de la síntesis utilizando la biblioteca personalizada en una tecnología de 180nm. Nótese que la herramienta utiliza diferentes celdas combinacionales generadas para implementar el circuito que aumenta la cuenta de los registros.

La figura 4.7 muestra que la biblioteca utilizada es la biblioteca personalizada generada por la herramienta y que la herramienta de síntesis es capaz de reconocer y utilizar los diferentes diseños provistos por los archivos compilados en la biblioteca, lo cual valida la implementación debido a que es posible integrar los resultados del generador de celdas a un flujo de diseño de circuitos integrados.

4.0.8 Trazados en la tecnología de 180nm

Los trazados generados por la herramienta comprenden diferentes celdas los cuales se generan libres de DRCs las cuales son necesarias para garantizar un mínimo que se considere como una biblioteca, los trazados generados por la herramienta comprenden el grupo mínimo especificado.

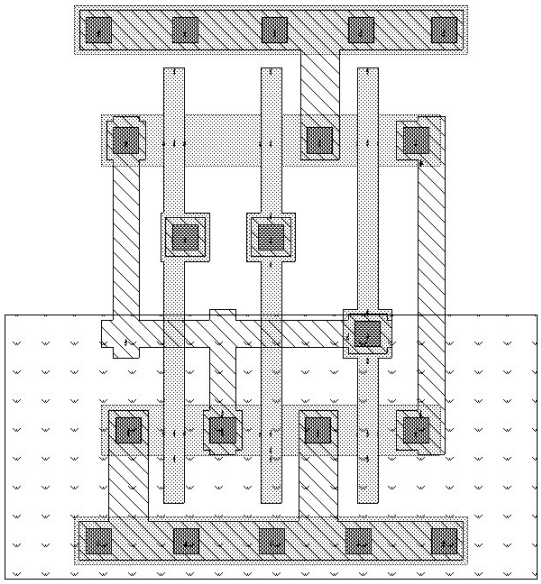


Figure 4.8: Trazado completo de una compuerta AND con extracción de elementos parásitos en una tecnología de 180nm generado mediante el generador de celdas. Nótese que existen forma no uniformes cuando las interconexiones cambian de vertical a horizontal.

Celda AND

La figura 4.8, muestra el trazado completo obtenido por el generador de celdas. La técnica utilizada para realizar las plantillas se basa en redes de *pull-up* con transistores PMOS y redes *pul-down* con transistores NMOS debido a la simplicidad del diseño que producen. Otro efecto que se observa son formas no uniformes conocidas como *notches*, estos se generan cuando la herramienta cambia de dirección de durante el proceso de generar las interconexiones. Estas formas no uniformes se generan debido a que el proceso de traducción al mundo físico se realiza de manera independiente para los polígonos que se desplaza de manera vertical y para los que se desplazan de manera horizontal, implementar una función que permita suavizar los *notches* es una mejora que debe implementarse al generador de celdas con el fin de mejorar la calidad del diseño auqnue no represente un riesgo funcional.

Celda OR

La figura 4.9 muestra el trazado que el generador produjo. El trazado consta de una fila conectada en serie de transistores tipo P y una fila de transistores tipo N conectados en paralelo, Los transistores tipo N y tipo P se encuentran alineados de acuerdo al modelado del mallado para facilitar la generación del trazado. Otro aspecto del trazado a discutir es que las interconexiones están implementadas en tamaño mínimo debido a la complejidad de implementar las interconexiones internas de la celda, lo cual puede impactar la calidad

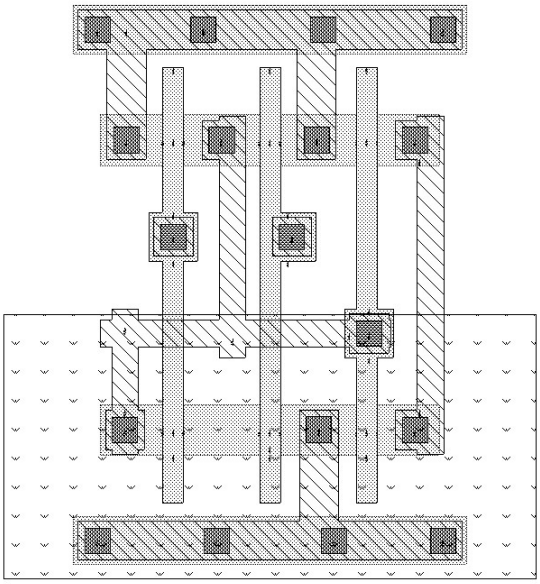


Figure 4.9: Trazado completo de una compuerta OR con extracción de elementos parásitos en una tecnología de 180nm generado mediante el generador de celdas. Nótese que las interconexiones de metal están dimensionadas en tamaño mínimo.

del diseño, es un aspecto de mejora a futuro implementar la capacidad de cambiar el tamaño de las interconexiones de acuerdo a entradas que el usuario decida implementar.

