



UNIVERSIDAD DE BUENOS AIRES  
FACULTAD DE INGENIERÍA  
LABORATORIO DE MICROELECTRÓNICA

TESIS DE GRADO EN INGENIERÍA ELECTRÓNICA

---

**Desarrollo de un Kit de Diseño Interoperable  
y un Conjunto de Celdas Estándar Abiertos  
para un Proceso CMOS Escalable**

---

3 de diciembre de 2015

*Autor:*  
Gabriel A. SANCA

*Director:*  
Dr. Ing. M. GARCÍA INZA  
*Co-director:*  
Ing. O. H. ALPAGO



UNIVERSIDAD DE BUENOS AIRES

FACULTAD DE INGENIERÍA  
LABORATORIO DE MICROELECTRÓNICA  
TESIS DE GRADO EN INGENIERÍA ELECTRÓNICA

**Desarrollo de un kit de diseño interoperable y un conjunto de celdas estándar abiertos para un proceso CMOS escalable**

**Autor** Gabriel Andrés Sanca  
**Director** Dr. Ing. Mariano García Inza  
**Co-Director** Ing. Octavio Hernan Alpago

DICIEMBRE 2015





Tesis presentada para optar al Título de

## INGENIERÍA ELECTRÓNICA

por la

Facultad de Ingeniería de la Universidad de Buenos Aires

Director:

---

Dr. Ing. Mariano García Inza

Co-director:

---

Ing. Octavio Hernán Alpago

Miembros del Jurado:

---

Ing. Adolfo Bertetta

---

Ing. José Alberto Bertuccio

---

Ing. Nicolás Alvarez

Calificación: \_\_\_\_\_

Fecha: \_\_\_\_\_



*A mi familia  
...y al silencio,  
por hacer esto posible*

# Agradecimientos

Ésta sección está planteada como un prólogo, y como todo prólogo, me ayudará a justificar muchas de las cosas hechas por mí y por todos los que me ayudaron en este camino; y siento la enorme necesidad de hacerselo saber. Así, me reservo cierta libertad poética, a la vez que utilizo la primera persona, para escribir estas palabras, pues no son ni para mí, ni para el jurado, ni para el conocimiento humano. Son para personas, que se merecen lo mejor.

Me gustaría comenzar con algo que escribí hace ya un tiempo, pero refleja mucho de lo que siento en este momento:

“Si se plantea rápidamente una definición de felicidad como la siguiente: *La felicidad es un estado de ánimo que se produce en la persona cuando cree haber alcanzado una meta deseada y buena*<sup>1</sup> muchos dirán que es bastante acertada, o que encaja con lo que ellos piensan que es la felicidad. Ahora, analicemos esa frase detenidamente.

Comienza afirmando que la felicidad es un estado de ánimo, o sea, es algo simplemente pasajero, o por lo menos, es factible al cambio fácilmente. Esto lo relaciona además con cierta sensibilidad. La felicidad es sensible, se puede sentir. Seguido a esto, el verbo producir nos indica que no es algo natural, sino algo forzado, en un sentido de generación: **algo externo nos debe ayudar a ser felices**. Pero no termina allí, adhiere, y éste, creo, es el punto más importante, algo más: “*cuando cree haber alcanzado una meta deseada y buena*”. Agrega un cuando. No en cualquier momento, sino en uno específico, luego de concretado lo recientemente descripto. Pero no solo es importante el cuándo, sino también el por qué del cuándo. Cuando cree -no demuestra certeza- haber alcanzado una meta deseada y buena. De aquí se desprenden dos cosas. Primero, cuando dice deseada y buena se está haciendo un juicio ético y moral, es decir, la felicidad es independiente para cada individuo, lo cual nos puede parecer lógico, y creo que es así. Segundo, y esto es lo que me aterra un poco más, es qué es lo bueno y lo deseado,

---

<sup>1</sup>“Felicidad”, <http://es.wikipedia.org/wiki/Felicidad>

que es lo necesario para ser feliz: **cumplir una meta**. Es decir, el camino, el recorrido y el esfuerzo puestos en llegar a tal objetivo, bueno y deseado, no es suficiente para ser feliz. Esto indica que sólo el éxito nos hace felices. Creo que me animo a disentir. ¿Cómo es posible dejar de lado el camino recorrido? ¿El éxito de cumplir nuestros objetivos tiene la misma felicidad sin importar las consecuencias, sin importar los recorridos? ¿El fin justifica los medios? Y algo más preocupante desde mi punto de vista hacia la felicidad, ¿no nos hace felices sortear los obstáculos? ¿Sólo llegar nos genera felicidad? Entonces, ¿qué sucede luego de llegar? Es raro que esperemos el momento de ver cumplidos nuestros sueños para ser felices. Es como que el acto mismo de soñar no nos impregna felicidad.”

Cuando decidí estudiar ingeniería, sabía que me esperaba un largo camino por recorrer. Lo que no sabía en ese momento, era que **aprendería a disfrutar cada etapa del mismo**. Haber aprendido sobre física, química o matemática, fue sólo una excusa: en la facultad “*aprendí a aprender*”. Y eso es mucho más valioso. Eso es una herramienta infinitamente más poderosa que cualquier otra. Y aprendí a caminar por un camino que se resume de una manera simplista y casi obscena en este escaso trabajo de apenas 100 páginas. Pero no fui autodidacta. Hubo, y hay, muchas personas que me acompañaron, y lo siguen haciendo. Y nada de esto hubiese sido posible sin su infinita paciencia, ganas, amistad, amor, compromiso y pasión.

Para ser justo, debo comenzar por mis directores Mariano (García Inza) y Octavio (Alpago); y quien fue una suerte de tutor externo, Ronald Valenzuela, que me brindaron todo su tiempo y conocimiento para ayudarme a que este prólogo tenga su epílogo. Y para ser completamente justo, debo incluir en este primer grupo a Sebastián Pazos, quien también me dio una mano bárbara.

Agradecer a todos los que están leyendo éstas palabras, en especial al jurado: Adolfo Bertetta, Alberto Bertuccio, Nicolás Alvarez; que me prestaron parte de su tiempo para evaluar éste trabajo.

Agradecer al Instituto Austral de Enseñanza de Comodoro Rivadavia, donde tuve una excelente educación académica, y en especial a dos docentes: a Alejandra Cavallieri, quien me presentó formalmente con la física y la matemática; y a Daniel Kirs, quien, sutilmente y sin darme cuenta, me fue introduciendo en la electrónica.

Agradecerle a toda la gente de la UNPSJB, donde comencé a ser ingeniero. A Rómulo, Carlos; a mis compañeros Pancho, Cala, Cristian, Christian; y a todos los maravillosos docentes y profesionales que allí conocí.

Agradecerle a la Facultad de Ingeniería de la UBA. Al Laboratorio de

Física de Dispositivos - Microelectrónica, al Laboratorio de Microelectrónica y todo ese *staff* de genios: Gabriel Redin, Lucas Sambuco, Martín Carrá, Ignacio Martinez, José Lipovetzky y Adrián Faigon.

Un agradecimiento y abrazo especial para Seba Carbonetto, el Colo (Fede Zacchigna), Lucho Natale, Pablo Gómez y Ari Lutenberg por tanto tiempo prestado, por tantas birras, fútbol y por la amistad que se fue forjando en este proceso.

A todo el equipo de MeMO y LabO Sat por el tiempo concedido cuando fuese necesario. Al Centro de Micro y Nanotecnología del Bicentenario del INTI por las instalaciones y la buena onda de siempre.

Agradecer a Nicolas Calarco quien impulsó y apoyó el uso del kit en el track básico de la EAMTA2016 (Escuela Argentina de Micro-Nanoelectrónica Tecnología y Aplicaciones) y a Diego Martín, quien fue una especie de brújula en los tormentosos inicios.

Agradecerle a mis compañeros de FIUBA Fabri Alcalde, Andrés Dalmati, Pablo Boer, Leo Giaccone, Nacho Lesser, Pedro Barri y Gabriel Postolov.

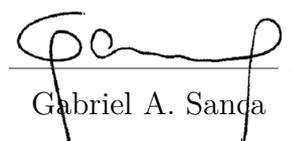
A José González, Alan Parson y Lana del Rey, por haber sido el *sound-track* de esta tesis.

Por último, y jamás menos importante, debo agradecerle a mi familia y a mis amigos, principalmente a Fede Die, Nico Die, Emi Ágreda, Bren Roqueta, Tincho Colombo, Nico Montaruli, Mauro Montaruli, Coco Valles, Agustín Baron y a la hermosa Florencia Mathiasen por todo el apoyo, y por haber sido mi sustento moral, ético, ideológico e intelectual.

*“Nada me importa más que hacer el recorrido, más que saber a donde voy”*<sup>2</sup>

*“En la vida hay que ser felices, el resto, viene solo”*<sup>3</sup>

¡Gracias... totales!



Gabriel A. Sanca

---

<sup>2</sup>“Magia”, Gustavo Cerati

<sup>3</sup>Frase propia



# Resumen

En este trabajo se presenta el desarrollo de un Kit de Diseño de Procesos Interoperable (iPDK) y el diseño de un conjunto de celdas digitales estándar.

El iPDK se desarrolla para el proceso escalable CMOS SCN3ME\_SUBM de ON Semiconductors, accesible a través de MOSIS. El mismo incluye reglas de DRC y LVS, modelos de SPICE, archivos de tecnología, archivos de verificación, archivos de extracción, scripts de ejecución, bibliotecas de símbolos y celdas paramétricas (PCells).

Utilizando este iPDK se diseñaron un conjunto celdas estándar para su uso en la síntesis de circuitos digitales con herramientas de diseño automatizado (EDA).

Para corroborar el correcto funcionamiento del kit y para probar el funcionamiento de las celdas estándar, se fabricó un chip con estructuras de testeo, el cual fue medido exitosamente en el laboratorio.



# Índice

<b>Agradecimientos</b>	<b>I</b>
<b>Resumen</b>	<b>V</b>
<b>1. Introducción al trabajo de tesis</b>	<b>1</b>
1.1. Objeto y Área de la Tesis . . . . .	2
1.1.1. Motivaciones . . . . .	2
1.1.2. Alcance proyectado para la tesis . . . . .	3
1.2. Diseño VLSI . . . . .	3
1.2.1. Introducción . . . . .	3
1.2.2. Flujo de Diseño VLSI . . . . .	5
<b>2. Tecnología CMOS</b>	<b>9</b>
2.1. Dispositivo MOSFET . . . . .	10
2.2. Fabricación CMOS . . . . .	14
2.2.1. Tecnología Planar . . . . .	14
2.2.1.1. Obtención del wafer . . . . .	14
2.2.1.2. Oxidación . . . . .	15
2.2.1.3. Etching (remoción) . . . . .	17
2.2.1.4. Fotolitografía . . . . .	19
2.2.1.5. Dopaje . . . . .	20
2.2.2. Reglas de Diseño . . . . .	21
2.3. Circuitos digitales CMOS . . . . .	22
2.3.1. El transistor MOS como llave . . . . .	23

2.3.2.	El inversor CMOS . . . . .	24
2.3.2.1.	Análisis estático . . . . .	25
2.3.2.2.	Transferencia . . . . .	27
2.3.2.3.	Márgenes de ruido . . . . .	27
2.3.2.4.	Análisis dinámico . . . . .	29
2.3.3.	Lógica combinacional . . . . .	31
2.4.	Síntesis de compuertas . . . . .	31
2.4.1.	Algoritmo para sintetizar funciones lógicas . . . . .	32
2.4.2.	Lógica Secuencial . . . . .	33
2.4.2.1.	Latches . . . . .	33
2.4.2.2.	Flip-flops . . . . .	33
2.5.	Procesos CMOS . . . . .	34
2.5.1.	Procesos propietarios . . . . .	35
2.5.2.	Procesos escalables . . . . .	36
2.5.3.	Proceso C5NF - SCMOS_SUBM . . . . .	36
2.6.	Conclusiones . . . . .	37
<b>3.</b>	<b>Desarrollo de un Kit de Diseño Interoperable</b>	<b>39</b>
3.1.	iPDK . . . . .	39
3.1.1.	Estructura Propuesta . . . . .	40
3.2.	Archivos de tecnología . . . . .	41
3.2.1.	Tech files . . . . .	41
3.2.1.1.	Reglas para escribir un archivo de tecnología . . . . .	43
3.2.2.	Display Resource File . . . . .	43
3.3.	Componentes Front-End . . . . .	44
3.3.1.	Component Description Format . . . . .	44
3.3.2.	Callbacks . . . . .	45
3.4.	PCells . . . . .	46
3.4.1.	Aplicación . . . . .	46
3.4.2.	Implementación . . . . .	47
3.5.	PyCells . . . . .	47

3.5.1.	PyCell Studio . . . . .	48
3.5.2.	Python . . . . .	49
3.6.	Archivos de verificación . . . . .	49
3.6.1.	DRC . . . . .	50
3.6.2.	LVS . . . . .	50
3.7.	Archivos de extracción . . . . .	52
3.8.	Conclusiones . . . . .	52
<b>4.</b>	<b>Diseño de un conjunto de celdas estándar</b>	<b>53</b>
4.1.	Analogía Lego . . . . .	54
4.2.	Diseño . . . . .	54
4.2.1.	Ruteo <i>Over-the-Cell</i> . . . . .	55
4.2.2.	Pitch matching . . . . .	57
4.2.3.	Estilos de layout . . . . .	58
4.2.3.1.	Different-height cell in one row . . . . .	58
4.2.3.2.	Single-height cells in single Row . . . . .	58
4.2.3.3.	Single-and double-or multiple-height cells . . . . .	59
4.2.4.	Grilla de ruteo . . . . .	59
4.2.5.	Parámetros adoptados en el diseño . . . . .	60
4.2.6.	Dimensionamiento de los transistores . . . . .	61
4.2.7.	Inversor . . . . .	63
4.2.8.	Compuertas NAND y NOR . . . . .	64
4.2.9.	Celdas compuestas . . . . .	65
4.2.10.	Tri-states . . . . .	65
4.2.11.	Multiplexores . . . . .	67
4.2.12.	Circuitos secuenciales . . . . .	68
4.2.12.1.	Latches . . . . .	68
4.2.12.2.	Flip-flops . . . . .	68
4.3.	Extracción de parásitos . . . . .	71
4.4.	Conclusiones . . . . .	72

<b>5. Resultados</b>	<b>73</b>
5.1. PDK . . . . .	73
5.2. Celdas Estándar . . . . .	74
5.2.1. Nomenclatura . . . . .	75
5.2.2. Parámetros globales . . . . .	75
5.3. Resultados experimentales . . . . .	76
5.3.1. Estructuras de testeo . . . . .	76
5.3.2. Análisis estático . . . . .	78
5.3.3. Análisis dinámico . . . . .	78
5.3.3.1. Análisis de los resultados . . . . .	79
5.4. Conclusiones . . . . .	80
<b>6. Resumen y Conclusiones</b>	<b>83</b>
6.1. Trabajos a futuro . . . . .	84
<b>A. Layout de las Celdas Estándar</b>	<b>85</b>
<b>B. Mediciones</b>	<b>91</b>
B.1. Osciladores en anillo . . . . .	91
<b>C. Capacidades parásitas</b>	<b>95</b>
C.1. Computando capacidades . . . . .	95
C.1.1. Capacidad gate-drain $C_{gd12}$ . . . . .	95
C.1.2. Capacidad de difusión $C_{ab1}$ y $C_{ab2}$ . . . . .	96
C.2. Capacidad de ruteo $C_w$ . . . . .	96
C.3. Capacidad de gate $C_{g3}$ y $C_{g4}$ . . . . .	97
<b>D. Mosis Test Data</b>	<b>99</b>
<b>E. Trabajos publicados</b>	<b>103</b>
<b>Glosario</b>	<b>105</b>

# Índice de Tablas

5.1.	Parámetros de las pycells diseñadas. . . . .	75
5.2.	Especificaciones físicas. . . . .	76
5.3.	Resultados de las mediciones de los osciladores en anillo. . . . .	80
5.4.	Resultados de las mediciones de los osciladores en anillo realizadas por MOSIS. . . . .	80
B.1.	Resultados de las mediciones de los diferentes osciladores en anillo. . . . .	93



# Índice de Figuras

1.1.	Ley de Moore. Predicción original [3]. . . . .	4
1.2.	Flujo de diseño digital. . . . .	5
2.1.	Esquema de un capacitor MOS. . . . .	10
2.2.	Densidad de portadores para una tensión $V_{GB} = V_T$ , en una juntura MOS de sustrato tipo P y polisilicio $N^+$ . . . . .	11
2.3.	Esquema de un transistor nMOS. . . . .	12
2.4.	Esquema de un transistor nMOS. Vista de planta. . . . .	13
2.5.	Sección transversal de un inversor CMOS [10]. . . . .	14
2.6.	Proceso Czochralski: obtención del wafer [13]. . . . .	15
2.7.	Espesor de la oxidación en función del tiempo [13]. . . . .	16
2.8.	Perfiles del proceso de etching [17]. . . . .	18
2.9.	Pasos del proceso de transferencia del patrón de $SiO_2$ [15]. . . . .	20
2.10.	CMOS: PUN + PDN. . . . .	23
2.11.	El transistor como llave [10]. . . . .	25
2.12.	Esquemático de un inversor CMOS. . . . .	25
2.13.	Curva de transferencia del inversor CMOS [15]. . . . .	27
2.14.	Transferencia del inversor CMOS linealizada por tramos [15]. . . . .	28
2.15.	Capacidades parásitas. Influyen en el comportamiento tran- sitorio del par inversor en cascada [15]. . . . .	29
2.16.	Tiempo de propagación y tiempos de rise y fall [10]. . . . .	30
2.17.	Diferentes topologías de flip-flops [19]. . . . .	35
3.1.	Estructura de archivos de un iPDK [23]. . . . .	41

3.2.	Instanciando una PyCell. . . . .	48
3.3.	Reglas DRC para el layer POLY [20]. . . . .	51
4.1.	Layout de un procesador MIPS de 8 bits [10]. . . . .	55
4.2.	Distribución de celdas con canales de ruteo [27]. . . . .	56
4.3.	Layout de un microcontrolador de 8 bits. . . . .	56
4.4.	Snap-together cells. Los rieles de VDD y GND se superponen. . . . .	57
4.5.	Esquema de las celdas <i>different-height cell in one row</i> . . . . .	58
4.6.	Esquema de las celdas <i>single-height cells in single row</i> . . . . .	59
4.7.	Esquema de las celdas <i>single-and double-or multiple-height cells</i> . . . . .	59
4.8.	Spacing. . . . .	60
4.9.	Grilla de ruteo. . . . .	62
4.10.	Simulación de la transferencia del inversor CMOS mínimo variando las características de los transistores. . . . .	64
4.11.	Celda CMOS compuesta para la función $Y = (A \cdot B) + (C \cdot D)$ [10]. . . . .	66
4.12.	Buffer tri-state [10]. . . . .	66
4.13.	Inversor tri-state [10]. . . . .	67
4.14.	Multiplexor transmission gate [10]. . . . .	68
4.15.	Latch D [10]. . . . .	69
4.16.	Flip-flop D [10]. . . . .	70
4.17.	Esquemático de la celda DFFR. . . . .	71
4.18.	Simulación estática de la celda DFFR. . . . .	71
5.1.	Layout de las PyCells de los transistores MOS. . . . .	74
5.2.	Layout de la PyCell básica de la resistencia HR. . . . .	74
5.3.	Layout de la PyCell de la resistencia HR con algunos parámetros modificados respecto de los parámetros por defecto. . . . .	74
5.4.	Diagrama en bloques de las estructuras de testeo diseñadas. . . . .	77
5.5.	<i>Top level</i> del tiny chip fabricado por MOSIS. . . . .	77
5.6.	Banco de mediciones estáticas. . . . .	78

5.7.	Banco de mediciones dinámicas. . . . .	79
A.1.	Celdas AND. . . . .	85
A.2.	Celdas compuestas. . . . .	85
A.3.	Buffers. . . . .	86
A.4.	Layout del DFFR. . . . .	86
A.5.	Fillers. . . . .	86
A.6.	Inversores. . . . .	87
A.7.	Multiplexores. . . . .	87
A.8.	Celdas NAND. . . . .	88
A.9.	Celdas NOR. . . . .	88
A.10.	Celdas OR. . . . .	89
A.11.	Celdas XOR. . . . .	89
B.1.	Tiempo de propagación estimado a partir de las mediciones de frecuencia de los diferentes osciladores en anillo. . . . .	92
C.1.	Efecto Miller. . . . .	96



# Capítulo 1

## Introducción al trabajo de tesis

Con la omnipresencia de la tecnología inalámbrica y el cambio inevitable hacia mayores niveles de integración, los circuitos integrados actuales cada vez disponen de mayor cantidad de elementos analógicos o de señal mixta (AMS, del inglés *Analog/Mixed-Signal*). A su vez, la repentina demanda de dispositivos de Internet de las Cosas (IoT, del inglés *Internet Of Things*) crea requisitos únicos para un flujo completo de diseño AMS, que sea asequible y fácil de usar, pero lo suficientemente potente como para crear una gama muy diversa de productos para el despliegue de IoT. Más que nunca, diseñadores, *foundries*<sup>1</sup> y desarrolladores de herramientas EDA<sup>2</sup> deben poder familiarizarse con el diseño AMS a nivel PDK.