Celda NAND

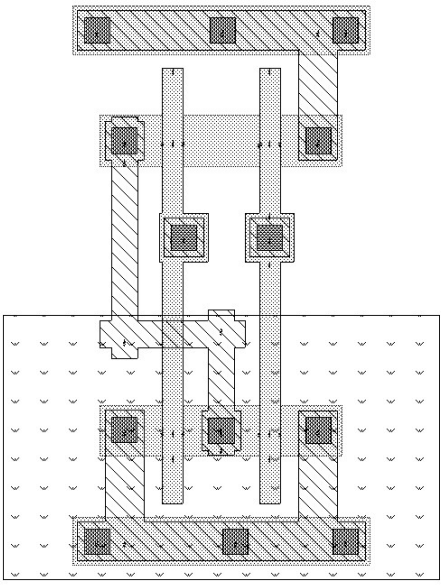


Figure 4.10: Trazado completo de una compuerta NAND con extracción de elementos parásitos en una tecnología de 180nm generado mediante el generador de celdas. Nótese las vías están implementadas de manera uniforme a lo largo del diseño.

La figura 4.10 muestra el trazado de la compuerta NAND creado por la herramienta, el proceso de construcción de la compuerta NAND es similar al de la compuerta AND, la principal diferencia es que se remueve la creación del inversor de salida, lo cual implica un paso menos a la hora de realizar las interconexiones de la compuerta. Para implementar las vías con el fin de interconectar diferentes capas de metal, fue necesario utilizar un acercamiento en el cual un objeto vía es definido, este objeto se optimiza de manera independiente de acuerdo las capas de material pero se desplaza como un elemento único, por ende los diferentes polígonos que componen las vías siempre se encontraran alineados al rededor de un punto central.

Celda NOR

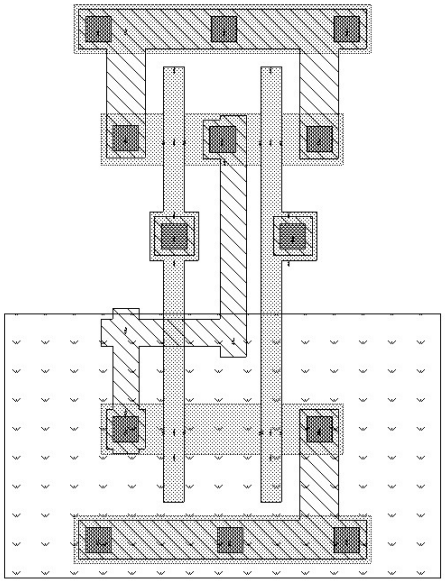


Figure 4.11: Trazado completo de una compuerta NOR con extracción de elementos parásitos en una tecnología de 180nm generado mediante el generador de celdas. Nótese como el algoritmo desplaza los diferentes elementos del diseño de izquierda a derecha.

La metodología para generar la celda NOR vista en la figura 4.11 se asemeja a la utilizada para crear la compuerta OR y, de igual forma, que lo visto en la celda NAND, utiliza los mismos pasos exceptuando la generación del inversor de salida. Como se mencionó en la el capítulo 3, el algoritmo de optimización posee desplazamientos preferidos que van a abajo hacia arriba y de izquierda a derecha como se observa en el trazado final.

Celda FLOP

Un diseño más complejo son los flip flops, estas celdas constituyen agrupamientos de mayor tamaño por lo que la complejidad de la generación de celdas aumenta. El resultado final de la topología escogida es el siguiente:

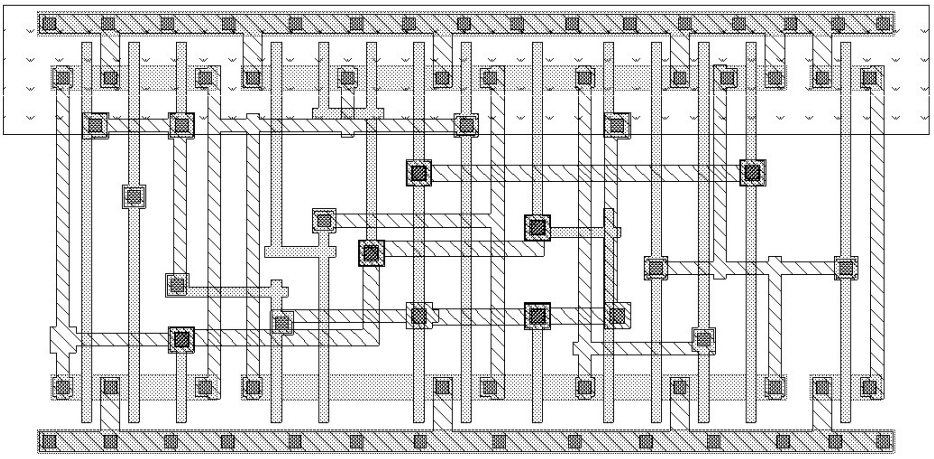


Figure 4.12: Trazado completo de una compuerta FLOP con extracción de elementos parásitos en una tecnología de 180nm generado mediante el generador de celdas. Nótese como se utilizaron diferentes capas de material para realizar las interconexiones en el canal de la celda.

Al implementar la celda 4.12 se utiliza una topología la cual requiere rompimiento de difusión y uso de metales superiores (metal 2) si bien no es deseable, las limitaciones de la herramienta implicaron utilizar dicha topología y la razón es que el algoritmo no permite trabajar con ángulos que no sean rectos por lo que las guías de enrutamiento solo poseen dos direcciones, vertical y horizontal. También se debe recalcar que se utilizaron diferentes capas de material para realizar las interconexiones, desde polisilicio hasta metal 2, lo cual muestra la flexibilidad que permite la herramienta al utilizar diferentes capas de material, aunque la principal limitación es que el usuario debe especificar en cual capa de material se realizan las interconexiones como parte de la configuración inicial (debido a que es un acercamiento mixto entre procedimiento y optimizador).

Al implementar la celda se encontró un problema con la construcción de las líneas de metal, si se coloca una vía y esta interrumpe el camino predeterminado, se genera un abierto.

Este problema se observa en la figura 4.13, la solución que se determinó correcta es dividir la interconexión en dos partes conectando siempre de vía a vía, limpiando así el abierto generado por la vía.

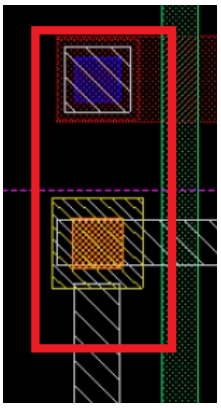


Figure 4.13: Fallo en interconexión o abierto debido a la interacción de reglas de diseño asociadas a vías e interconexiones durante el proceso de enrutamiento. Obsérvese como la vía impide el paso entre la interconexión vertical y la vía.

4.0.9 Biblioteca personalizada en la tecnología 90nm

Al moverse a la tecnología de 90nm encontramos un problema y es que no es posible realizar la extracción, la versión más actualizada de StarRC (herramienta de extracción) tiene un problema de compatibilidad con el archivo de configuración causando que el proceso de extracción no pueda concluir correctamente, por lo que únicamente se realizó la validación a nivel de trazado (comparación de área) y simulaciones funcionales a nivel de esquemático para validar el funcionamiento de las celdas ya que si la validación de esquemático determina que el trazado y el esquemático son equivalentes, funcionalmente no existe ningún problema.

Los resultados comparativos a nivel de área se presentan a continuación:

Celda	Personalizada μm^2	Industrial μm^2	Diferencia (%)
AND21	7,32	7,6672	5
OR21	5,429	5,7504	6
NAND21	7,32	7,6672	5
NOR21	5,429	6,7088	19
DFLOP	55,414	25,8768	-114

Tabla 4.7: Comparación a nivel de área entre el trazado realizado por la herramienta generadora de celdas y la referencia de calidad industrial para una tecnología de 90nm. Nótese como la diferencia en área es menor en la tecnología de 90nm debido a que las reglas de diseño no escalan de manera lineal.

En la tabla 4.7, se observa como a nivel de área la topología utilizada escala de manera similar a las celdas de tamaño mínimo de la tecnología a excepción del flop debido a que (similar a 180nm) se tuvo que aumentar el tamaño de la resolución y del *pitch* con la finalidad de lograr un diseño libre de violaciones a las reglas de diseño lo cuál es esperable que posea un área de menor calidad. También se debe mencionar que al comparar los

resultados obtenidos y los mostrados en la tabla 4.1, la tecnología de 90nm posee resultados más cercanos a la biblioteca de referencia, esto se debe a que las reglas de diseño no escala de manera lineal y las topologías implementadas por la biblioteca de 90nm no es la misma que la utilizada por la tecnología de 180nm pero en el caso de los diseños generados mediante la herramienta si utilizan la misma plantilla.

4.0.10 Trazados en la tecnología 90nm

Con el fin de validar la herramienta se utiliza una nueva tecnología ya que una de las principales características que la herramienta debe poseer es la capacidad de escalar hacia diferentes nodos tecnológicos, se utiliza 90nm ya que es una tecnología que posee acceso de licencia de investigación por parte de la compañía Synopsys.