En el entorno del mundo puramente digital, las herramientas de automatización de diseño tienen el reto de mantener el ritmo de la complejidad del diseño con un gran número de transistores disponibles en los nodos de proceso actuales. Contrariamente, el diseño analógico es difícil y lento, y a la vez que la complejidad aumenta, se convierte en un serio cuello de botella en el flujo de diseño de señal mixta. Aunque existen herramientas de diseño de AMS, este tipo de diseño es mucho más difícil de automatizar. Además, es recomendable tener un conocimiento detallado y acabado del proceso de fabricación con el cual se está trabajando [1].

El presente trabajo de tesis se encuentra enfocado en el contexto de diseño de hardware digital para dispositivos ASIC<sup>3</sup>. El resultado del trabajo es

---

<sup>1</sup>Se utiliza el término *foundry* para referirse a las fábricas de circuitos integrados que llevan adelante los procesos fotolitográficos que definen a los chips, su empaquetado y testeo. Se detalla esto en la Sección 2.2.

<sup>2</sup>Automatización de diseño electrónico, del inglés *Electronic design automation*, es una categoría de herramientas de software para el diseño de sistemas electrónicos integrados.

<sup>3</sup>Acrónimo del inglés *application-specific integrated circuit*, Circuito Integrado para

brindarle soporte a cada etapa del flujo de desarrollo y diseño de circuitos integrados a gran escala de integración.

## 1.1. Objeto y Área de la Tesis

La tesis tiene como objetivo principal desarrollar, diseñar, caracterizar y organizar un Kit Abierto de Diseño de Proceso, según la **IPL Alliance**<sup>4</sup>, llamados “Interoperables” (iPDK, por sus siglas en inglés) para un proceso CMOS escalable. Esto incluye las reglas de diseño DRC y LVS, modelos de SPICE, archivos de tecnología, archivos de verificación y extracción, scripts de ejecución, bibliotecas de símbolos, celdas paramétricas (PCells) y celdas digitales estándar.

El plan de tesis propuesto plantea la motivación de llevar adelante un flujo de trabajo completo desde el desarrollo del transistor, hasta la caracterización de una librería digital, con la cual se puede diseñar todo tipo de circuitos digitales integrados. De esta forma, se completa el flujo de diseño digital, permitiendo la síntesis en silicio de circuitos integrados digitales a partir de código HDL, como puede ser la implementación de un protocolo de comunicación, circuitos de señal mixta o inclusive un microcontrolador.

La idea central es diseñar un conjunto de celdas estándar para utilizar en futuros diseños digitales usando las herramientas EDA/CAD, con lo cual es necesario desarrollar a fondo el PDK correspondiente. Las librerías quedarán disponibles para el uso por parte de otras universidades.

### 1.1.1. Motivaciones

- Llevar adelante un flujo de trabajo completo desde el desarrollo del transistor, hasta la caracterización de una librería digital, con la cual se puede diseñar un amplio espectro de circuitos digitales. De esta forma, se parte desde un dispositivo semiconductor básico, para poder diseñar cada etapa y concluir en un circuito complejo, como puede ser un protocolo de comunicación implementado en silicio, circuitos de señal mixta o inclusive un microcontrolador.

---

Aplicaciones Específicas.

<sup>4</sup>El consorcio IPL (del inglés *Interoperable PDK Libraries*) Alliance es una organización conformada por varios actores de la industria de semiconductores, establecida para desarrollar un ecosistema interoperable para el diseño *custom* de circuitos integrados. El enfoque actual es crear y promover estándares para el desarrollo de PDKs y *analog constraints* [2].

- Familiarizarse a fondo con las herramientas de trabajo EDA.
- Familiarizarse con el estado del arte del diseño de circuitos integrados.
- Desarrollar un producto final funcional y operativo, que permita el desarrollo de otras áreas de investigación dentro del Laboratorio de Microelectrónica.
- Desarrollar un iPDK que pueda ser distribuido y utilizado por otras universidades, públicas y privadas, no solo de Argentina sino también de la región y el mundo.
- Sentar las bases para la generación de un flujo de trabajo de desarrollo colectivo, generando documentación abierta, en el área de diseño de circuitos integrados.
- Generar bibliografía especializada en idioma español

### 1.1.2. Alcance proyectado para la tesis

- Desarrollar en detalle las celdas paramétricas para el iPDK y los archivos de tecnología, incluyendo reglas de DRC y LVS, así como tech files para Custom Designer y para PyCell Studio.
- Desarrollar las celdas digitales previstas.
- Evaluar los resultados obtenidos mediante las mediciones correspondientes.
- Proponer trabajos futuros y/o mejoras.

## 1.2. Diseño VLSI

### 1.2.1. Introducción

El 19 de abril 1965, Gordon Moore, en su famoso artículo publicado en la revista Electronics, titulado “*Cramming More Components onto Integrated Circuits*”, definió una estrategia de mercado para la industria de los semiconductores, en donde predijo que la cantidad de transistores en los circuitos

integrados se duplicaría cada año<sup>5</sup> [3]. Esta afirmación se conocería con el tiempo como “Ley de Moore”, cuyo cumplimiento se puede constatar hasta el día de hoy [4].

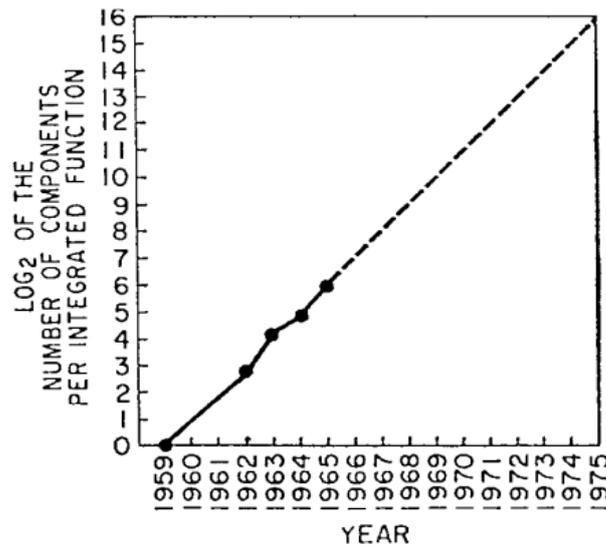


Figura 1.1: Ley de Moore. Predicción original [3].

Éste veloz crecimiento de la tecnología de integración a gran escala ha sido, y sigue siendo posible, gracias a la automatización de las diversas etapas implicadas en el diseño y fabricación de chips digitales. Los circuitos integrados a gran escala se componen de un número muy elevado de dispositivos electrónicos, contruidos por capas (*layers*) de varios materiales diferentes, de una manera bien definida, sobre una base de silicio, llamada oblea (*wafers*). El diseñador transforma una descripción funcional del circuito en una descripción geométrica. El diseño consiste en un conjunto de formas geométricas planas en las diferentes capas. La geometría se comprueba para asegurar que cumple con todos los requisitos de fabricación (reglas DRC) y se contrasta con el diseño esquemático (reglas LVS). El resultado es un conjunto de archivos que describen el diseño, que se utilizan para producir las *máscaras*. Durante la fabricación, estas máscaras se utilizan para transmitir los diferentes patrones geométricos al silicio, según una secuencia de pasos fotolitográficos que definen al proceso [6].

**VLSI** es la sigla para *Very Large Scale Integration* (integración a gran escala), y se refiere al conjunto de tecnologías y diseños que definen un gran

<sup>5</sup>Luego, en 1975, Moore modificó su predicción, estableciendo que la cantidad de transistores se duplicaría cada 24 meses [5].

nivel de integración de dispositivos electrónicos en un área finita y reducida de silicio. Los circuitos integrados VLSI son generalmente circuitos digitales, donde el gran número de transistores posibilita un gran poder de procesamiento.

### 1.2.2. Flujo de Diseño VLSI

El ciclo de diseño VLSI comienza con una especificación formal de un chip que, siguiendo una serie de pasos bien definidos, da por resultado un circuito integrado digital. Un ciclo de diseño típico puede ser representado por el diagrama de flujo mostrado en la Figura 1.2.

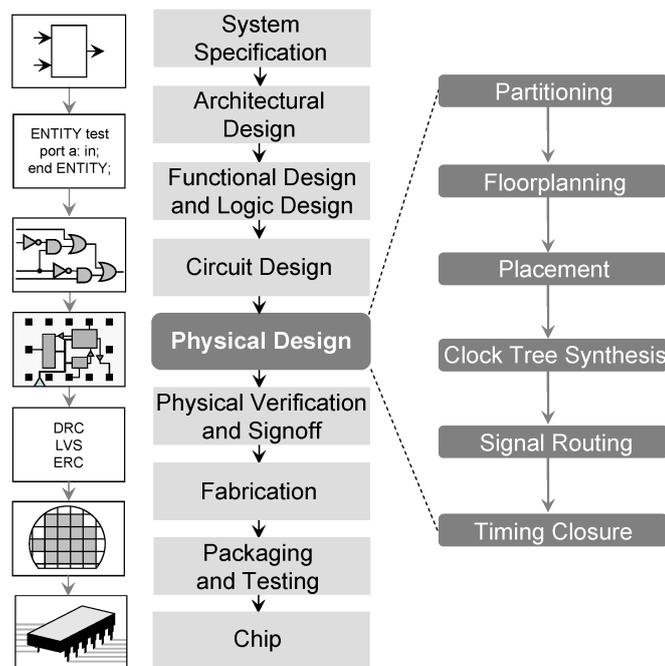


Figura 1.2: Flujo de diseño digital.

1. **Especificación del sistema:** El primer paso de cualquier proceso de diseño tiene por objeto establecer las especificaciones del sistema, en una representación de alto nivel. Los factores a tener en cuenta en este proceso incluyen: funcionalidad, rendimiento y dimensiones físicas. La especificación de un sistema es un compromiso entre las necesidades del mercado, la tecnología y la viabilidad económica. Los resultados

finales son especificaciones para el tamaño, la velocidad, potencia y funcionalidad del sistema VLSI.

2. **Diseño a nivel arquitectura:** Luego, se define la arquitectura básica del sistema. Esto incluye decisiones como RISC versus CISC, número de ALU, unidades de punto flotante, número y estructura de pipelines, tamaño de las memorias caché, entre otros. El resultado del diseño arquitectónico es una especificación de la micro arquitectura.
3. **Diseño funcional:** En este paso se identifican las principales unidades funcionales del sistema. Esto también identifica los requisitos de interconexión entre las unidades. El área, el consumo, y otros parámetros de cada unidad son estimados. Por ejemplo, se puede especificar que se requiere una multiplicación, pero no se especifica exactamente en qué modo debe implementarse dicha función. Se puede optar por una gran variedad de hardware que realice multiplicaciones, en función de las características que se requieran optimizar. La idea clave es definir el comportamiento, en términos de entrada y salida, sin especificar su estructura interna.
4. **Diseño lógico:** En este paso se definen aspectos como el flujo de control, ancho de palabra, registros de asignación, operaciones aritméticas y operaciones lógicas de diseño. Esta descripción se denomina *register-transfer level* (RTL). El RTL se expresa en algún lenguaje de descripción de hardware (HDL), como VHDL o Verilog. Esta descripción se utiliza además en las etapas de simulación y verificación. Las expresiones booleanas se reducen al mínimo para lograr que el diseño de la lógica digital sea la más pequeña que se ajuste al diseño funcional. En algunos casos especiales, el diseño a nivel lógico se puede automatizar utilizando herramientas de síntesis de alto nivel. Estas herramientas, Matlab por ejemplo, producen una descripción RTL partir de una descripción del comportamiento del diseño, aunque su rendimiento no es óptimo.
5. **Diseño de Circuitos:** El propósito del diseño a nivel de circuitos es desarrollar una representación circuital basado en la lógica digital sintetizada. Las expresiones booleanas se convierten en esquemáticos de un circuito concreto, teniendo en cuenta los requisitos de velocidad y potencia del diseño original. Existen numerosas técnicas y análisis para poder asegurarse de cumplir con los requerimientos técnicos y funcionales.

6. **Diseño físico:** En este paso la representación de un circuito, en formato “*netlist*”, se convierte en una representación geométrica. Es un proceso muy complejo y, en general, se lo divide en varias etapas. Puede ser completa o parcialmente automatizado, y el diseño se puede generar directamente desde la netlist por herramientas de diseño de síntesis. La mayor parte de la disposición de un diseño de alto rendimiento, o las etapas más críticas, pueden realizarse manualmente, mientras que muchos diseños de bajo o medio rendimiento, o diseños que necesitan disminuir sus tiempos de fabricación con fines comerciales, pueden hacerse automáticamente.
7. **Fabricación:** Luego de la etapa de verificación, el diseño está listo para su fabricación. Los archivos con datos del diseño se convierten en máscaras fotolitográficas, una para cada layer. Las máscaras identifican diferentes espacios en la oblea, donde ciertos materiales deben ser depositados, difundidos o incluso eliminados. Las extremadamente pequeñas dimensiones de los dispositivos VLSI requieren que las obleas de silicio sean pulidas a la perfección. El proceso de fabricación consta de varias etapas que implican la deposición y la difusión de diversos materiales sobre la oblea. Durante cada paso se utiliza una máscara. Varias docenas de máscaras pueden utilizarse para completar el proceso de fabricación.
8. **Packaging, pruebas y depuración:** Por último, la oblea se fabrica y se cortan en chips individuales, llamados *dies*. Cada chip se empaqueta y se prueba para asegurar que cumple con todas las especificaciones de diseño y que funciona correctamente. Existen diversos encapsulados, que varían según la temperatura que se debe disipar, la cantidad de pines requeridos, el tipo de montura o soldado [6].



# Capítulo 2

## Tecnología CMOS

CMOS es el acrónimo del inglés *Complementary Metal Oxide Semiconductor*, y define la familia lógica más extendida en la actualidad empleada para fabricar circuitos integrados digitales. Es utilizada en diversos sistemas, tales como microprocesadores, microcontroladores, memorias estáticas y dinámicas e infinidad de circuitos digitales y de señal mixta. Además, también se utiliza para fabricar circuitos integrados analógicos, tales como sensores de imagen, conversores de datos, y todo tipo de transductores. En 1963, mientras trabajaba para Fairchild Semiconductor, Frank Wanlass fue quien patentó la tecnología CMOS [7].

Las dos características más sobresalientes de los dispositivos CMOS son su bajo consumo de energía y su alta inmunidad al ruido. Ésta tecnología permite integrar una gran cantidad de dispositivos, y así obtener una alta densidad de funciones lógicas en un mismo chip. Fue principalmente por esta razón que CMOS se convirtió en la tecnología más utilizada para la implementación de chips VLSI. De ésta forma, por sus características, la familia lógica CMOS es la que más se aproxima a las características de una familia lógica ideal [8].

En éste capítulo se describe cualitativamente el dispositivo MOSFET, presentando sus principales características, así como también se presenta en detalle los principales conceptos de la tecnología CMOS, desde los procesos de fabricación hasta la funcionalidad de algunos circuitos digitales elementales CMOS.

## 2.1. Dispositivo MOSFET

El transistor de efecto de campo de juntura metal óxido semiconductor, MOSFET del inglés *Metal Oxide Semiconductor Field Effect Transistor*, es sin duda el dispositivo de estado sólido de mayor éxito comercial, siendo el componente principal en los chips VLSI. Para comprender su funcionamiento, es necesario primero entender el comportamiento de la juntura metal-óxido-semiconductor.

Un capacitor MOS es una estructura compuesta por una compuerta o “gate”, un material dieléctrico y un sustrato o “bulk”, tal como se ilustra en la Figura 2.1. El valor  $t_{ox}$  representa el espesor del óxido, siendo éste mucho menor a las restantes dimensiones de la estructura. En la actualidad, el gate se fabrica utilizando silicio policristalino, denominado “polisilicio” (a.k.a. *poly*), el cual se dopa fuertemente para disminuir su resistividad. El material dieléctrico es una delgada capa de dióxido de silicio ( $\text{SiO}_2$ ), crecido por un proceso de oxidación seca (ver Sección 2.2.1.2). La excelente calidad de la interfaz Si- $\text{SiO}_2$  es uno de los factores principales para el desarrollo de la tecnología MOS. El sustrato es simplemente la oblea de Si, que se halla por lo general levemente dopada [9].