Celda AND

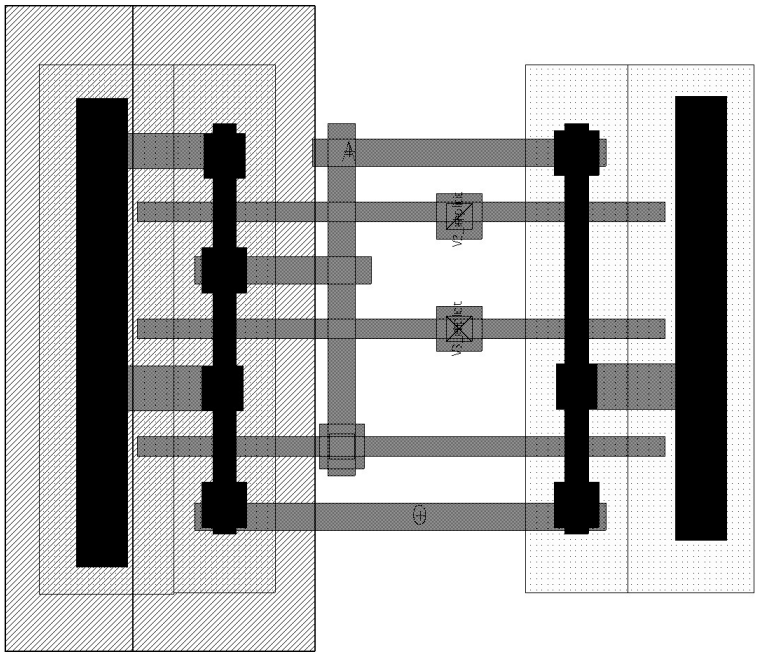


Figure 4.14: Trazado completo de una compuerta AND en una tecnología de 90nm generado mediante el generador de celdas. Nótese como existe un mayor espacio en el canal de la celda al comparar el trazado obtenido con la tecnología de 180nm.

La figura 4.14, muestra como la herramienta es capaz de escalar la topología en una tecnología diferente manteniendo la integridad de la implementación. También se observa que al comparar el trazado resultante con el visto en la tecnología de 180nm 4.8, el espacio que existe en el canal es mayor, una implicación directa es que las topologías pueden presentar resultados óptimos en una tecnología pero no así en otra.

Celda FLOP

El flip flop representa un desafío un más alto al ser una celda más compleja por lo que el resultado también presenta un amento en el consumo de área similar al resultado visto en la tecnología de 180nm.

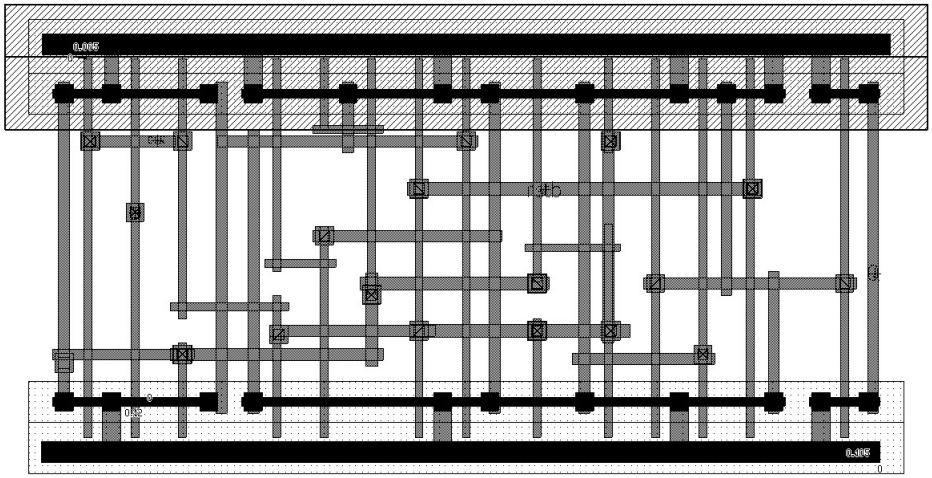


Figure 4.15: Trazado completo de una compuerta FLOP en una tecnología de 90nm generado mediante el generador de celdas. Cabe notar que comparando con el trazado obtenido en 180nm, la topología presenta menor densidad de interconexión en el canal.

La figura 4.15 presenta la celda completa escalada en la tecnología de 90nm. Se oberva que la densidad del canal es menor que el resultado obtenido en la figura 4.12. Si bien el resultado es un trazado completo y libre de reglas de diseño, se presentaron desafíos los cuales son necesarios de mejorar en revisiones futuras de la herramienta. Los desafíos mas comunes que se encontrar fueron abiertos (similar a 180nm) debido a las vías y si interacción con las interconexiones durante el proceso de enrutamiento.

Otro tipo de problema fue la generación de abiertos durante el proceso de optimización debido al movimiento de las líneas de metal cuando interactúan con otros elementos.

La figura 4.16, este error ocurre por interacciones de reglas de diseño vistas en 90nm, la forma de solucionar este problema es revisando la topología y realizando los cambios adecuados, los cambios implican cambiar de posición los elementos que pueden propiciar desplazar de manera inesperada los polígonos de metal.

Finalmente, un problema que se observó únicamente en la tecnología de 90nm son desplazamientos propician que las vías se desplazan de manera inesperada.

Este desplazamiento no fue visto en la tecnología de y se debe al tamaño de las interconexiones y la resolución del mallado utilizada, la forma de arreglar el problema es comprender que causa el desplazamiento y adecuar el tamaño de los polígonos desde la plantilla de configuración, este problema ejemplifica de como una topología que funciona correcta-

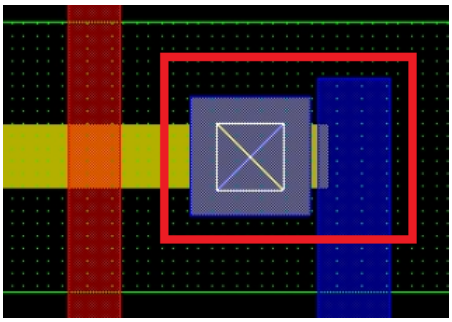


Figure 4.16: Espaciamiento no deseado debido a la interacción de las reglas de diseño durante el proceso de optimización y limpieza de las reglas de diseño. Nótese como la interconexión no se traslapa con la vía y se desplazó hasta no establecer contacto directo produciendo un abierto.

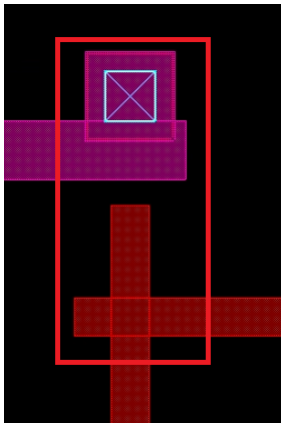


Figure 4.17: Problemas de alineación debido a interacciones de reglas de diseño durante el proceso de optimización y limpieza de las reglas de diseño. Cabe resaltar como la vía se desplazó alejándose de la interconexión causando un abierto.

mente en una tecnología puede tener problemas de escalado a tecnologías más pequeñas debido a las implementaciones de los algoritmos o a las complejidades de las reglas de diseño.

El resto de los trazados para la tecnología de 90nm se pueden observar en el apéndice B.

4.0.11 LVS: Esquemático contra trazado

El proceso de validación de los trazados consiste en verificar si existe una equivalencia adecuada entre el trazado y el esquemático, esto se realiza mediante una comprobación LVS *layout vs schematic*. La tabla 4.8 muestra los resultados a la hora de ejecutar la comprobación de esquemático contra trazado, los resultados para todas las celdas son positivos, lo cual índica que el generador de celdas cumple los requisitos de manera adecuada a la hora de interconectar los transistores y de generar las celdas, estos resultados son correctos en la tecnología de 90nm lo cuál indica que nuestros diseños están correctos (validan la implementación de la topología y la escalabilidad de la herramienta en

Celda	Resultado LVS
AND	Equivalente
OR	Equivalente
NAND	Equivalente
NOR	Equivalente
DFLOP	Equivalente

Tabla 4.8: Resultados de validación LVS la cual compara el trazado utilizado contra el esquemático de spice. Cabe resaltar que una validación LVS garantiza la integridad del diseño pero no su funcionamiento.

diferentes tecnologías).

4.0.12 DRC: Validación de reglas de diseño

La validación de las reglas de diseño *design rule check* consiste en validar las relgas de diseño implementadas, si bien la herramienta considera reglas de diseño para poder validar si la herramienta cumple con los requisitos, se ocupa validar los resultados para efectos de este proyecto.

Celda	Resultado DRC
AND	Limpio
OR	Limpio
NAND	Limpio
NOR	Limpio
DFLOP	Limpio

Tabla 4.9: Resultados de las celdas implementadas al validar si existen violaciones a las reglas de diseño utilizando un analizador de reglas de diseño. Nótese como los resultados están limpios implicando que la herramienta es capaz de producir diseños correctos desde construcción.

La tabla 4.9 muestra los resultados obtenidos por la celdas estándar debido a que no se presentaron problemas de reglas de diseño debido a las distancias durante el proceso del enrutamiento. Introducir una etapa de optimización durante el enrutamiento para un acercamiento de mixto como el planteado por esta tesis, ayuda significativamente a generar diseños libres de violaciones a las reglas de diseño.

Chapter 5

Conclusiones y trabajo futuro

Utilizando un acercamiento híbrido entre la metodología de abajo hacia arriba y de arriba hacia abajo, se comprobó que es posible obtener resultados comparables a una biblioteca de calidad industrial en términos de velocidad y potencia como se observa en la tabla 4.6 cuyas diferencias rondan el 20% en favor del diseño generado por la herramienta. Se destaca que al generar varios trazados y escalando los diseños a diferentes tecnologías la herramienta muestra flexibilidad de implementación ya que los resultados muestran diferencias de área menores al 20% em ambas tecnologías al compararse contra la biblioteca de referencia para cada respectiva tecnología. Cabe resaltar que los resultados reflejan la posibilidad de mejora en los algoritmos utilizados para interconectar los diferentes elementos del trazado, ya hay diferencias mayores a un 30% al comparar la velocidad con respecto a un diseño totalmente personalizado, aunque también es necesario valorar el compromiso que existe entre complejidad de implementación y calidad de resultados.

El propósito detrás de aumentar la flexibilidad a la hora de implementar el diseño mediante una plantilla y de utilizar algoritmos de optimización topológica conscientes de las reglas de diseño se plasma a la hora de generar la celda secuencial Flip Flop D, esta celda presentó desafíos de implementación los cuales se atacaron directamente sobre la plantilla y la capacidad de ejecutar código procedimental mediante los métodos implementados por la herramienta los cuales permiten manipular con una alta granularidad la topología de la interconexión a implementar permitiendo generar un diseño libre de violaciones a las reglas de diseño aunque comprometiendo el tamaño del trazado resultante en un 93% para 180nm y 93% para 90nm.