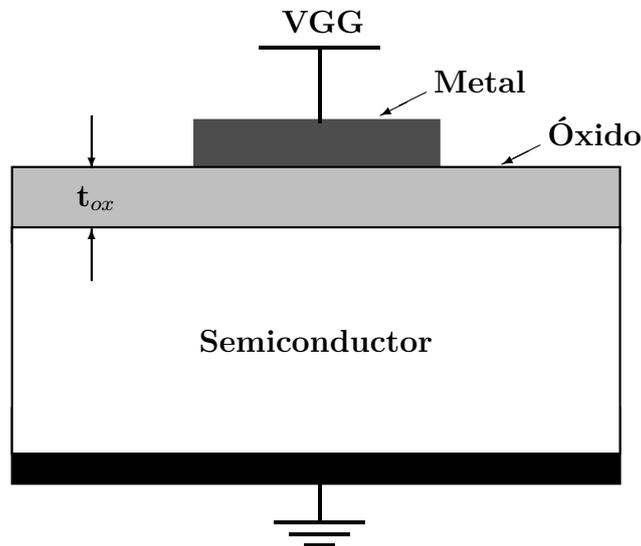


Figura 2.1: Esquema de un capacitor MOS.

La juntura metal-óxido-semiconductor presenta diferentes regímenes de

operación, dependiendo de la tensión entre gate y bulk. Para comprender el funcionamiento de los transistores MOS, debemos concentrarnos particularmente en los regímenes de inversión, en el cual se acumulan portadores minoritarios en la interfaz Si-SiO<sub>2</sub>, y corte.

La Figura 2.2 representa la densidad de portadores, en escala logarítmica, en función de las dimensiones de la juntura. Los valores positivos de  $x$  representan el bulk, y los valores negativos, hasta  $-t_{ox}$ , el espesor del óxido. Este gráfico se obtiene para una juntura MOS con un bulk tipo P y polisilicio tipo N<sup>+</sup>.

En condición de equilibrio térmico con tensión de gate nula, las cargas dentro de la estructura se acomodan debido a los diferentes potenciales de los materiales involucrados. La distribución de portadores para esta situación se muestra en la Figura 2.2, en líneas punteadas. El campo eléctrico presente en el aislante genera un aumento de portadores minoritarios en la interfaz Si-SiO<sub>2</sub> y una zona de vaciamiento en el sustrato semiconductor. Al aplicarse una tensión  $V_{GB}$  positiva entre el gate y el sustrato, aumenta la densidad de portadores minoritarios en la interfaz.

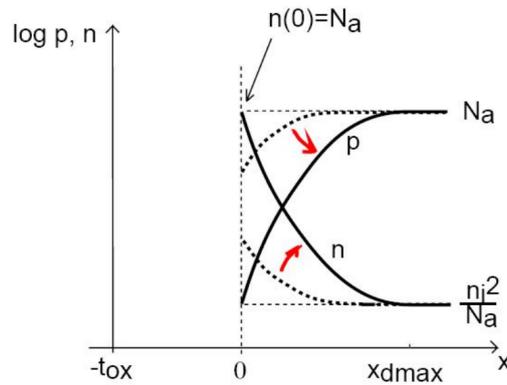


Figura 2.2: Densidad de portadores para una tensión  $V_{GB} = V_T$ , en una juntura MOS de sustrato tipo P y polisilicio N<sup>+</sup>.

Se define como tensión umbral ( $V_T$ ) al potencial que debe aplicarse en el gate para que los portadores minoritarios en la interfaz igualen la concentración del dopaje. Esta situación se esquematiza en línea continua en la Figura 2.2. Superado éste límite, no es posible desprestigiar la contribución de los portadores minoritarios a la electrostática de la juntura.

Ahora, la única forma de balancear la electrostática de la juntura es la aparición de carga superficial en la interfaz Si-SiO<sub>2</sub>, entre el sustrato y el óxido. Ésta es la llamada condición de inversión, ya que en esa zona del material

existe mayor cantidad de portadores minoritarios que mayoritarios. Éste exceso de carga, debido a la presencia de portadores minoritarios, se denomina **canal**. Para transistores tipo nMOS, este canal será de electrones, debido al aumento de portadores minoritarios en un sustrato tipo P, mientras que para transistores tipo pMOS, este canal será de huecos, debido al aumento de portadores minoritarios en un sustrato tipo N.

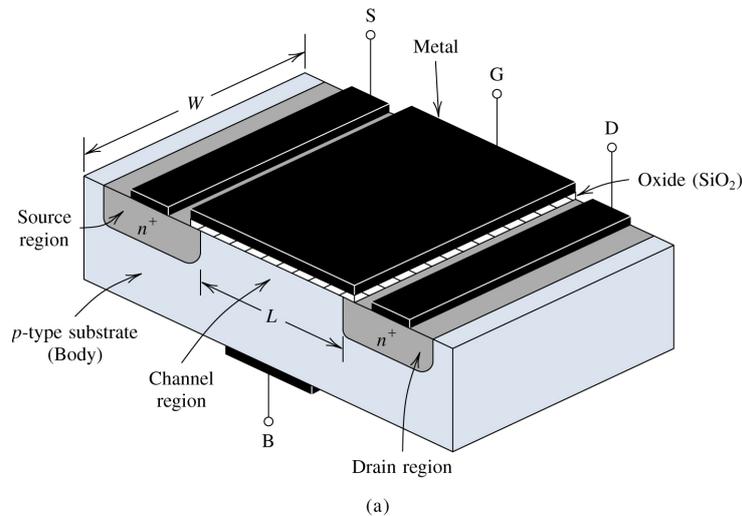


Figura 2.3: Esquema de un transistor nMOS.

Constructivamente, para transformar una juntura MOS en un transistor, es necesario generar puntos de acceso al canal formado debajo del óxido. En el caso del transistor nMOS, con sustrato tipo P y canal conformado por electrones, el acceso será a través de zonas fuertemente dopadas tipo N, y en el caso del transistor pMOS, con sustrato tipo N y canal conformado por huecos, las zonas de acceso están fuertemente dopados tipo P.

A éstas difusiones se las denomina *source* y *drain*, y poseen conexión eléctrica hacia exterior. Se define al source como el contacto del cual parten los portadores, y deberá ser conectado al menor potencial en los transistores tipo nMOS y al mayor potencial en los transistores pMOS; y al drain como el contacto hacia donde llegan los portadores, y deberá ser conectado al mayor potencial en los transistores nMOS y al menor potencial en los transistores tipo pMOS.

Al aplicar una tensión mayor que  $V_T$  se formará canal debajo del óxido, es decir carga por presencia de portadores minoritarios. Si se aplica una tensión entre drain y source, los portadores generarán una corriente debido al fenómeno de arrastre, producido por el campo eléctrico. Esta corriente

puede ser controlada tanto por la tensión aplicada entre el gate y el sustrato, que modula la carga debajo del óxido, como por la tensión aplicada entre drain y source, que modulan el campo eléctrico y la corriente de arrastre. A éste régimen de operación del MOSFET se lo denomina **tríodo o lineal**.

Cuando la tensión drain-source ( $V_{DS}$ ) supera al valor ( $V_{GS} - V_T$ ), se produce el efecto denominado “*pinch-off*” o estrangulamiento del canal, en donde la carga de inversión en el drain se iguala a cero. Aquí, la corriente es independiente del campo eléctrico generado por el potencial  $V_{DS}$ , y a éste régimen de operación se lo denomina **saturación**.

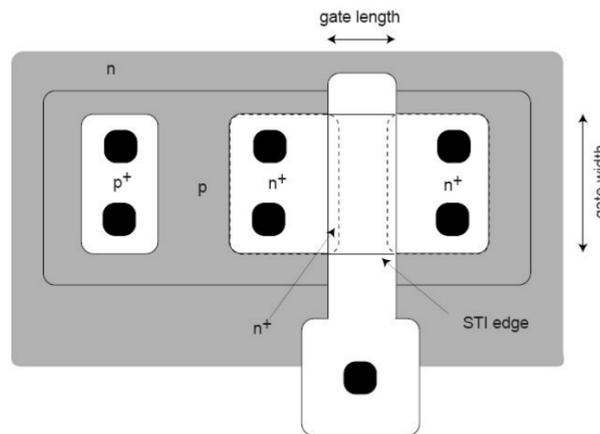


Figura 2.4: Esquema de un transistor nMOS. Vista de planta.

Para el enfoque digital, es importante tener presentes los regímenes de corte y saturación del transistor MOS. En el primer caso, la corriente  $I_D$  será siempre cero, al ser  $V_{GS} < V_T$ , y no formarse canal. Para la condición de saturación, en donde  $V_{GS} > V_T$  y  $V_{DS} > V_{GS} - V_T$ , la ecuación que gobierna la corriente del MOSFET es:

$$I_D = \frac{k}{2} (V_{GS} - V_T)^2 [1 + \lambda(V_{DS} - V_{DSsat})] \quad (2.1)$$

con  $k = \mu_x \frac{W}{L} C_{ox}$

El término  $\lambda$  en este contexto es una constante de proporcionalidad que introduce el efecto de modulación del largo del canal. No se debe confundir con la unidad métrica de los procesos escalables.

## 2.2. Fabricación CMOS

En la presente sección se presenta brevemente las principales etapas que constituyen al proceso de fabricación en tecnología CMOS, desde la obtención de la oblea hasta el empaquetado final de los chips.

### 2.2.1. Tecnología Planar

Los circuitos integrados, se fabrican sobre obleas de silicio delgadas de alta pureza, denominadas *wafers*. Se puede entonces examinar la disposición física de los transistores desde dos perspectivas: vista superior (es decir observando desde la superficie) o sección transversal (obtenida mediante el corte de la oblea).

La Figura 2.5 muestra una sección transversal correspondiente a un inversor y su esquemático. En este diagrama, el inversor está construido sobre un sustrato de tipo p. Los gates de los transistores están unidos entre sí en algún lugar hacia el interior de la página y forman la entrada A. La salida Y está formada por las metalizaciones de entre los drains de los dos transistores.

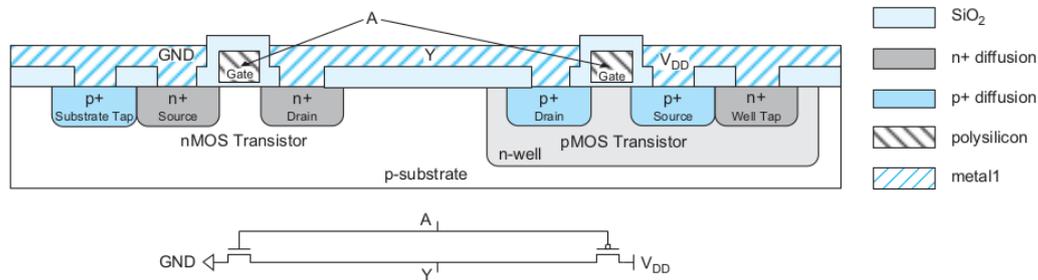


Figura 2.5: Sección transversal de un inversor CMOS [10].

Entender los pasos del proceso de fabricación CMOS es fundamental para comprender las reglas de diseño y las limitaciones de los dispositivos. Cuanto mayor sea el grado de entendimiento de éste, el diseñador podrá aprovechar mejor las herramientas disponibles. En esta sección, se discutirán los puntos más importantes de un proceso de fabricación CMOS estándar, desde la obtención del silicio, hasta el encapsulado final de los dies.

#### 2.2.1.1. Obtención del wafer

La fabricación de circuitos integrados requiere silicio monocristalino de muy alta pureza para confeccionar las obleas o *wafers*. A su vez, el corte de

éstas es un proceso complejo, donde se deben cortar wafers de 10 a 30cm de diámetro y 1mm de espesor, desde el lingote monocristalino de silicio. Dicho lingote se obtiene mediante el proceso *Czochralski*, llamado así en honor al químico polaco Jan Czochralski. Este método es utilizado para obtener silicio monocristalino mediante un cristal semilla [11].

La semilla, un monocristal de Si, se introduce en un crisol de cuarzo con silicio fundido. Después de la siembra, se cultiva el cristal cilíndrico a partir de una masa fundida de silicio, el cual tendrá la misma estructura cristalina que la semilla inicial. Utilizando un calentador óhmico se irradia energía hacia el crisol o cuba, mientras que a su vez el cristal irradia calor hacia la capa exterior, la cual es refrigerada por agua, según se observa en la Figura 2.6a.

Las interacciones entre las fases líquida y sólida se llevan a cabo a lo largo del frente de solidificación, cuya forma está directamente relacionada con el equilibrio de transferencia de calor en sus proximidades. Durante el crecimiento, el calor liberado por la solidificación debe ser evacuado por conducción a través del cuerpo de cristal y radiación de su superficie. Como consecuencia, el frente de solidificación es básicamente cóncavo, con respecto al cristal, a altas tasas de extracción, y convexa en las bajas tasas de extracción [12].

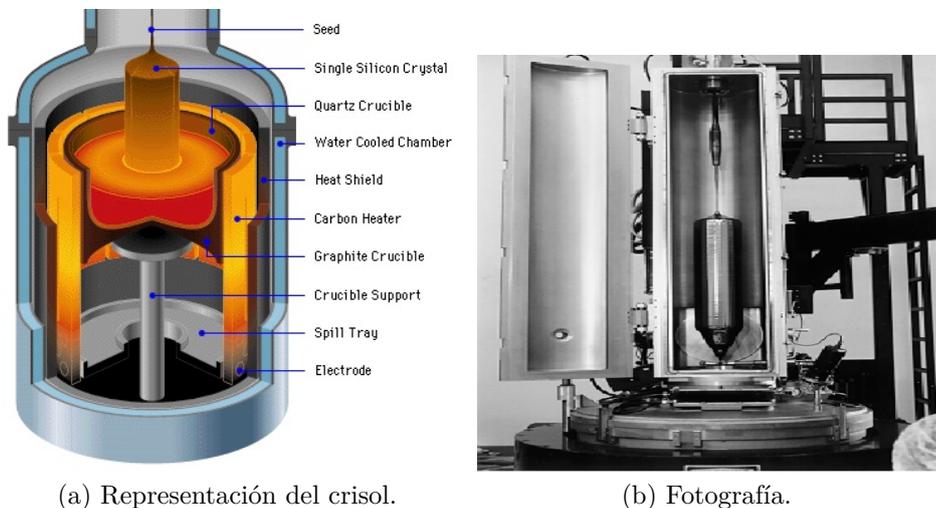


Figura 2.6: Proceso Czochralski: obtención del wafer [13].

### 2.2.1.2. Oxidación

Esta es una parte vital del proceso, ya que la calidad del óxido del gate y su correcta continuación de la red periódica cristalina del Si son fundamentales

para el funcionamiento de los dispositivos MOS. Además, se deben recubrir las zonas sin utilizar con aislantes, para evitar contactos eléctricos indeseados. Se emplean dos diferentes técnicas para obtener  $\text{SiO}_2$ , una de mayor calidad y menor velocidad, y la otra de menor calidad y mayor velocidad.

En la Figura 2.7 se muestra el espesor del óxido (en micrones) en función de la duración del proceso de oxidación (en horas), para 1000 y 1200°C. Se diferencian los procesos húmedos (en líneas continuas) y los procesos secos (en líneas punteadas).

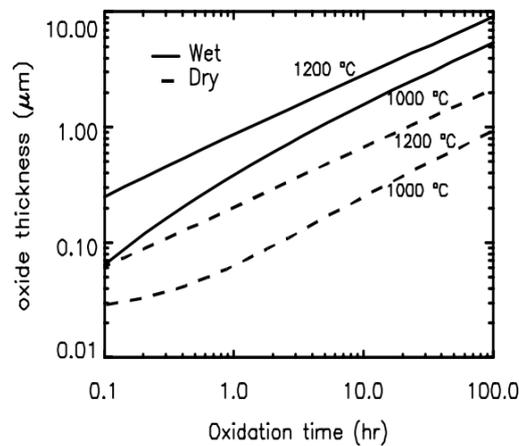
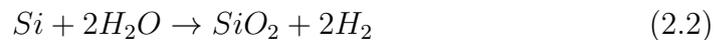


Figura 2.7: Espesor de la oxidación en función del tiempo [13].

La oxidación del silicio se consigue mediante el calentamiento de obleas de silicio en una atmósfera oxidante. Los siguientes son algunos de los enfoques comunes:

### Húmeda



Cuando la atmósfera oxidante contiene vapor de agua. La temperatura varía por lo general entre 900°C y 1000°C. Este proceso también se llama oxidación pirogénica cuando se utiliza una mezcla con relación 2:1 de hidrógeno y oxígeno. Oxidación húmeda es un proceso rápido. Produce óxidos de menor calidad y se utiliza para crecer el material aislante, llamado *field oxide* (óxido de campo).

### Seca



Cuando la atmósfera oxidante es oxígeno puro. La temperatura utilizada ronda los 1200°C para lograr una tasa de crecimiento aceptable. El proceso de oxidación en seco forma una mejor calidad que el óxido de oxidación húmeda. Se utiliza para formar, óxidos delgados de compuerta.

### 2.2.1.3. Etching (remoción)

El *etching*, es un proceso químico por el cual se remueven los materiales barrera. Existen tres tipos: wet etching, dry etching y reactive-ion etching. Una vez que un material se ha depositado, el etching se utiliza para formar selectivamente patrones tales como líneas de conexión y los agujeros de contacto.

**Dirección** Las remociones pueden ser isotrópicas o anisotrópicas, dependiendo del agente, de la dirección y el perfil del ataque. Para determinar estas características, se define el factor  $R_L$  como *Lateral Etch Ratio* como

$$R_L = \frac{\text{HorizontalEtchRate}}{\text{VerticalEtchRate}} = \frac{r_H}{r_L} \quad (2.4)$$

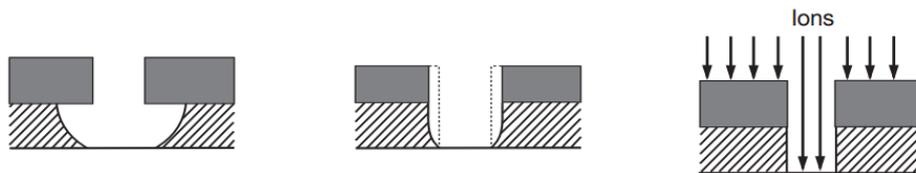
Para los perfiles isotrópicos, el lateral etch ratio es igual para las direcciones verticales y horizontales, es decir  $R_L = 1$ . Para los perfiles anisotrópicos, las dimensiones son diferentes, de la forma  $0 < R_L < 1$  [14].

**Wet etching** El material se elimina selectivamente de las áreas de la oblea que no están cubiertos por la resina fotosensible. Esto se logra mediante el uso de muchos tipos diferentes de ácidos, bases o soluciones cáusticas. Gran parte del trabajo con productos químicos tiene lugar en grandes bancos húmedos donde se preparan soluciones especiales para estas tareas. Debido a la peligrosidad de algunos de estos disolventes, la seguridad e impacto ambiental son preocupaciones primordiales. El proceso de wet etching hace uso tanto de ácidos como de soluciones básicas. Por ejemplo, el ácido fluorhídrico *buffered* con fluoruro de amonio se utiliza típicamente para grabar  $\text{SiO}_2$ . Este método es el más utilizado y el más antiguo [15].

Los procesos de wet etching generan *undercutting*, lo que resulta en un perfil de remoción isotrópico, donde la tasa de ataque vertical es aproximadamente igual a la velocidad de ataque horizontal (Figura 2.8a). Como fabricantes de semiconductores siguen disminuyendo tamaños de características, esta subvaloración se hace más intolerable, haciendo grabado húmedo una técnica menos deseable para la eliminación de material [17].

**Dry etching** Es uno de los procesos más utilizados en la fabricación de semiconductores. Los procesos secos utilizan plasma generado por radicales libres para eliminar el material sólo en el área marcada por el patrón fotorresistente, creado durante la etapa de fotolitografía. Sólo la remoción seca proporciona la capacidad de controlar a través de ataque químico anisotrópico iones no reactivos y reactivos de plasma [17]. En la Figura 2.8b se muestra un esquema del perfil del ataque dry etching.

**RIE** Acrónimo de Reactive-ion etching, y es un tipo particular de dry etching. En los procesos VLSI modernos se evita atacar con ácido, y se utiliza RIE en su lugar. Una oblea se coloca en la cámara del equipo de etching y se le aplica una carga eléctrica negativa. La cámara se calienta a  $100^{\circ}\text{C}$  y se lleva a un nivel de vacío de 10 militorrs. Luego se llena con plasma cargado positivamente (por lo general una mezcla de nitrógeno, cloro y tricloruro de boro). Las cargas eléctricas opuestas hacen que las moléculas de plasma se mueven rápidamente y se alinean en una dirección vertical, atacando el material expuesto, el cual es removido. RIE tiene la ventaja de ofrecer una direccionalidad bien definida a la acción de ataque químico, resultando en de patrones con contornos verticales [15]. En la Figura 2.8c se muestra un esquema del perfil del ataque dry etching.



(a) Remoción isotrópica. (b) Perfil de remoción: Dry etching. (c) RIE: etching anisotrópico.

Figura 2.8: Perfiles del proceso de etching [17].

El plasma es un gas ionizado, formado por la aplicación una de fuerte señal de radiofrecuencia en condiciones de vacío. El plasma genera electrones, iones, y radicales libres que son, químicamente, especies reactivas, tales como flúor o cloro. Estos iones reactivos tienen energía de desprendimiento de material, así, los perfiles de remoción dependen de la densidad de iones y su energía [17].

#### 2.2.1.4. Fotolitografía

La fotolitografía es el proceso de transferencia de patrones geométricos, dados por una máscara, sobre una capa fina de material fotosensitivo, llamado *photoresist*, con el cual se cubre la superficie la oblea de Si. La transferencia de patrones es siempre acompañada por un proceso de remoción (etching), para remover selectivamente óxido de silicio o metal.

**Photoresist** El *photoresist* es un compuesto sensible a radiación UV. La exposición a esta radiación le produce un cambio en sus propiedades de solubilidad, que puede ser clasificado como positivo o negativo. Para los positivos, las regiones expuestas se vuelven más solubles y son más fácilmente removibles. El resultado es que en el photoresist positivo se presenta el mismo patrón que en la máscara. Para los negativos, las regiones expuestas se vuelven menos solubles, y los patrones formados son inversos a los patrones de las máscaras.

**Máscaras** Una máscara es una placa opaca con agujeros o transparencias que permiten que la luz pase a través de un patrón definido. Están fabricadas en cuarzo, el cual es un mineral derivado del sílice ( $\text{SiO}_2$ ), y se diseñan para diferentes longitudes de onda, dependiendo del proceso.

El diseño de circuitos integrados mediante sistemas EDA se basa en el diseño de las máscaras que serán utilizadas en el proceso. Es importante que estén hechas de un material poco sensible a la dilatación, tengan una baja absorción de luz UV y sean limpiadas periódicamente. Para evitar errores en la transferencia del patrón se debe alinear la máscara. Existen distintos tipos de alineamiento:

- Por contacto: la máscara se deteriora
- Por proximidad: limitada por la difracción de la luz
- Por proyección: método utilizado hoy en día. Además, se realiza húmedo, para disminuir la difracción

En la Figura 2.9 se presentan los pasos de fabricación genéricos necesarios para transferir un patrón de  $\text{SiO}_2$ . Se parte de la oblea, en donde se crece la capa de óxido, que se lo recubre con material fotoresistivo. Luego se alinea la máscara y se expone el wafer a radiación UV, lo cual vuelve más soluble la sección descubierta de material fotoresistivo. Mediante algún proceso de

remoción, se elimina el dióxido de silicio expuesto, resultando en el patrón geométrico deseado. Finalmente se quitan los restos del photoresist.

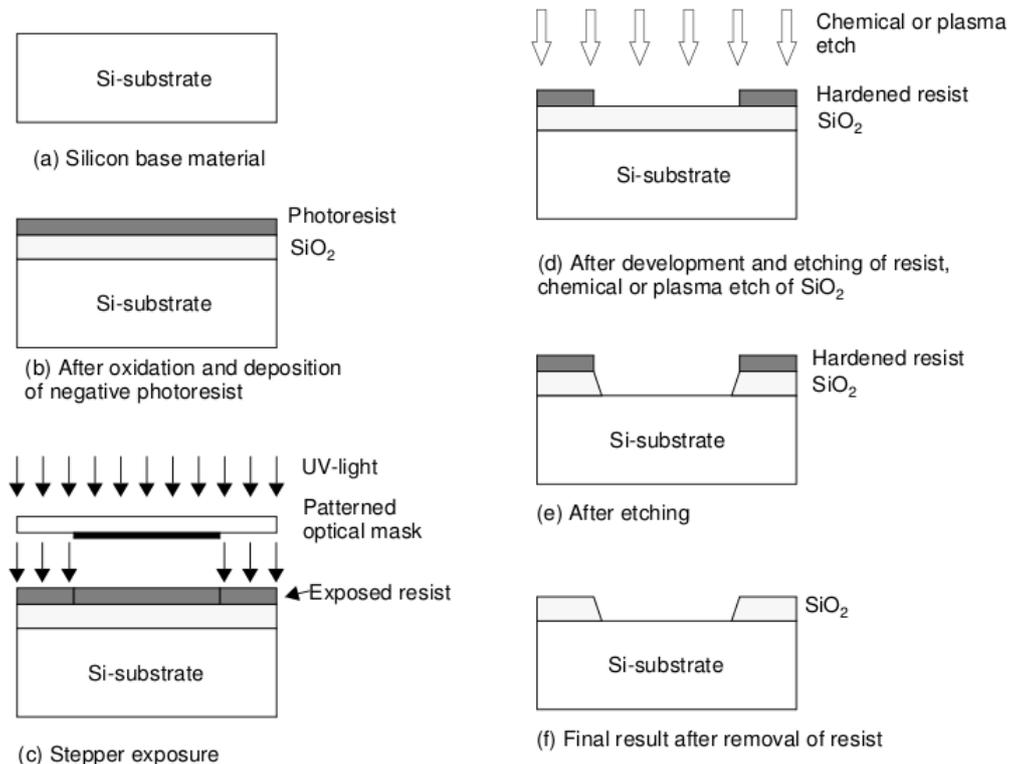


Figura 2.9: Pasos del proceso de transferencia del patrón de  $\text{SiO}_2$  [15].

### 2.2.1.5. Dopaje

Existen dos técnicas de dopaje, difusión e implantación iónica. En ambas, el área que será dopada es expuesta, mientras que el resto del wafer es recubierto, generalmente con dióxido de silicio.

**Difusión** Las obleas se colocan en un tubo de cuarzo incrustado en un horno caliente y se introduce un gas que contiene el dopante. Las altas temperaturas del horno, típicamente 900 a 1100°C, hacen que los agentes de dopado se difundan en la superficie expuesta, tanto vertical como horizontalmente. La concentración de dopante final es la mayor en la superficie, y disminuye con un perfil gaussiano más profundamente en el material.

**Implantación iónica** Este sistema de implantación dirige un haz de iones sobre la superficie del semiconductor. La aceleración de los iones determina la profundidad a la que penetran en el material, mientras que la corriente del haz y el tiempo de exposición determina la dosis. La implantación de iones permite un control independiente de la profundidad y la dosis.

**Deposición de films - Metalizaciones** Los procesos CMOS requieren la deposición repetitiva de capas de un material sobre la oblea, ya sea para actuar como *buffers* para alguna etapa del proceso, como aislante o como capas conductoras. Más allá del proceso de oxidación, que permite cultivar una capa de  $\text{SiO}_2$ , otros materiales requieren diferentes técnicas. Por ejemplo, el nitruro de silicio ( $\text{Si}_3\text{N}_4$ ) se utiliza como un material auxiliar durante la formación del *field oxide*. Este componente se deposita en todas partes mediante un proceso llamado deposición química de vapor (CVD, por sus siglas en inglés *chemical vapor deposition*), que utiliza una reacción en fase gaseosa a una temperatura alrededor de  $850^\circ\text{C}$ . Por otro lado, el polisilicio se deposita mediante un procedimiento químico, en el que se hace fluir gas silano (hidruro de silicio,  $\text{SiH}_4$ ) sobre la oblea recubierta con  $\text{SiO}_2$  caliente, a una temperatura de aproximadamente  $650^\circ\text{C}$ . La reacción resultante produce un material amorfo, llamado polisilicio. Para aumentar la conductividad del material, el depósito es seguido por una etapa de implantación.

Las capas de interconexión de aluminio se generan típicamente usando un proceso conocido como *sputtering* o pulverización catódica. El aluminio se evapora en vacío, con el calor entregado por el bombardeo de iones o electrones. Otros materiales de interconexión metálicos como el cobre requieren diferentes técnicas de deposición [15].

Luego del dopaje, el proceso de metalización se usa para formar los contactos óhmicos e interconexiones. Las capas metálicas se pueden lograr mediante aplicaciones de vapor químico o vapor físico.

### 2.2.2. Reglas de Diseño

Las restricciones propias del proceso, impuestas a la geometría de un diseño, con el fin de garantizar la factibilidad de fabricación de cierto circuito integrado con un rendimiento esperable, se denominan **reglas de diseño**. Estas reglas son, primeramente, un juego de instrucciones escritas en un lenguaje informal, que luego se formalizan en un juego de archivos, cuyo lenguaje depende de cada herramienta EDA de verificación. Si el diseñador se adhiere a estas normas, obtiene la garantía de que su circuito será manufacturable.

Por lo general, las reglas de diseño se establecen cuando se está generando un nuevo proceso, o cuando un proceso se actualiza de una generación a la siguiente. El establecimiento de nuevas reglas de diseño es normalmente un trabajo en conjunto entre ingenieros diseñadores VLSI e ingenieros de proceso. Los diseñadores de circuitos precisan reglas de diseño con dimensiones más pequeñas y más estrictas, para mejorar el rendimiento y disminuir el área del chip, mientras que los ingenieros de proceso intentan producir reglas de diseño que concluyan en procesos de fabricación controlables y reproducibles. El resultado es un conjunto de reglas de diseño que produce un circuito competitivo diseñado y fabricado de una manera rentable. Se pretende que las reglas de diseño sean simples, constantes en el tiempo, aplicables a varios procesos y estandarizables entre varias fabricas.

Se clasifican las reglas en tres grandes grupos:

- **Reglas Tamaño (size rules):** El tamaño de la característica mínima de un dispositivo o una línea de interconexión se determina por la capacidad de los equipos de fotolitografía utilizados en el proceso de fabricación. La regla de diseño debe especificar los tamaños mínimos para diferentes capas, dependiendo del material y el proceso.
- **Reglas de separación (spacing):** Los caminos de interconexión, sean de un mismo o diferente material deben tener una cierta separación mínima entre sí. En general, la separación entre líneas es similar al tamaño mínimo, para poder lograr así una buena densidad de interconexión. La mayoría de los procesos tienen reglas de espaciado para cada capa.
- **Reglas de solapamiento (overlap):** El diseño físico de transistores MOS, exige el solapamiento de una capa de silicio policristalino sobre una zona activa. Así, las reglas de solapamiento son muy importantes para la formación de los transistores y contactos.

## 2.3. Circuitos digitales CMOS

Los circuitos digitales CMOS son estructuras basadas en transistores MOS que trabajan de forma complementaria. Su principal ventaja en comparación con otras familias lógicas es su robustez (baja sensibilidad al ruido). A su vez tienen muy buen rendimiento y bajo consumo de energía estática. Sus márgenes de ruido están cerca de  $\frac{V_{DD}}{2}$ , como en la definición de familia lógica ideal.

Como definición fundamental, las celdas CMOS tienen una red denominada *pull-down*, constituida por transistores del tipo nMOS, la cual impone un “**cero fuerte**”, generando un camino de baja impedancia entre tierra (GND) y la salida; y una red constituida por transistores del tipo pMOS, denominada *pull-up*, que impone un “**uno fuerte**”, generando un camino de baja impedancia entre la alimentación (VDD) y la salida, como se detalla en la Figura 2.10. Las redes están dispuestas de tal manera que nunca se genera un camino de baja impedancia entre VDD y GND, es decir, trabajan de forma **complementaria**.

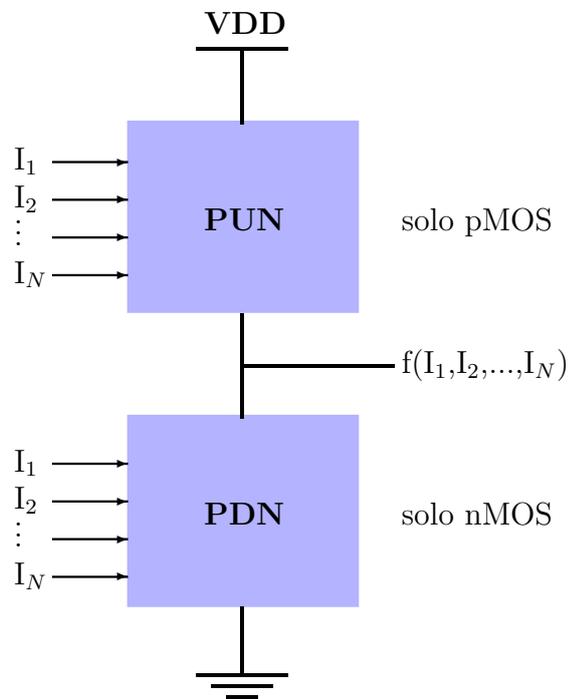


Figura 2.10: CMOS: PUN + PDN.

### 2.3.1. El transistor MOS como llave

Se puede simplificar el comportamiento del transistor MOS reduciendo su funcionamiento a solo dos regímenes de operación, llamados “corte” y “saturación”:

1. Cuando  $|V_{GS}| > |V_T|$  se produce la inversión del semiconductor en

cercanías de la interfaz Si-SiO<sub>2</sub>, es decir se *forma canal* de conducción entre drain y source.

2. Cuando  $|V_{GS}| < |V_T|$  no se forma el canal y no hay conducción entre source y drain.

Puede observarse que este comportamiento es análogo al de una llave. Dependiendo del terminal de control, se comporta como un elemento conductivo o como un circuito abierto. En este caso, actúan como llaves controladas por tensión, donde el terminal de control corresponde al gate. De ésta forma, el gate de los transistores MOS es la entrada de señal digital, el cual controla el estado de la llave. Luego, se diseñan las diferentes celdas estándar pensando en analogía a la lógica con llaves. Al tener dos llaves en serie, es necesario que ambas llaves estén cerradas para que haya conducción entre los extremos de la misma (entre drain y source para un transistor MOS). Se observa en las Figuras 2.11a y 2.11b como actúan los transistores nMOS y pMOS. Los primeros se encienden con un uno lógico y los segundos un cero lógico. Entonces, de la Figura 2.11a se desprende que

$$\mathbf{a} = \mathbf{b} \text{ if } (\mathbf{g1} \ \& \ \mathbf{g2}) \quad (2.5)$$

mientras, de la Figura 2.11b se infiere que

$$\mathbf{a} = \mathbf{b} \text{ if } (\overline{\mathbf{g1}} \ \& \ \overline{\mathbf{g2}}) \quad (2.6)$$

Si ahora conectamos ambas llaves en paralelo, basta con que solo una de ellas conduzca para establecer un camino entre a y b. Lo cual se observa en las Figuras 2.11c y 2.11d. De la primera, es claro entonces que

$$\mathbf{a} = \mathbf{b} \text{ if } (\mathbf{g1} \ | \ \mathbf{g2}) \quad (2.7)$$

mientras que en la Figura 2.11d se observa que

$$\mathbf{a} = \mathbf{b} \text{ if } (\overline{\mathbf{g1}} \ | \ \overline{\mathbf{g2}}) \quad (2.8)$$

### 2.3.2. El inversor CMOS

El inversor es el circuito digital más elemental. El comportamiento eléctrico de las compuertas que conforman circuitos complejos puede ser aproximado mediante la extrapolación de los resultados obtenidos para los inversores.

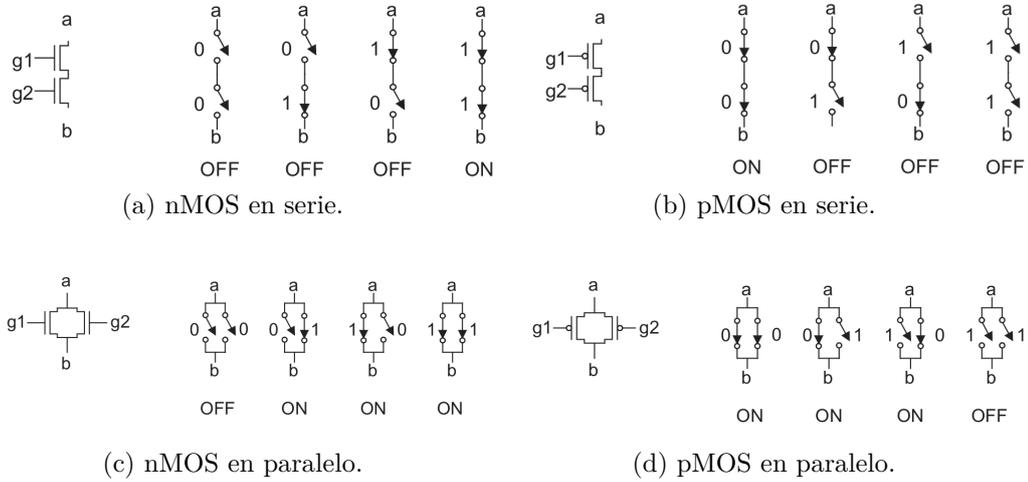


Figura 2.11: El transistor como llave [10].