La introducción de las reglas de diseño en el proceso de optimización del trazado permite reducir la dependencia que existe en la experiencia para generar diseños funcionalmente válidos especialmente por que los diseños que se implementaron partieron de un diagrama de palitos hacia una descripción relativamente sencilla la cual requiere la especificación de los diferentes elementos que componen el trazado mediante puntos en una malla los cuales se introdujeron como entradas al enrutador para generar los diferentes diseños por tanto que los resultados observados presentan un panorama positivo sobre el acercamiento utilizado y motiva a pensar que, en el largo plazo, se puede automatizar el proceso de gen-

eración de bibliotecas produciendo diseños de calidad industrial reduciendo la interacción humana.

5.0.1 Trabajo futuro

1. Mejoras al enrutador.

    - Con el fin de desarrollar una herramienta que genere celdas estándar de mayor calidad, es necesario introducir mejoras al algoritmo utilizado para generar las guías de enrutamiento debido a que actualmente la etapa de aserciones ocupa una definición en el mallado que imponga bloqueos en algunos nodos con el fin de dirigir al enrutador hacia el punto final, por lo que no es totalmente automático.
    - Aumentar la granularidad con la que el usuario controla las especificaciones de diseño debido a que en este punto, las restricciones de entrada se aplican de manera global para las diferentes capas de material y no a interconexiones específicas, esta mejora introduce complejidades las cuales no pudieron se abarcadas durante el desarrollo de la herramienta.
    - Permitir la manipulación de ángulos diferentes a noventa grados, debido a la facilidad de implementar polígonos cuyo desplazamiento ocurren únicamente de manera horizontal o vertical, se comprometió la optimización de algunos diseños aumentando la cantidad de área requerida para en comparación con las celdas de calidad industrial.

2. Rendimiento de la herramienta.

    - Mejorar el rendimiento de la herramienta, se podría plantear el uso de la paralelización paralelización como método de mejora en procesadores multinúcleo con la finalidad de manejar diseños mas complejos sin comprometer la calidad de los resultados.

3. Introducir nuevas capacidades a la herramienta.

    - Introducir capacidades novedosas para analizar y mejorar la topología de los resultados mediante un proceso de estimación del impacto de las reglas de diseño en una etapa temprana antes de la implementación del trazado con el fin de aumentar la calidad de la implementación y facilitar el proceso de fabricación.

Bibliografía

[1] C. J. Alpert, D. P. Mehta, and S. S. Sapatnekar, *Handbook of Algorithms for Physical Design Automation*, 1st. USA: Auerbach Publications, 2008, ISBN: 0849372429.

[2] N. Baha, M. Beddiaf, and A. .-. Gadiri, "Galsy, an automatic layout generator of symbolic layouts from mos circuit schematics," in *[1991] Proceedings. First Great Lakes Symposium on VLSI*, Mar. 1991, pp. 296–300. DOI: 10.1109/GLSV.1991.143982.

[3] J. Cortadella, J. Petit, S. Gómez, and F. Moll, "A boolean rule-based approach for manufacturability-aware cell routing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 33, no. 3, pp. 409–422, 2014. DOI: 10.1109/TCAD.2013.2292514.

[4] F. CURATELLI, D. D. CAVIGLIA, G. M. BlSIO, and M. CHIRICO, "Implementation of efficient strategies for cell generation in vlsi design," *International Journal of Electronics*, vol. 73, no. 6, pp. 1285–1299, 1992. DOI: 10.1080/00207219208925800. eprint: https://doi.org/10.1080/00207219208925800. [Online]. Available: https://doi.org/10.1080/00207219208925800.

[5] M. Daniel, "Layout automation in analog ic design with formalized and nonformalized expert knowledge," Ph.D. dissertation, University of Stuttgart, Stuttgart, 2018. [Online]. Available: https://elib.uni-stuttgart.de/handle/11682/10248?locale=en.

[6] K. Doerffer, A. T. Teby, O. Anton, and D. A. Mlynski, "Kagen-a generator of static cmos-cell layout from circuit schematics," in *1993 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 1993, 1845–1848 vol.3. DOI: 10.1109/ISCAS.1993.394106.

[7] R. S. Ghaida, T. Sahu, P. Kulkarni, and P. Gupta, "A methodology for the early exploration of design rules for multiple-patterning technologies," in *2012 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Nov. 2012, pp. 50–56.

[8] K. Golshan, *Physical Design Essentials: An ASIC Design Implementation Perspective*. Berlin, Heidelberg: Springer-Verlag, 2007, ISBN: 0387366423.

[9] D. F. Gomez Prado, "Tutorial on fpga routing," *Electrónica - UNMSM*, no. 17, pp. 23–33, Jun. 2006. [Online]. Available: https://revistasinvestigacion.unmsm.edu.pe/index.php/electron/article/view/4510.

[10] Hong Li and Wei Li, "Aisce: A layout synthesis system for asic design," in *China., 1991 International Conference on Circuits and Systems*, Jun. 1991, 419–422 vol.1. DOI: 10.1109/CICCAS.1991.184377.

[11] S. Hougardy, T. Nieberg, and J. Schneider, "Bonncell: Automatic layout of leaf cells," in *2013 18th Asia and South Pacific Design Automation Conference (ASP-DAC)*, Jan. 2013, pp. 453–460. DOI: 10.1109/ASPDAC.2013.6509638.