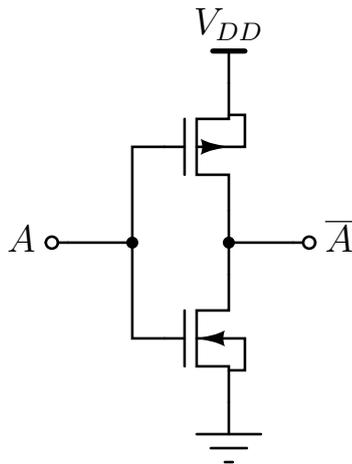


Figura 2.12: Esquemático de un inversor CMOS.

### 2.3.2.1. Análisis estático

La Figura 2.12 muestra el diagrama esquemático de un inversor CMOS estático. Su funcionamiento se comprende fácilmente con la ayuda del modelo simplificado del transistor MOS pensado como llave. Así, cuando  $V_{IN}$  se encuentra en un estado alto, e igual a  $V_{DD}$ , el transistor nMOS está encendido, mientras que el transistor pMOS se encuentra apagado. Se genera un camino de baja impedancia entre  $V_{OUT}$  y el nodo de tierra, resultando en un valor de estado estacionario a la salida de 0V.

Por otro lado, cuando la tensión de entrada es baja e igual a GND, el transistor nMOS se encuentra en alta impedancia y el transistor pMOS se encuentra conduciendo. Existe un camino de baja impedancia entre  $V_{DD}$  y  $V_{OUT}$ , produciendo una tensión de salida alta, es decir, un ‘1’ lógico.

Se desprenden una serie de propiedades importantes de la lógica CMOS estática desde éste análisis del inversor nivel de modelo de llaves del transistor MOS:

- Los niveles altos y bajos de salida son  $V_{DD}$  y GND, respectivamente. Así, la variación de voltaje es igual a la tensión de alimentación, traduciéndose en altos márgenes de ruido.
- Los niveles lógicos no dependen de los tamaños relativos de los dispositivos, de manera que los transistores pueden ser de tamaño mínimo. Ésta propiedad se denomina *ratioless*, y está en contraste con la lógica *ratioed*, donde los niveles lógicos están determinadas por las dimensiones relativas de los transistores.
- En estado estacionario, siempre existe un camino con una resistencia finita entre la salida y  $V_{DD}$  o GND. El inversor CMOS, tiene una baja impedancia de salida, logrando baja sensibilidad al ruido y perturbaciones.
- La resistencia de entrada del inversor CMOS es extremadamente alta, ya que el gate de un transistor MOS está aislado y no consume corriente continua de entrada. Un solo inversor puede conducir, teóricamente, un número infinito de compuertas (es decir *fan-out*  $\rightarrow \infty$ ) y seguir siendo funcionalmente operativo. Sin embargo, aumentar el fan-out de salida también aumenta el retardo de propagación. Así es que, aunque el fan-out de salida no tiene ningún efecto sobre el comportamiento en estado estacionario en la lógica CMOS, si tiene injerencia en la velocidad de operación.
- No existe un camino directo entre los rails de alimentación y de tierra, en condiciones en estado estacionario, es decir, cuando la entrada y la salida se mantienen constantes. La ausencia de flujo de corriente, ignorando las corrientes de fuga, indica que la compuerta no consume energía estática, lo cual es una de las más importantes propiedades de la familia CMOS<sup>1</sup>.

---

<sup>1</sup>En los nodos de fabricación actuales, la propiedad de no disipación de potencia estática no es aplicable, ya que las corrientes de fuga son del orden de magnitud de las corrientes

### 2.3.2.2. Transferencia

Se toma al inversor como una caja negra, en donde se coloca una señal de entrada  $in$  y produce una salida  $out$ . La respuesta eléctrica se observa en su función transferencia (VTC, del inglés *voltage-transfer characteristic*), en donde se grafica la tensión de salida en función de la tensión de entrada:

$$V_{out} = f(V_{in}) \quad (2.9)$$

La curva de transferencia estática de un inversor CMOS se presenta en la Figura 2.13. Posee una zona de transición muy estrecha, resultado de la alta ganancia del mismo. En esa región de operación, un pequeño cambio en las tensión de entrada genera una variación grande de la salida.

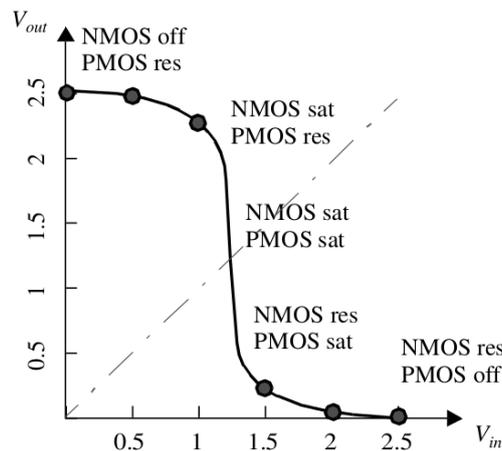


Figura 2.13: Curva de transferencia del inversor CMOS [15].

### 2.3.2.3. Márgenes de ruido

Para que una compuerta lógica sea robusta, es necesario que las zonas permitidas de los valores lógicos sean lo más amplio posible. Para cuantificar la sensibilidad de una compuerta al ruido, se definen los márgenes de ruido NM (del inglés *noise margin*), tanto para el nivel bajo  $NM_L$ , como para el nivel alto  $NM_H$ :

$$NM_L = V_{IL} - V_{OL} \quad (2.10)$$

propias de los dispositivos. De todas formas, existen contados esfuerzos para lograr reducir la potencia estática en nodos sub-45nm, como es la introducción de dispositivos *Tri-gate* o FinFET [16].

$$NM_H = V_{OH} - V_{IH} \quad (2.11)$$

donde  $V_{IL}$  es la mínima tensión de entrada en estado bajo;  $V_{IH}$  es la máxima tensión de entrada en estado alto;  $V_{OL}$  es la máxima tensión de salida en estado bajo y  $V_{OH}$  es la mínima tensión de salida en estado alto.

Se definen los márgenes de ruido para el inversor CMOS como los puntos de polarización donde:

$$\frac{dV_{OUT}}{dV_{IN}} = -1. \quad (2.12)$$

Estos son los puntos donde la ganancia  $A_v$  del amplificador, formado por el inversor, es igual a -1. Si bien es posible derivar expresiones analíticas para  $V_{IH}$  y  $V_{IL}$ , estos tienden a ser más complejos. Un enfoque más sencillo es utilizar una aproximación lineal por tramos para la transferencia del inversor. La región de transición es aproximada por una línea recta, cuya ganancia es igual a la ganancia  $A_v$  en el umbral de conmutación  $V_M$ , el cual se ubica en  $\frac{V_{DD}}{2}$ . La intersección de  $V_{OH}$  con la recta VTC y las líneas  $V_{OL}$  se utilizan para definir los puntos de  $V_{IH}$  y del  $V_{IL}$  respectivamente. El error introducido es pequeño y sirve como modelo de primera aproximación. Todo esto se detalla en la Figura 2.14.

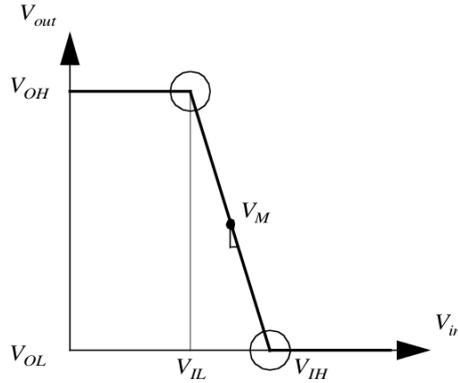


Figura 2.14: Transferencia del inversor CMOS linealizada por tramos [15].

De la Figura 2.14 se observa que

$$NM_L = (V_M - \delta) - 0 = V_M - \delta = \frac{V_{DD}}{2} - \delta \quad (2.13)$$

$$NM_H = V_{DD} - (V_M + \delta) = V_{DD} - \frac{V_{DD}}{2} - \delta = \frac{V_{DD}}{2} - \delta$$

donde  $\delta$  representa la pendiente de la VTC, es decir, la ganancia del inversor. Al aumentar la ganancia, la transferencia y los márgenes de ruido del inversor CMOS tienden a los valores de la familia lógica ideal.

### 2.3.2.4. Análisis dinámico

El retardo de propagación del inversor CMOS se determina por el tiempo que tarda cargar y descargar la capacidad de carga equivalente de salida  $C_L$  a través de los transistores pMOS y nMOS, respectivamente. Esta observación sugiere que conseguir una capacidad  $C_L$  lo más pequeña posible es crucial para el diseño de circuitos CMOS de alto rendimiento.

La capacidad  $C_L$  incluye todas las capacidad parásitas del nodo de salida: capacidades de juntura de las difusiones, capacidades de las interconexiones y de la entrada del inversor de carga.

La Figura 2.15 muestra el esquemático de dos inversores en cascada. Incluye todas las capacitancias que influyen en la respuesta transitoria del nodo  $V_{OUT}$ . Se asume inicialmente que la entrada  $V_{IN}$  es excitada por una fuente de tensión ideal. El detalles de las capacidades parásitas se explica en el Apéndice C.

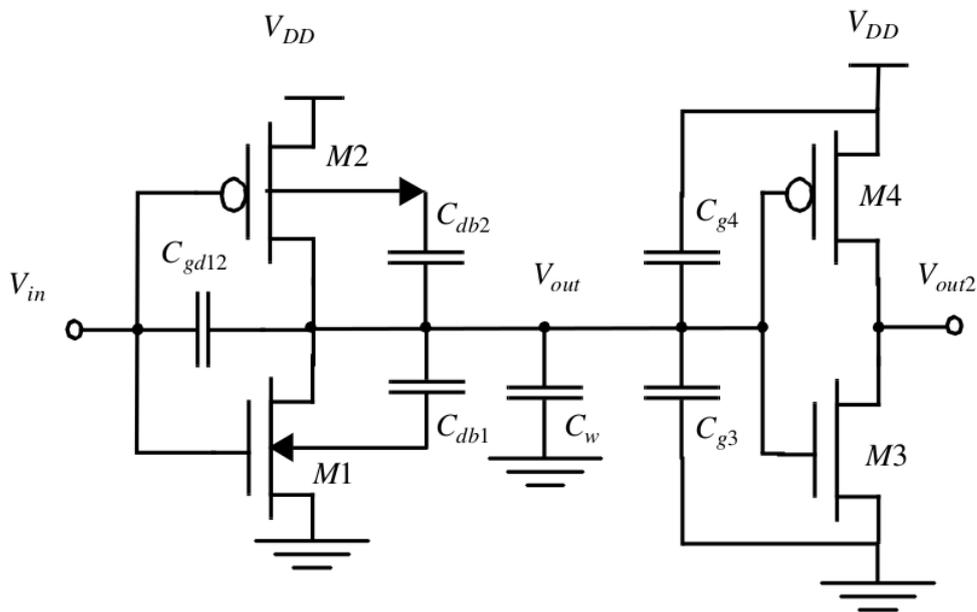


Figura 2.15: Capacidades parásitas. Influyen en el comportamiento transitorio del par inversor en cascada [15].

**Retardo de propagación: análisis de primer orden** Una forma de calcular el retardo de propagación del inversor es integrando la corriente

de carga (o descarga) del capacitor  $C_L$ . Esto resulta en la expresión de la Ecuación 2.14.

$$t_p = \int_{v_1}^{v_2} \frac{C_L(v)}{i(v)} dv \quad (2.14)$$

siendo  $i(v)$  la corriente de carga/descarga,  $v$  la tensión sobre el capacitor y  $v_1$  y  $v_2$  las condiciones iniciales y finales. Siendo que tanto  $C_L(v)$  como  $i(v)$  son funciones no lineales de  $v$ . Para simplificar el cálculo, se utiliza el modelo de llave del transistor MOS para derivar una aproximación razonable del retardo de propagación. La resistencia promedio de conducción del transistor MOS se presenta en la Ecuación 2.15.

$$R_{eq} = \frac{1}{\frac{V_{DD}}{2}} \int_{V_{DD}/2}^{V_{DD}} \frac{v}{I_{DSAT}(1 + \lambda v)} dv \approx \frac{3}{4} \frac{V_{DD}}{I_{DSAT}} \left(1 - \frac{7}{9} \lambda V_{DD}\right) \quad (2.15)$$

con

$$I_{DSAT} = k \frac{W}{L} \left( (V_{DD} - V_T) V_{DSAT} - \frac{V_{DSAT}^2}{2} \right) \quad (2.16)$$

Los límites de integración se establecen en  $V_{DD}$  y  $\frac{V_{DD}}{2}$  ya que el tiempo de propagación para la familia lógica CMOS se define como el tiempo máximo desde que la señal de entrada cruza el 50% hasta al cruce del 50% de la señal de salida, como se muestra en la Figura 2.16.

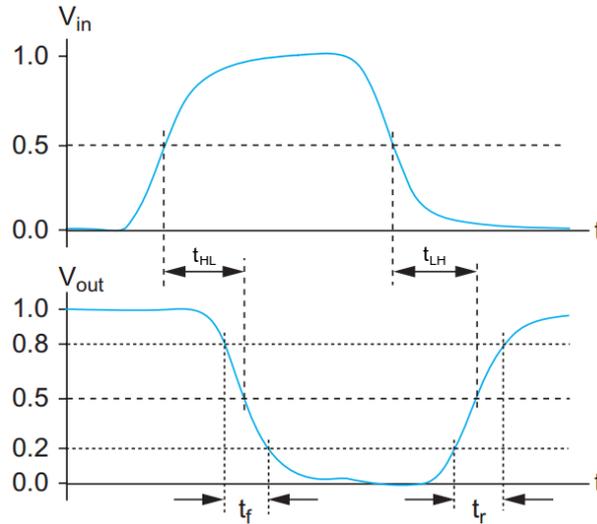


Figura 2.16: Tiempo de propagación y tiempos de rise y fall [10].

Derivar el retardo de propagación del circuito resultante resulta más sencillo. El análisis ahora es el mismo al de una red RC lineal de primer orden.

El retardo de propagación de dicha red para una escalón de voltaje en la entrada es proporcional a la constante de tiempo de la red resultando, formada por la resistencia pull-down y capacidad de carga. Por lo tanto,

$$t_{pHL} = \ln(2)R_{eqn}C_L \approx 0,69 \times R_{eqn}C_L \quad (2.17)$$

De la misma forma, el retardo de propagación para la transición de bajo a alto

$$t_{pLH} \approx 0,69 \times R_{eqp}C_L \quad (2.18)$$

asumiendo que la capacidad de carga es igual para ambas transiciones. En general, el retardo de propagación será:

$$tp = \frac{t_{pHL} + t_{pLH}}{2} \approx 0,69C_L \left( \frac{R_{eqn} + R_{eqp}}{2} \right) \quad (2.19)$$

Generalmente, es deseable que una compuerta tenga de retardos de propagación similares para ambas transiciones. Esta condición se puede lograr haciendo que las resistencias de conducción de ambos transistores sean aproximadamente igual. Esto es similar a decir que ambos transistores tengan la misma capacidad de conducción de corriente [15].

### 2.3.3. Lógica combinacional

Se denomina sistema o lógica combinacional a todo sistema digital en el que sus salidas son solo función de los valores de entrada actuales, sin que intervengan en ningún caso estados anteriores de las entradas o de las salidas. Por tanto, carecen de memoria o realimentación. La lógica combinacional está formada por ecuaciones simples a partir de las operaciones básicas del álgebra booleana. Ejemplos de éste tipo de circuitos son las compuertas lógicas, multiplexores, contadores, comparadores, unidades lógicas aritméticas (ALU), etc. Es posible extender el modelo estudiado del inversor mínimo para abordar la síntesis de compuertas digitales de diferente índole, como por ejemplo NOR, NAND y XOR.

## 2.4. Síntesis de compuertas

Para sintetizar compuertas de lógica combinacional CMOS, se considera al transistor MOS como una llave controlada por tensión, como se demostró en la Sección 2.3.1. Se definen dos grupos de complementariedad, los grupos

productos y los grupos sumas, de forma de diseñar las redes de pull-up y pull-down. El primer grupo está compuesto por transistores del tipo nMOS en serie y pMOS en paralelo. Si tomamos la Ecuación 2.5

$$\mathbf{a} = \mathbf{b} \text{ if } (\mathbf{g1} \cdot \mathbf{g2}) \quad (2.20)$$

y si se complementa dos veces y se aplica la ley de De Morgan a la condición de la Ecuación 2.8

$$\overline{\overline{\mathbf{g1} + \mathbf{g2}}} = \overline{\overline{\mathbf{g1}} \cdot \overline{\mathbf{g2}}} = \overline{\mathbf{g1} \cdot \mathbf{g2}} \quad (2.21)$$

se demuestra que los transistores nMOS en serie y los transistores pMOS en paralelo pertenecen al grupo productos, es decir, representan una función AND.

Del modo contrario, si complementamos y aplicamos De Morgan a la condición de la Ecuación 2.6

$$\overline{\overline{\mathbf{g1} \cdot \mathbf{g2}}} = \overline{\overline{\mathbf{g1}} + \overline{\mathbf{g2}}} = \overline{\mathbf{g1} + \mathbf{g2}} \quad (2.22)$$

y se toma la Ecuación 2.7

$$\mathbf{a} = \mathbf{b} \text{ if } (\mathbf{g1} + \mathbf{g2}) \quad (2.23)$$

se demuestra que los transistores nMOS en paralelo y los transistores pMOS en serie pertenecen al grupo sumas, es decir, representan una función OR.

### 2.4.1. Algoritmo para sintetizar funciones lógicas

Se presenta entonces un algoritmo para sintetizar funciones lógicas, el cual se deriva del modelo del transistor como llave y el concepto de grupo suma y grupo producto para elaborar las redes de pull-up y pull-down. Así, las celdas básicas<sup>2</sup> se diseñaron siguiendo el algoritmo.

1. Escribir la función lógica de forma:

$$F(X) = \text{not}[\text{logic}(X)] \quad (2.24)$$

Donde  $\text{logic}(X)$  debe ser una combinación de sumas y productos del conjunto de variables de entrada

---

<sup>2</sup>Definimos celdas básicas a las funciones booleanas o combinaciones de éstas, resultando siempre ser celdas combinacionales. Para sintetizar celdas secuenciales se utilizan otros criterios, así como para sintetizar celdas con diferentes características, como aquellas que tengan salida *tri-state*.

2. Síntetizar la red de *pull-down* a través de  $logic(X)$  considerando que los productos de dicha expresión corresponden a transistores nMOS en configuración serie y que las sumas corresponden a transistores nMOS en configuración paralelo.
3. Síntetizar la red de *pull-up* considerando que es complementaria a la red de *pull-down*, es decir que los productos de variables en  $logic(X)$  corresponden a transistores pMOS en configuración paralelo y las sumas a transistores pMOS en serie.
4. Conectar la red *pull-down* entre GND y el nodo de salida.
5. Conectar la red *pull-up* entre VDD y el nodo de salida.
6. Asignar a cada gate de cada transistor una entrada del sistema.

## 2.4.2. Lógica Secuencial

Todos los sistemas digitales requieren almacenamiento de información de estado, lo que lleva a otra clase de sistemas llamados circuitos lógicos secuenciales. En estos circuitos, la salida no depende sólo de los valores actuales de las entradas, sino que también de los valores de entrada pasados. Es decir, un circuito secuencial recuerda algunos de los estados pasados del sistema. Así, se dice que los circuitos secuenciales tienen memoria [15]. Un circuito secuencial incluye una parte de lógica combinacional y un módulo que contiene el estado. Entre los circuitos secuenciales destacamos los latches y flip-flops.

### 2.4.2.1. Latches

Un latch es un circuito biestable asíncrono usado para almacenar información en sistemas lógicos digitales. Puede almacenar un bit de información, asimismo los latches se pueden agrupar de tal manera que logren almacenar más de 1 bit. No tienen entrada de reloj, sino que son activados por nivel. La salida es función del estado presente en las entradas y de los estados previos en las salidas.

### 2.4.2.2. Flip-flops

Un flip-flop es un circuito lógico secuencial biestable capaz de permanecer en uno de sus dos estados durante un tiempo determinado. Es un circuito sincrónico, es decir, cambia de estado solo en los flancos de reloj.

Un flip-flop universal, con un buen rendimiento, menor consumo de energía y mayor robustez sería un componente ideal para ser incluido en una librería de celdas estándar. Sin embargo, como se muestra en [19], el aumento del rendimiento de los flip-flops generalmente implica una relación de compromiso con el consumo de potencia y la robustez del sistema. Por lo tanto, un conjunto de diferentes latches y flip-flops con diferentes rendimientos, son esenciales para limitar el consumo de energía y brindar inmunidad al ruido, según la complejidad de las estructuras.

Existen diferentes topologías de flip-flops. En ésta sección se describirán aquellas de lógica estática. La Figura 2.17a muestra la clásica compuerta flip-flop basada en transmission-gate, denominada TGMS, del inglés *transmission-gate master-slave*. Otra variación de TGMS es la topología que se muestra en la Figura 2.17b, que se deriva del TGMS PowerPC<sup>3</sup>603. En PowerPC 603 las realimentaciones en los registros se basa en inversores  $C^2MOS$ .

La Figura 2.17c muestra la tercera topología, que es un inversor de reloj modificado ( $mC^2MOS$ ), donde el master-slave dinámico  $C^2MOS$  se modifica por un  $C^2MOS$  pseudo-estático, mediante la adición de una realimentación  $C^2MOS$  en las salidas. El cuarto flip-flop master-slave, Figura 2.17d, se basa en la topología tradicional SR-latch, con realimentación cruzada de compuertas NADN/NOR.

## 2.5. Procesos CMOS

En la presente sección se discutirá la diferencia entre los procesos propietarios o *vendor rules* y los procesos *escalables*, presentando las principales características de ambos, resaltando las ventajas y desventajas.

Los procesos escalables CMOS (SCMOS) son un conjunto de capas lógicas junto con sus reglas de diseño, que conforman un proceso e interfaz métrica con reglas de tipo *vendor-independent rules*, para muchos nodos de fabricación CMOS disponibles a través MOSIS. El diseñador trabaja en las capas abstractas SCMOS en referencia a la unidad mínima métrica denominada “lambda”. Luego, se especifica el proceso y el valor final de lambda, con

---

<sup>3</sup>PowerPC es el nombre original de la arquitectura de computadoras de tipo RISC, que fue desarrollada por IBM, Motorola, y Apple. Los procesadores de esta familia fueron producidos por IBM y Freescale Semiconductor (que era la división de semiconductores y microprocesadores de Motorola), siendo utilizados principalmente en ordenadores o computadores Macintosh de Apple Computer hasta el año 2006 y en varios modelos IBM [18].

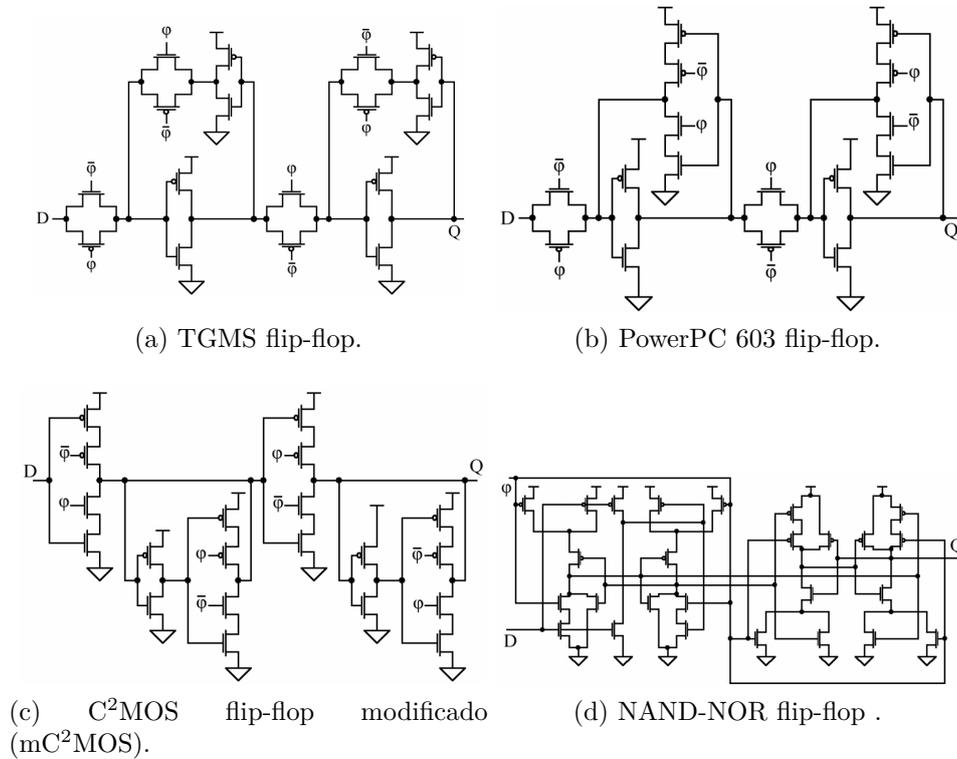


Figura 2.17: Diferentes topologías de flip-flops [19].

lo cual Mosis mapea el diseño SCMOS, generando las verdaderas capas y dimensiones absolutas requeridas por el proceso *vendor*.

Por el contrario, el uso de reglas de diseño propietarias ("*vendor rules*") resulta en un diseño que no es probable, pero en el cual se pueden controlar muchas más variables del mismo [20].

### 2.5.1. Procesos propietarios

Se define a los procesos propietarios, en contraposición con los procesos escalables, a aquellos procesos del tipo *vendor rules*. Por lo general se necesitan más capas para definir a éstos que en los procesos SCMOS, ya que se trabaja directamente con las máscaras del proceso fotolitográfico. Mayor cantidad de layers se traduce en mayor cantidad y complejidad de reglas de diseño.

Las reglas propietarias resultan más apropiados cuando se busca el máximo aprovechamiento del área de silicio, un control más directo sobre los

parámetros del circuito analógico, o en grandes series de producción, donde la inversión agregada en el tiempo de desarrollo y la pérdida de la portabilidad de diseño está claramente justificada [20].

### 2.5.2. Procesos escalables

La unidad fundamental en la definición de un conjunto de reglas de diseño es el ancho mínimo, *minWidth*, y es sinónimo de la dimensión mínima de la máscara que se puede transferir de forma segura al material semiconductor. En general, el ancho mínimo se establece por la resolución del proceso, que se basa comúnmente en la litografía óptica.

El enfoque de reglas de diseño escalable, popularizado por Mead y Conway, define todas las reglas en función de un solo parámetro, llamado lambda ( $\lambda$ ). Las reglas se eligen de modo que un diseño pueda ser fácilmente portado entre varios procesos industriales. El escalado de la dimensión mínima se logra simplemente cambiando el valor de  $\lambda$ , lo cual resulta en una escala lineal de todas las dimensiones.

Para un proceso dado,  $\lambda$  es definido con un valor específico, y todas las dimensiones del diseño en consecuencia se traducen en números absolutos. Típicamente, el ancho de canal de un transistor MOS mínimo de un proceso se establece en  $2\lambda$  [15].

Dentro de los procesos ofrecidos por MOSIS en su programa académico (MEP), existen varios procesos SCMOS (CMOS escalables) que son compatibles entre sí, por ejemplo los procesos de TSMC de 0.35, 0.25 y 0.18 micrones y los procesos de ON de  $0.5\mu\text{m}$ .

En contraposición a un proceso tipo *vendor rules*, los procesos escalables son mucho más simples, ya que poseen capas lógicas, con lo cual el juego de reglas de diseño disminuye, así como también la complejidad en muchas etapas del PDK.

### 2.5.3. Proceso C5NF - SCMOS\_SUBM

El proceso elegido, para trabajar reúne varias condiciones que benefician el desarrollo de la tesis. Primeramente, es un proceso para el cual no se ha desarrollado un iPDK, brindando así la posibilidad de generar un trabajo innovador, tanto desde la teoría y la generación de conocimiento, como desde la parte práctica y su resultado. A su vez, la simpleza del mismo convierte a éste en un trabajo más conceptual que laborioso, pudiendo dedicarse mayor

tiempo a estudiar nuevas formas de verificación de cada etapa del proceso. Por último, es importante resaltar también que el hecho de que éste sea un proceso escalable propone un mejor marco para trabajar en el desarrollo de celdas digitales estándar, generando un pitch y grilla única de trabajo, escalable según el valor lambda [10].

Otro aspecto importante del parámetro lambda, es que convierte a las librerías digitales en librerías portátiles a diversos procesos, modificando justamente el lambda. Éste método define la resolución y grilla del proceso (la grilla es generalmente de medio lambda), y lambda es generalmente la mitad del largo mínimo de los canales de los transistores. El servicio que ofrece MOSIS, en sus procesos escalables, siguen las reglas de Mead y Conway [21].

## 2.6. Conclusiones

En el presente capítulo se presentaron las generalidades del proceso CMOS, desde una descripción cualitativa del funcionamiento del transistor MOS, núcleo fundamental para el desarrollo de los circuitos integrados VLSI, hasta la descripción de los circuitos lógicos y el proceso de fabricación en sí. Se hizo énfasis en describir detalladamente el funcionamiento y características del inversor CMOS, las cuales se pueden extrapolar para entender cualquier circuito digital CMOS:

- Altos márgenes de ruido.
- Los niveles lógicos no dependen de los tamaños relativos de los dispositivos, de manera que los transistores pueden ser de tamaño mínimo. Ésta propiedad se denomina *ratioless*.
- Siempre existe un camino con una resistencia finita entre la salida y  $V_{DD}$  o GND. Un inversor CMOS bien diseñado, tiene una baja impedancia de salida, lo que genera menor sensibilidad al ruido y perturbaciones.
- La resistencia de entrada del inversor CMOS es extremadamente alta. Teóricamente: *fan-out*  $\rightarrow \infty$ )
- Aumentar el fan-out de salida aumenta el retardo de propagación. Esto degrada la respuesta transitoria.
- Consumo mínimo de potencia estática.

También se presentaron diferentes topologías para las celdas flip-flops, las cuales son fundamentales para el diseño de integración a gran escala. Según las diferentes características, se decidió diseñar éstas celdas utilizando la topología transmission-gate master-slave, detallada en la Figura 2.17a, debido a su simpleza y reducido área.

## Capítulo 3

# Desarrollo de un Kit de Diseño Interoperable

Un kit de diseño de proceso, PDK del inglés *process design kit*, es un conjunto de archivos que contienen toda la información necesaria para realizar diseño de circuitos integrados, en un proceso específico.

Un PDK incluye símbolos de esquemático, modelos de SPICE, *design rules* (que permiten validar los diseños antes de su fabricación), celdas paramétricas (*PCells*), archivos de parámetros, denominados archivos CDF, del inglés *Component Describe File*, y las devoluciones de llamada o código evaluador (en inglés *callbacks*) que se deben ejecutar cuando uno o más de estos parámetros cambia. Generalmente, están compuestos por muchos archivos de datos con formatos propietarios, exclusivos para cada *EDA vendor*, con lo cual cada PDK tradicional debe ser escrito para cada herramienta.

### 3.1. iPDK

El concepto de Kit de Diseño de Procesos Interoperable (iPDK) fue introducido por el consorcio IPL Alliance, para la generación de PDKs estándar e interoperables, es decir, escritos en lenguajes abiertos y con la posibilidad de ser utilizados en un gran abanico de diferentes herramientas EDA. Está basado en la base de datos OpenAccess y utiliza lenguajes estándar como Tcl y Python, que garantizan la interoperabilidad entre todas las herramientas de proveedores de EDA. Estos PDKs interoperables incluyen un conjunto completo de APIs para permitir la personalización, apoyar características

avanzadas para los PDK y proporcionar un entorno interactivo para el desarrollo PDK. Se utilizan celdas paramétricas escritas en lenguaje Python, denominadas “PyCells”.

La necesidad de desarrollar un Kit de Diseño de Procesos Interoperable se debió al hecho de que los PDKs tradicionales, usualmente, son diseñados para ser compatibles solo con herramientas EDA específicas, y no concibiendo la posibilidad de migrar dentro de un conjunto de herramientas comunes EDA disponibles en la industria. El concepto iPDK aborda esta necesidad y ha sido resultado del trabajo en conjunto entre desarrolladores de herramientas EDA. El iPDK ofrece las siguientes ventajas:

1. Un único PDK para todas las herramientas de diseño, reduciendo de este modo el desarrollo PDK y el tiempo de mantenimiento, generando a su vez un fuerte vínculo de realimentación y cooperativismo entre los diferentes entes que utilizan los mismos procesos.
2. Libre elección de las herramientas EDA
3. Base de datos libre (OpenAccess) para el desarrollo del PDK
4. Elimina la necesidad de traducir datos
5. Posibilita el uso de celdas paramétricas de proceso portable

### 3.1.1. Estructura Propuesta

Un Kit de diseño de proceso interoperable tiene una estructura definida por las entidades que comprenden la IPL Alliance, la cual hay que respetar para mantener la interoperabilidad del mismo. Dentro de ésta podemos distinguir dos grupos de componentes, aquellos que son suministrados por el foundry (tal vez no en formato OpenAccess) y aquellos componentes que deben ser parte del iPDK.

En el primer grupo ubicamos toda la información referida a la verificación, como son las reglas de diseño (DRC) o las reglas layout versus schematic (LVS); el modelo de SPICE de los transistores del proceso y todo lo referido a la extracción de parásitos.

En el segundo grupo ubicamos cinco grandes sub-grupos: la librería PDK propiamente dicho (con layouts y esquemáticos de los diferentes dispositivos), la documentación clara y fehaciente del proceso y del PDK, los archivos de tecnología y display, los scripts para ejecutar las diferentes herramientas y los archivos ‘\*.defs’, que definen cómo abrir la librería.

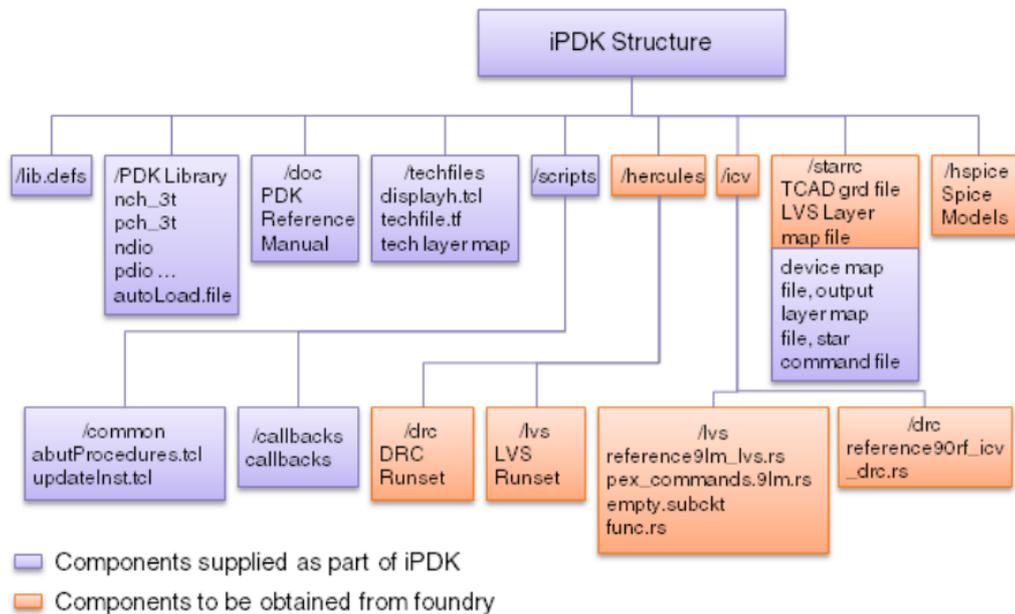


Figura 3.1: Estructura de archivos de un iPDK [23].

## 3.2. Archivos de tecnología

Los archivos de tecnología son una serie de documentos escritos en formato ASCII que proporcionan información relativa al proceso de fabricación, tales como los nombres y orden de los layers, las características físicas de cada material y algunas reglas de diseño, además de añadir información sobre los recursos de visualización.