[12] S.-W. Hur, A. Jagannathan, and J. Lillis, "Timing-driven maze routing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 19, no. 2, pp. 234–241, 2000. DOI: 10.1109/43.828552.

[13] C. .-. Hwang, Y. .-. Hsieh, Y. .-. Lin, and Y. .-. Hsu, "A fast transistor-chaining algorithm for cmos cell layout," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 9, no. 7, pp. 781–786, Jul. 1990, ISSN: 1937-4151. DOI: 10.1109/43.55207.

[14] C. Jia and S. Ganbing, "New pcell based ring oscillator layout auto-generation method and application in advanced spice model verification," in *2015 China Semiconductor Technology International Conference*, 2015, pp. 1–3. DOI: 10.1109/CSTIC.2015.7153337.

[15] K. Jo, S. Ahn, T. Kim, and K. Choi, "Cohesive techniques for cell layout optimization supporting 2d metal-1 routing completion," in *2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC)*, Jan. 2018, pp. 500–506. DOI: 10.1109/ASPDAC.2018.8297373.

[16] L. Lavagno, G. Martin, and L. Scheffer, *Electronic Design Automation for Integrated Circuits Handbook - 2 Volume Set*. USA: CRC Press, Inc., 2006, ISBN: 0849330963.

[17] L. W. Liebmann and R. O. Topaloglu, "Design and technology co-optimization near single-digit nodes," in *2014 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Nov. 2014, pp. 582–585. DOI: 10.1109/ICCAD.2014.7001409.

[18] V. Litovski, Z. Randjelovic, and M. Damnjanovic, "Routing in standard cells," in *Proceedings of International Conference on Microelectronics*, vol. 2, Sep. 1995, 461–466 vol.2. DOI: 10.1109/ICMEL.1995.500910.

[19] R. R. Manikandan, V. N. R. Vanukuru, A. Chakravorty, and B. Amrutur, "A parameterized cell design for high-q, variable width and spacing spiral inductors," in *2014 IEEE International Microwave and RF Conference (IMaRC)*, 2014, pp. 312–315. DOI: 10.1109/IMaRC.2014.7039049.

[20] D. Marolt, J. Scheible, G. Jerke, and V. Marolt, "Analog layout automation via self-organization: Enhancing the novel swarm approach," in *2016 IEEE 7th Latin American Symposium on Circuits Systems (LASCAS)*, Feb. 2016, pp. 55–58. DOI: 10.1109/LASCAS.2016.7451008.

[21] B. Prautsch, U. Eichler, T. Reich, and J. Lienig, "Mesh: Explicit and flexible generation of analog arrays," in *2017 14th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*, Jun. 2017, pp. 1–4. DOI: 10.1109/SMACD.2017.7981572.

[22] B. Prautsch, U. Hatnik, U. Eichler, and J. Lienig, "Template-driven analog layout generators for improved technology independence," in *ANALOG 2018; 16th GMM/ITG-Symposium*, Sep. 2018, pp. 1–6.

[23] F. P. Preparata and M. I. Shamos, *Computational Geometry: An Introduction*. Berlin, Heidelberg: Springer-Verlag, 1985, ISBN: 0387961313.

[24] N. Ryzhenko and S. Burns, "Standard cell routing via boolean satisfiability," in *DAC Design Automation Conference 2012*, Jun. 2012, pp. 603–612. DOI: 10.1145/2228360.2228470.

[25] J. Scheible and J. Lienig, "Automation of analog ic layout: Challenges and solutions," in *Proceedings of the 2015 Symposium on International Symposium on Physical Design*, ser. ISPD '15, Monterey, California, USA: Association for Computing Machinery, 2015, pp. 33–40, ISBN: 9781450333993. DOI: 10.1145/2717764.2717781. [Online]. Available: https://doi.org/10.1145/2717764.2717781.

[26] N. A. Sherwani, *Algorithms for VLSI Physical Design Automation*. USA: Kluwer Academic Publishers, 1993, ISBN: 0792392949.

[27] A. Unutulmaz, G. Dündar, and F. V. Fernández, "Lds - a description script for layout templates," in *2011 20th European Conference on Circuit Theory and Design (ECCTD)*, Aug. 2011, pp. 857–860. DOI: 10.1109/ECCTD.2011.6043824.

[28] Z.-W. Wang, I.-L. Tseng, and A. Postula, "Procedural module generation for parameterized layouts," in *IEEE 2013 Tencon - Spring*, 2013, pp. 548–551. DOI: 10.1109/TENCONSpring.2013.6584505.

[29] N. Weste and D. Harris, *CMOS VLSI Design: A Circuits and Systems Perspective*, 4th. USA: Addison-Wesley Publishing Company, 2010, ISBN: 0321547748.