Los archivos de tecnología se almacenan en formato binario dentro de la herramienta EDA de diseño custom. Sin embargo, se puede exportar y editar los datos en formato ASCII o cargar los datos nuevamente en una biblioteca. La extensión predeterminada para archivos de datos de tecnología es \*.tf, pero puede especificar cualquier extensión de archivo. Se puede exportar el archivo de recursos de visualización como datos ASCII, ya sea en formato de Tcl (\*.tcl) o mostrar en formato de recursos (\*.drf).

### 3.2.1. Tech files

Los *Tech files*, o archivos de tecnología propiamente dichos, se utilizan para añadir información de tecnología a una biblioteca. El tech file contiene

normalmente información sobre los layers, definiciones de reglas de diseño, y asociaciones con los paquetes de visualización definidas en el archivo de recursos de visualización. Se presentan datos sobre la grilla de ruteo, las capas física, lógicas y de asistencia, tamaño de contactos y definiciones de vías.

Cuando un archivo de tecnología es exportado en formato ASCII, éste se compone de varias secciones y subsecciones. Cada sección o subsección comienza con una palabra clave seguida de un paréntesis abierto ‘(’. Seguido a esto, el cuerpo de la sentencia está compuesta por la sección de atributos de declaraciones o las diferentes subsecciones. Cada sección termina con un paréntesis de cierre ‘)’. Cada declaración de atributos dentro de una sección consta de una secuencia de atributos encerradas entre paréntesis. Los atributos se separan con uno o más espacios.

El archivo de tecnología debe contener las siguientes secciones y subsecciones en el orden en que se enumeran aquí:

- *Controls section*
  - **techParams:** Define parámetros estándar
    - maskGrid
    - cadGrid
    - drcGrid
    - mfgGrid
  - **techPermissions:** Define permisos de usuarios.
  - **viewTypeUnits:** Define las unidades estándar y las unidades de base de datos por unidad de usuario, para cada vista de diseño.
  - **mfgGridResolution:** Define la resolución de la grilla de fabricación (*manufacturing grid*).
- *Layer definitions*
  - **techPurposes:** Define las funcionalidades de las capas; incluye tanto funciones reservadas del sistema y funciones definidos por el usuario.
  - **techLayers:** Nombre y numeración de los distintos layers.
  - **techLayerPurposePriorities:** Asocia cada layer con una función.

- **techDisplays:** Asocia un paquete de visualización (definidos en el *Display Resource File* - ver Sección 3.2.2) con un par función-layer.
  - **techLayerProperties:** Asigna diversas propiedades a las capas.
  - **techDerivedLayers:** Define capas que se derivan de otras capas.
- *Layer rules:* La sección `layerRules` define reglas personalizadas que se aplican para diferentes capas. Estas incluyen propiedades como resoluciones de grilla de fabricación de capas específicas o la dirección de ruteo para layers específicos (para setear la dirección de los metales en el ruteo Manhattan).
  - *Via definitions:* La sección `viaDefs` define cada uno de los tipos de vía que se utilizan para el diseño. Define la forma de la vía, los layers que se conectan, y las reglas de diseño asociadas a la vía.
  - *Constraint Groups:* La secciones `constraintGroups` define reglas de diseño que restringen la interacción física entre los layers. Cada restricción puede anidarse jerárquicamente dentro de otros grupos de restricciones.

### 3.2.1.1. Reglas para escribir un archivo de tecnología

Se deben seguir las siguientes reglas para crear un archivo de tecnología y conservar su formato:

- Cada sección o subsección debe comenzar con una palabra clave seguida de un paréntesis abierto “(” y terminar con un cierre de paréntesis “)”
- Cada sección o subsección en el archivo de la tecnología puede contener más de una declaración de atributos
- Cada declaración atributos dentro de una sección puede consistir en una secuencia de atributos, que debe estar contenida entre paréntesis

### 3.2.2. Display Resource File

Los archivos de recursos de visualización, en inglés *Display Resource File*, son una serie de archivos de texto que contiene un conjunto de comandos Tcl<sup>1</sup>

---

<sup>1</sup>Tcl, del acrónimo en inglés “Tool Command Language”, es un lenguaje de script utilizado para escribir rutinas y funciones en las herramientas de diseño, como pueden ser los diferentes menues de la interfaz gráfica.

que son interpretados por la herramienta EDA. El objetivo de estos archivos es controlar como se verán los layers en pantalla, asignando diferentes colores y tramas a los diferentes layers. Hay dos formas de construir un display resource file. Primero, la mayoría de las herramientas EDA cuentan con un editor gráfico, generalmente denominado *Display Resource Manager*. La otra variante consta de escribir el archivo manualmente, codificado en lenguaje Tcl.

### 3.3. Componentes Front-End

Los componentes front-end son los bloques iniciales básicos de un PDK, y se detallan a continuación:

- Una biblioteca, que contendrá una colección de símbolos esquemáticos
- Un juego de archivos que describen los parámetros de las celdas y las bibliotecas, denominado *Component description format* (CDF)
- Información específica para la simulación
- Vistas de simulación
- *Callbacks prodecures*

#### 3.3.1. Component Description Format

Los archivos *Component Description Format* (CDF) describen los parámetros y atributos de parámetros de los diferentes componentes de una librería. Se utilizan para agregar información y funcionalidades a las bibliotecas y a las PCells; y para crear y describir componentes. Una descripción CDF asigna parámetros y atributos con diferentes propósitos, como:

- Asignar nombres a los parámetros
- Asignar unidades y valores por defecto
- Verificar que los valores se encuentran dentro de los rangos especificados
- Cambiar dinámicamente cómo se muestran los parámetros, dependiendo de las condiciones predefinidas
- Ejecutar las diferentes callbacks cuando corresponde, según como varían los parámetros

**Interoperable Component Description Format** Se define a los archivos CDF interoperables como iCDF, del inglés *Interoperable Component Description Format*. Éstos poseen una sintaxis determinada, diseñada por Synopsys para la IPL Alliance. En este trabajo, éstos parámetros se definieron a través del *Custom Designer* <sup>®</sup> *Parameter Definition Editor*.

### 3.3.2. Callbacks

Las funciones denominadas *callbacks* son procedimientos que se ejecutan cuando cambia un parámetro definido en el CDF. Los procedimientos de devolución de llamada se utilizan para verificar y validar las entradas del usuario, calcular los parámetros dependientes y comprobar los rangos de parámetros. Éstos programas se escriben en lenguaje Tcl.

Cada herramienta EDA cuenta con sus propios comandos *tool-specific*, y deben ser contemplados para escribir las diferentes callbacks. Para el desarrollo de los PDKs interoperables, se definió una API para escribir rutinas de devolución interoperables. Todos los comandos específicos deben comenzar con el prefijo “*ipdk\_*”, y cada vendedor debe desarrollar estos procedimientos para sus herramientas, de forma de garantizar la interoperabilidad [24].

A continuación se muestra un ejemplo de la devolución de llama escrita para la resistencia de ELECT y HR. Éste código se ejecuta cuando se varían los valores de los parámetros R, L o H, que representan el valor de la resistencia, el largo y el alto del dispositivo, respectivamente. La variable *ResBy* establece cuales son los grados de libertad para definir al resistor. El comando *ipDK\_setParamValue* pertenece a la API definida por la IPL Alliance y se utiliza para establecer un valor de un parámetro de una celda.

```

1  if {$ResBy == "L & H"} {
3      if {$edited_param == "R"} {
4          puts "ERROR: You must change L or H"
5      }
6      set res [expr ($Resrsq * ($ResL/$ResH))]
7      ipDK_setParamValue R $res $inst 0
8  }
9  elseif {$ResBy == "R & L"} {
11     if {$edited_param == "H"} {
12         puts "ERROR: You must change R or L"
13     }
14     set hei [expr ($Resrsq * ($ResL/$ResR))]
15     ipDK_setParamValue H $hei $inst 0
16 }
17 elseif {$ResBy == "R & H"} {
19     if {$edited_param == "L"} {
20         puts "ERROR: You must change R or H"

```

```
21     }  
22     set len [expr ($ResH * ($ResR/$Resrsq))]  
23     iPDK_setParamValue L $len $inst 0  
24 }  
25 else {  
26     puts "ERROR: Wrong value"  
27 }
```

## 3.4. PCells

Se llama PCell, acrónimo del inglés *parametric cell*, a las celdas paramétricas que representan una parte o un todo de un componente o dispositivo analógico, cuya estructura depende de uno o más parámetros. Así, una pcell es una celda que se genera y modifica automáticamente por eventos ocurridos dentro del entorno de software de automatización de diseño electrónico (EDA), basado en los valores de estos parámetros. Por ejemplo, pueden existir diferentes celdas paramétricas de transistores, y luego diferentes instancias de la misma, con diferentes parámetros, como pueden ser las longitudes y anchos del canal, definidos por el usuario.

### 3.4.1. Aplicación

En el diseños de circuitos electrónicos, las celdas paramétricas son las unidades básicas de funcionalidad. Una pcell es, lógicamente, más flexible que una celda no parametrizada, ya que las diferentes instancias pueden tener diferentes valores de parámetros y, por lo tanto, diferentes estructuras. Por ejemplo, en lugar de tener varias definiciones diferentes de celdas que representan transistores o resistencias, de diferentes tamaños, de diversas formas, se tiene una sola pcell que puede tomar dimensiones y topologías diferentes, definidos como parámetros.

Las estructuras dentro de un circuito integrado y las reglas de diseño que rigen las dimensiones físicas son a menudo complejas, resultando así tedioso el trabajo de dibujar a mano diferentes estructuras. Mediante el uso de pcells, un diseñador de circuitos puede generar fácilmente un gran número de diversas estructuras que sólo difieren en algunos parámetros, aumentando así la productividad del diseño y consistencia.

### 3.4.2. Implementación

Una PCell es una estructura generada a partir de una porción de código, en cierto lenguaje específico. Este código es responsable del proceso de creación de una estructura a nivel layout físico, a partir de la base de datos original y de los parámetros introducidos. Dado que la estructura en nivel código puede crear muchos objetos diferentes dependiendo de los parámetros, se denomina a la celda en el lenguaje definido como *Master PCell*. El objeto (a nivel layout) que es creado por el código, se llama *instanced PCell*.

Existen diferentes lenguajes estandarizados para escribir celdas paramétricas. SKILL <sup>®</sup> es sin dudas el más popular de éstos, siendo un lenguaje propietario y cerrado, propiedad de Cadence <sup>®</sup>. A su vez, Mentor Graphics <sup>®</sup> escribe sus PCells en lenguaje AMPLE, acrónimo del inglés *Advanced Mentor Programming Language*, el cual es también un lenguaje cerrado y propietario. Por su parte, Synopsys <sup>®</sup> propuso en un principio escribir las celdas paramétricas en Tcl, un lenguaje abierto y estándar en la industria de software. Siguiendo otra línea de trabajo, la comunidad OpenAccess estableció C++ como lenguaje para escribir celdas paramétricas, desarrollando una API abierta. Tiempo después, con la conformación del consorcio IPL Alliance, y siguiendo las directivas de la base de datos OpenAccess, se propuso utilizar Python, un lenguaje simple, abierto, orientado a objetos y popular, para escribir las celdas paramétricas. De esta forma, el consorcio definió una completa API para escribir pcells en Python, denominadas *PyCells*.

## 3.5. PyCells

Se conoce como PyCells a las celdas paramétricas generadas en lenguaje Python y son una extensión de la base de datos OpenAccess. Dentro de la API de Python desarrollada, una PyCell admite las siguientes funciones de diseño:

- Función de *auto-abutment* para transistores MOS
- Función de *stretch handles* para transistores MOS
- Opciones DFM<sup>2</sup>

---

<sup>2</sup>DFM, del inglés *Design for manufacturability* (Diseño para fabricación) es un enfoque en ingeniería, donde se pretende diseñar productos de tal manera que son fáciles de fabricar [22].

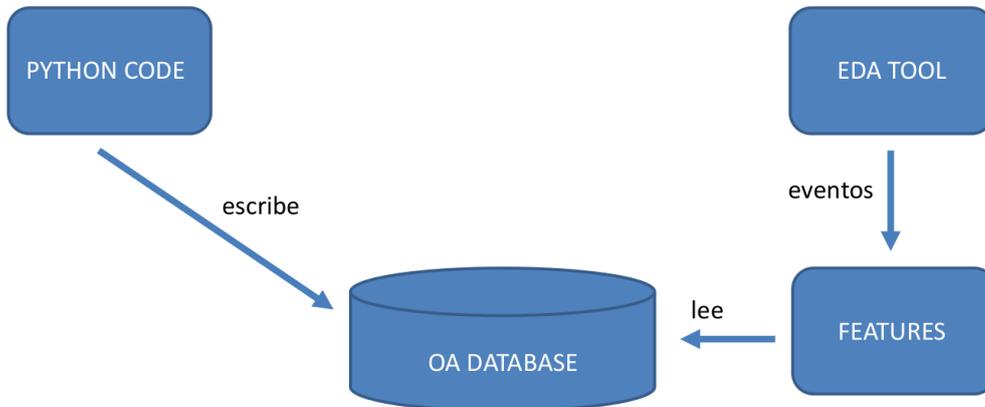


Figura 3.2: Instanciando una PyCell.

En la Figura 3.2 se observa el flujo que representa instanciar una PyCell en un diseño. El *Python code*, o master pycell, escribe la base de datos Open-Access. Al producirse un evento, la herramienta EDA lee la base de datos y con los parámetros de entrada genera la instancia particular de la pycell.

### 3.5.1. PyCell Studio

*PyCell Studio* es un entorno de trabajo diseñado para desarrollar celdas paramétrica utilizando el lenguaje Python, que contiene una IDE<sup>3</sup> en donde se puede depurar el código en python en tiempo real y ver el layout de la celda paramétrica. A su vez, este entorno contiene la completitud de la API<sup>4</sup> que ofrece diferentes funciones, métodos y clases específicas para el diseño de celdas paramétricas.

El PyCell Studio requiere que el archivo de la tecnología ASCII (\*.tf) se convierta al formato de archivo Santana.tech. El formato Santana.tech puede ser interpretada directamente por la API de PyCell Studio para crear PyCells.

Para un desarrollador de PDKs, la arquitectura PyCell facilita el trabajo,

<sup>3</sup>Un ambiente de desarrollo integrado o entorno de desarrollo interactivo, del inglés *Integrated Development Environment*, es una aplicación informática que proporciona servicios integrales para facilitarle al desarrollador o programador el desarrollo de software.

<sup>4</sup>Interfaz de programación de aplicaciones, abreviada como API, del inglés *Application Programming Interface*, es un conjunto de subrutinas, funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

ya que separa completamente la geometría del dispositivo de las reglas de diseño de procesos. Esto permite a los desarrolladores centrarse en la geometría, sin necesidad de comprender la complejidad de las reglas del proceso. La API construye la geometría correcta, y es capaz de optimizar el diseño. Esto significa que las mismas PyCells se pueden utilizar en una amplia variedad de procesos. Por ejemplo, la biblioteca desarrollada por la IPL Alliance es adaptable a más de una docena de tecnologías entre 250 a 40 nm. Además, las PyCells son compatibles con todas las herramientas EDA que trabajan con formato OpenAccess utilizados en el diseño de circuitos integrados, lo cual permite la portabilidad e interoperabilidad de las mismas.

### 3.5.2. Python

A continuación se muestra una parte del código en Python de la PyCell de un transistor MOS. En éste fragmento se define si el transistor tendrá o no P/NWELL y se definen los leyes para los WELL y los SELECT, dependiendo del tipo de transistor. Esto se realiza con la clase `Layer()`, que atribuye una capa definida en el Santana.tech a una estructura particular.

```

# define the layers which should be used for enclosure rectangles
2 if self.Well == 'yes':
    if self.tranType == 'nmos':
4         self.encLayers = [Layer('nselect'), Layer('pwell')]
        else:
6             self.encLayers = [Layer('pselect'), Layer('nwell')]
    else:
8         if self.tranType == 'nmos':
            self.encLayers = [Layer('nselect')]
10        else:
            self.encLayers = [Layer('pselect')]

```

## 3.6. Archivos de verificación

Los archivos de verificación son documentos que contienen información sobre las reglas de diseño, y sirven para comprobar el correcto diseño de los chips y asegurar su manufacturabilidad. Al ser el proceso SCN3ME un proceso escalable *independent-vendor*, se encuentran sus reglas publicadas con libre acceso [20].

### 3.6.1. DRC

Las reglas de DRC, del inglés *Design Rule Check*, sirven para comprobar la geometría de los diseños, contemplando la litografía óptica del proceso. Se escriben en base a los tres tipos básicos de reglas que se describieron en la Sección 2.2.2, y éstas son las reglas de tamaño mínimo, separación y solapamiento.

A modo de ejemplo, se presentan las reglas escritas para Calibre <sup>®</sup> para el layer POLY. Las mismas son las reglas 3.1, 3.2 y 3.3, según [20], y se observan gráficamente en la Figura 3.3. La regla 3.1 es el tamaño mínimo para el layer POLY, y es de  $2\lambda$ , es decir  $0.6\mu\text{m}$  (para el proceso SCNM3E\_SUBM). DRC3\_1 es la denominación de la regla, y lo escrito luego de @ es el comentario que explica la regla.

```

1 DRC3_1 { @ Poly: Minimum width = 0.6
3     INT poly < 0.6 SINGULAR
  }
```

La regla 3.2 describe la distancia mínima (spacing) que debe existir entre dos líneas de POLY, que es también  $2\lambda$ .

```

1 DRC3_2 { @ Poly: Minimum spacing over field = 0.9
3     EXT poly < 0.9 SINGULAR
  }
```

La regla 3.3 describe la extensión mínima de POLY sobre ACTIVE, también  $2\lambda$ . Las layers `pgate` y `ngate` son capas auxiliares que definen a los gates de transistores pMOS y nMOS respectivamente. Se utilizan para facilitar la escritura de las reglas. Por su parte, las capas `psrcdrn` y `nsrcdrn` definen las zonas activas, para transistores pMOS y nMOS respectivamente, de drain y source.

```

1 DRC3_3 { @ Poly: Minimum gate extension of active = 0.6
3     pgate TOUCH psrcdrn == 1
3     ngate TOUCH nsrcdrn == 1
5     ENC active poly < 0.6 ABUT == 0 SQUARE REGION
  }
```

### 3.6.2. LVS

Las reglas de LVS, del inglés *Layout Versus Schematic*, tienen la finalidad de constatar que el diseño a nivel layout se corresponde con el diseño a nivel

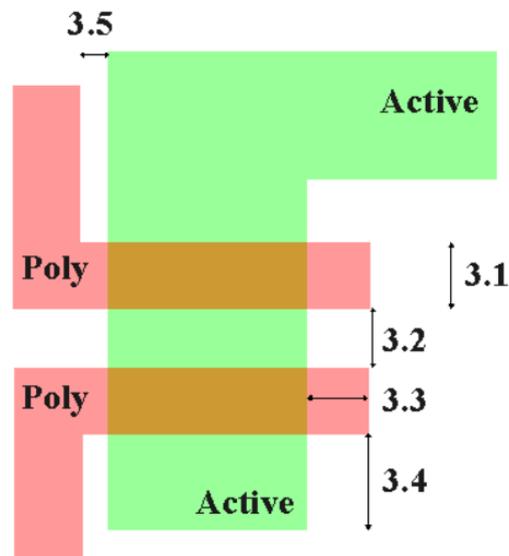


Figura 3.3: Reglas DRC para el layer POLY [20].

esquemático. Es uno de los pasos más importantes en el flujo de verificación, ya que, si bien un diseño puede pasar correctamente el chequeo de las reglas de diseño, puede que no se ajuste al diseño esquemático deseado. El LVS consta de tres pasos:

1. **Extracción:** A partir del layout físico del diseño, se hallan los diferentes dispositivos representados en el dibujo. Se determina el camino de interconexión de las capas de metal.
2. **Reducción:** Se combinan los componentes extraídos en configuraciones serie-paralelo. A su vez se genera el netlist del diseño.
3. **Comparación:** Se compara el diseño extraído y el esquemático, a nivel netlist. Se evalúan las dimensiones de los dispositivos. Si ésta comparación resulta exitosa, el análisis devuelve una evaluación positiva. De lo contrario, si la comparación no es igual, devuelve el error correspondiente [25].