Anexo A

Cálculo de carga para el banco de experimentos usando retardo FO4

El retardo FO4 es una medida del rendimiento que se mantiene relativamente constante al escalar las diferentes tecnología y se define como el retardo visto por una celda considerando una carga equivalente a cuatro veces el tamaño del inversor mínimo de la tecnología. Con el fin de calcular una carga representativa para ambas tecnología, se considera que la capacitancia de compuerta se calcula mediante la siguiente ecuación: W_{eff} x C_O de acuerdo a los parámetros de la tecnología de 180nm, la capacitancia para la biblioteca personalizada es aproximadamente 1fF y el valor es similar para la biblioteca de celdas estándar por lo que se utilizó un capacitor de 1fF para realizar obtener las métricas de rendimiento.

Anexo B

Trazados en 90nm

Celda OR

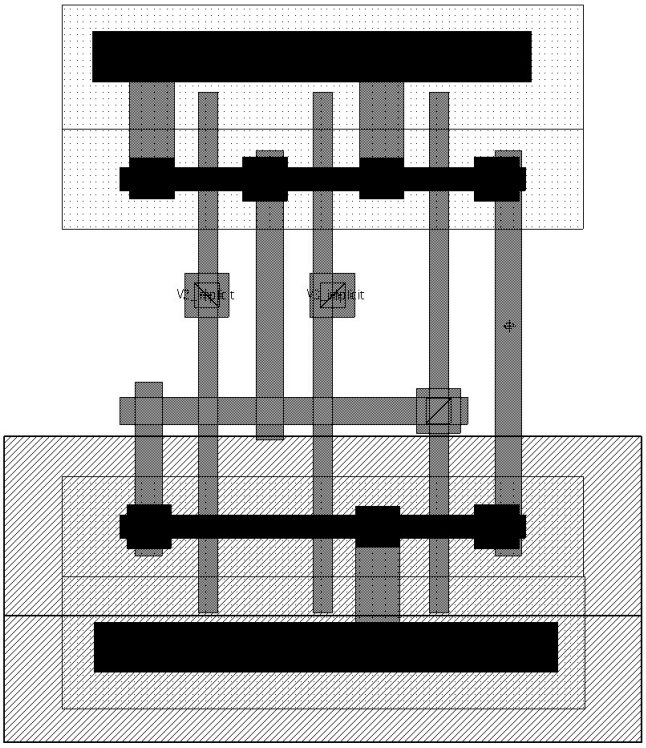


Figure B.1: Trazado completo de una compuerta OR en una tecnología de 90nm generado mediante el generador de celdas

Celda NAND

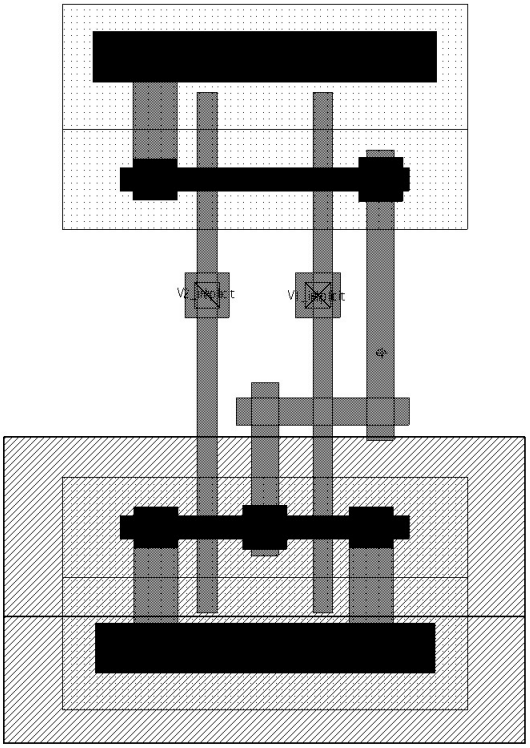


Figure B.2: Trazado completo de una compuerta NAND en una tecnología de 90nm generado mediante el generador de celdas

Celda NOR

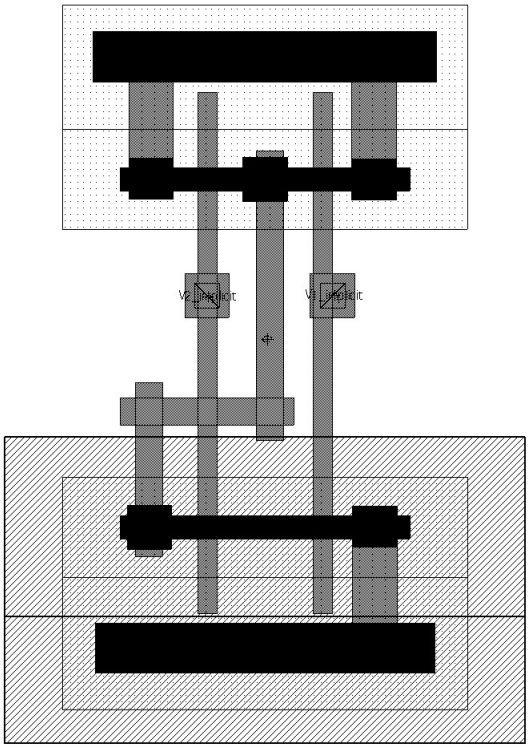


Figure B.3: Trazado completo de una compuerta NOR en una tecnología de 90nm generado mediante el generador de celdas