El proceso de extracción requiere que se definan las reglas de extracción. A continuación se muestran las reglas LVS escritas para el proceso en lenguaje de Calibre <sup>®</sup>, para un transistor nMOS.

```

1 DEVICE MN NGATE POLY(G) NSRCDRN(S) NSRCDRN(D) PSUB(B) (S D) NETLIST MODEL n
  [
3   prop w, l, AS, AD
   w = (perim_co(ngate,s) + perim_co(ngate,d))*0.5
5   l = (perim(ngate) - perim_co(ngate,s) - perim_in(ngate,s) - perim_co(
   ngate,d) - perim_in(ngate,d))*0.5
   AS = area(S)
7   AD = area(D)
  ]

```

Los valores  $w$  y  $l$  se extraen a través de los perímetros de las distintas áreas del dispositivo.

### 3.7. Archivos de extracción

Los archivos de extracción son archivos confidenciales provistos por el foundry. Éstos contienen información del proceso nativo o *vendor-dependent* que no puede ser divulgada. Por ésta razón, no se incluyen los archivos de extracción en el iPDK generado, pero si está contemplado dentro del entorno de trabajo.

### 3.8. Conclusiones

En el presente capítulo se presentó detalladamente el concepto de PDK y la variante introducida por la IPL Alliance de los PDKs interoperables. Esta nueva tendencia es importante porque posibilita utilizar un mismo PDK para diferentes entornos de trabajo y con diferentes herramientas EDA. Se presentaron los conceptos de celdas paramétricas, archivos de tecnología, reglas de diseño, callbacks y archivos de descripción de componentes, parte de la estructura PDK. A su vez, también se introdujo las versiones interoperables PyCells, celdas paramétricas escritas en Python y archivos iCDF.

# Capítulo 4

## Diseño de un conjunto de celdas estándar

El diseño de circuitos digitales complejos se realiza utilizando conjuntos de celdas estándar. Las mismas son un grupo de compuertas elementales de diferentes funciones lógicas, fan-in y fan-out; y otros circuitos básicos, como registros y memorias. Éste paradigma considera la disposición física de dispositivos dentro de celdas rectangulares, en principio de igual altura. Inicialmente, un circuito se divide en varios bloques funcionales más pequeños, para luego ser definido según su equivalente a algún subcircuito predefinido. La funcionalidad y las características eléctricas de cada celda son probadas, analizadas, y especificadas. Una colección de estas celdas se denomina biblioteca de celdas estándar. Por lo general, una biblioteca de celdas estándar consta de 500-1200 celdas.

El proceso de diseño basado en celdas estándar posibilita la realización de circuitos de gran escala de integración. Es importante destacar que:

- Permite la automatización en la síntesis de circuitos digitales
- Permite la optimización del diseño digital
- El desempeño del circuito depende de la calidad del diseño del conjunto de celdas estándar

En éste capítulo se introduce la teoría de diseño de celdas estándar, así como se introduce en el flujo de diseño digital, incluyendo la descripción de las herramientas de ruteo, estrategias de ruteo y estilos de celdas. En el Anexo A se presentan los layouts de las celdas estándar diseñadas.

## 4.1. Analogía Lego

La “*analogía lego*®” es un modelo que se definió para explicar y poder comprender de forma más intuitiva el flujo de diseño digital. En éste modelo, se definen a las diferentes celdas digitales estándar como bloques de plástico interconectables, denominados *ladrillitos*, y se definen a los circuitos digitales como estructuras construidas por éstos bloques, como por ejemplo un castillo.

Así, teniendo un bloque con una única función y color específico, es posible diseñar un castillo, pero al aumentar la cantidad y variedad de piezas, con funciones más específicas, como pueden ser puertas, arcos o bloques de diferentes colores, el diseño será más eficiente. Esto mismo ocurre con las celdas digitales: **todo circuito digital puede ser sintetizado con una compuerta NAND u OR y un flip-flop**, pero el circuito será más eficiente a nivel físico al desarrollar más celdas con funcionalidades específicas.

Se diseñan las celdas estándar de forma tal que la herramienta de *place & route* sea capaz de instanciar las diferentes celdas en una matriz ordenada, donde las celdas comparten el riel de alimentación y tierra. Todas las celdas están contenidas dentro de un borde denominado *P&R boundary*, el cual indica el límite de las celdas.

## 4.2. Diseño

La tendencia actual en el diseño VLSI es desarrollar chips de alto rendimiento. El objetivo principal del diseño a nivel físico es satisfacer las necesidades de rendimiento mientras se minimiza el tamaño del die. Para diseñar chips de alto rendimiento, como por ejemplo microprocesadores (que trabajan a frecuencias entre 100MHz a 1 GHz), se utiliza la técnica de ruteo denominada *over-the-cell*. Se emplea a través de todo el chip dentro del flujo automatizado de diseño. De esta forma, las interconexiones se realizan sobre las celdas [26].

Existe una relación de compromiso entre disminuir el área de cada celda y respetar las recomendaciones de la herramienta de *place & route*. Celdas más pequeñas ocuparán menos área, pero dificultarán el ruteo. Enfocarse en reducir el ruteo *over-the-cell* es una mejor estrategia de diseño que reducir el área individual de cada celda. Para esto, es fundamental diseñar cada celda según los lineamientos de las herramientas.

### 4.2.1. Ruteo *Over-the-Cell*

En procesos que disponen dos capas de metal, las celdas estándar se instancian en forma de mosaicos en filas, de forma que cada fila está separada verticalmente por una distancia no menor a, generalmente,  $110\lambda$  desde la base de la fila anterior. Las interconexiones de M1 (metal 1) horizontales se colocan en los canales de ruteo que se forman entre las filas. El número de conexiones define el alto de los canales de ruteo. La Figura 4.1 muestra el layout físico de un controlador de arquitectura MIPS, en un proceso de 2 metales, utilizando una herramienta de automatización de diseño digital. Se observa claramente el canal de ruteo entre las dos estructuras de celdas estándar. En la Figura 4.2 se muestra la distribución de las celdas en un arreglo con canales de ruteo. Las flechas indican la orientación de las celdas, y los espacios son los canales.

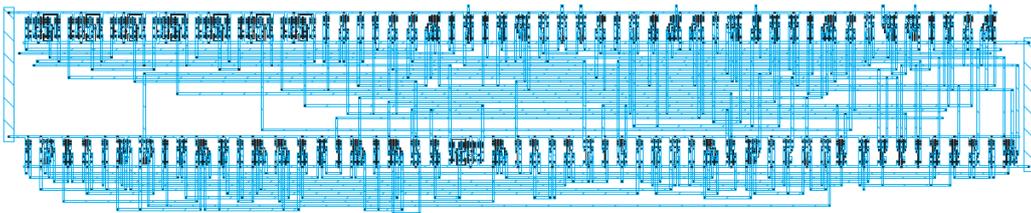


Figura 4.1: Layout de un procesador MIPS de 8 bits [10].

En procesos donde hay disponible mayor cantidad de capas de metal, el ruteo tiene lugar sobre las celdas y los canales de ruteo puede llegar a ser innecesarios. Este concepto se denomina ruteo *Over-the-Cell*, ya que se realiza sobre las celdas estándar. En un proceso de 3 metales, por ejemplo, las celdas se rutean internamente con POLY y M1, mientras que los layers M2 y M3 se utilizan para rutear, entre celdas [10].

En la Figura 4.3 se observa el layout de un microprocesador genérico de 8 bits de arquitectura acumulador<sup>1</sup>, diseñado con herramientas de place and route utilizando la técnica *over-the-cell*, en un proceso CMOS de 90nm y 6 metales. Como se puede observar, los canales de ruteo desaparecen, y se presenta un doble anillo alrededor de toda la estructura que son las líneas de alimentación.

En la Figura 4.4 se presenta la distribución de celdas con ruteo *over-the-cell*. A diferencia de la Figura 4.2, aquí no existen los canales de ruteo y la

<sup>1</sup>Éste diseño fue realizado como trabajo práctico final en la materia 66.33 Laboratorio de Sistemas Digitales.

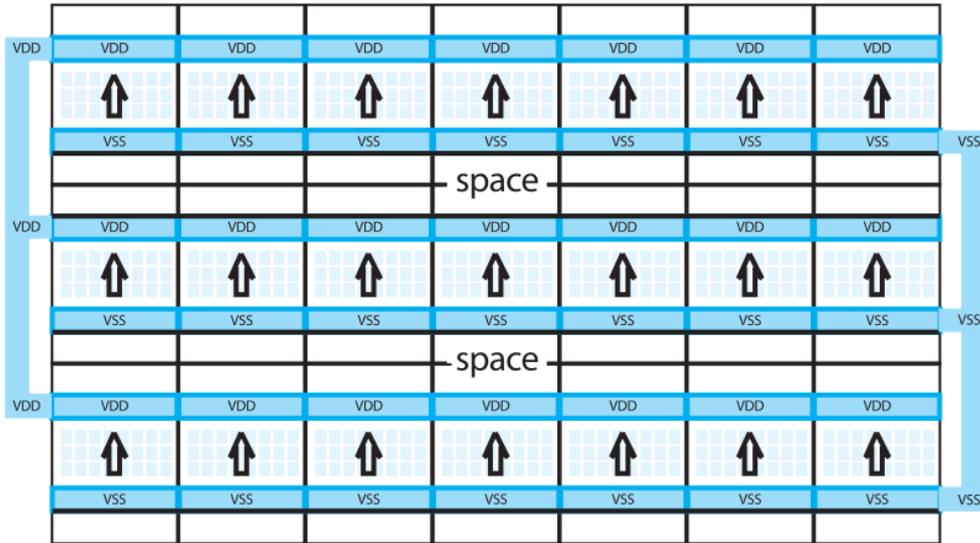


Figura 4.2: Distribución de celdas con canales de ruteo [27].

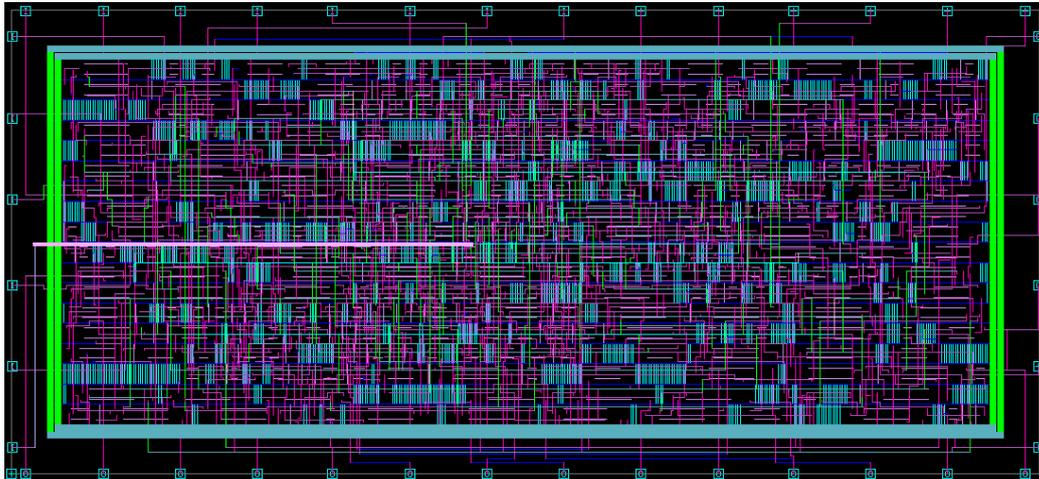


Figura 4.3: Layout de un microcontrolador de 8 bits.

orientación de las celda, indicada con flechas, varía según la posición de los rieles de alimentación.

Debido a que el ruteo sobre las celdas digitales es libre, las herramientas de P&R rutean en M2 y M3 tanto como sea posible. Para esto, existen diversos algoritmos de ruteo, que conducen a diferentes resultados y estilos de layout. Por lo tanto, es recomendable seguir las advertencias de las herramientas de place and route para lograr mayor eficacia de área en el layout final. Se

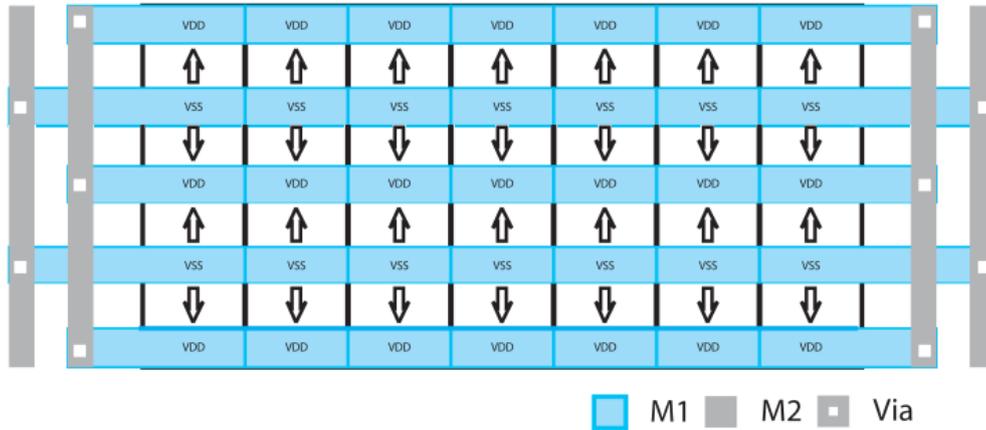


Figura 4.4: Snap-together cells. Los rieles de VDD y GND se superponen.

tomaron las indicaciones de la herramienta de place and route de la empresa Synopsys, que se detallan a continuación:

- Minimizar o eliminar el ruteo con M2 y M3 dentro de las celdas<sup>2</sup>.
- Minimizar el tamaño de los pines. Los pines extra se convierten en bloques para la herramienta de P&R e introducen capacidades parásitas.
- Evitar pines de M2. El P&R introduce las vias entre M1 y M2.

Es importante también considerar si el proceso en el cual se diseñan las celdas estándar es un proceso que admite *stacked vias*<sup>3</sup> o no, ya que esto condicionará también la libertad de la herramienta de place and route y se deberá actuar en consecuencia.

#### 4.2.2. Pitch matching

Se observa claramente en la Figura 4.1 que el área de silicio del controlador está dominado por los canales de ruteo. Cuando la lógica es más regular, la densidad del diseño se mejora debido al concepto *snap-together cells*, en donde los rieles de alimentación y tierra, y otras señales importantes, como el reloj, se conectan al unir las celdas. Las celdas snap-together se instancian en una

<sup>2</sup>El proceso SCN3ME dispone solo de tres metales, con lo cual se elimina toda posibilidad de rutear dentro de las celdas con las capas M2 y M3 de forma intrínseca.

<sup>3</sup>Se define *stacked vias* a las vias que pueden ser construidas una sobre la otra. El proceso SCN3ME admite *stacked vias*.

matriz ordenada y repetible, en donde se solapan los rieles de alimentación, y las celdas se instancian lo más cerca posible, haciendo coincidir su P&R boundary.

Este diseño requiere más esfuerzo, pero conduce a una reducción del área e interconexiones más cortas (lo cual mejora la velocidad). El objetivo de diseño es que todas las tengan el mismo *pitch matching*. Se define al pitch como la distancia que existe entre los rieles de alimentación y tierra en cada celda.

### 4.2.3. Estilos de layout

Existen diversos estilo de layout para el diseño digital VLSI. Sin embargo, existen similitudes en todos los estilos. Siempre se instancian las celdas en una matriz ordenada, según el concepto elegido.

#### 4.2.3.1. Different-height cell in one row

En ésta configuración, todas las celdas pueden diseñarse de diferentes alturas para optimizar el área de cada una. Luego, el *place and router* coloca todas las celdas en una misma fila. Ésta topología no es muy recomendada, salvo cuando se cuenta con solo uno o dos metales en el proceso, utilizando canales de ruteos, como ocurre en el ejemplo de la Figura 4.1. En la Figura 4.5 se presenta un esquema de la topología, donde cada rectángulo de igual color representa una celda de diferente funcionalidad. Se observa que el pitch no se conserva.

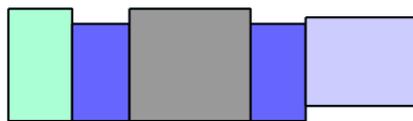


Figura 4.5: Esquema de las celdas *different-height cell in one row*.

#### 4.2.3.2. Single-height cells in single Row

Ésta topología es la más comúnmente utilizada y logra una gran eficiencia. Resuelve salomónicamente la relación de compromiso entre el área utilizado por celda y la libertad del ruteo over the cell. Ahora, todas las celdas tienen la misma altura entre los rieles de alimentación y tierra, el pitch. Se reduce la complejidad del diseño de las celdas estándar al mantener una grilla fija

de trabajo. En la Figura 4.5 se muestra un esquema de la topología, donde se observa que todas las celdas mantienen el pitch. Este arreglo se presenta tanto en filas como en columnas, generando una matriz ordenada.



Figura 4.6: Esquema de las celdas *single-height cells in single row*.

#### 4.2.3.3. Single-and double-or multiple-height cells

La topología *Single-and double-or multiple-height cells* es la más eficiente de todas, pero requiere de procesos más modernos y con mayor cantidad de metales disponibles. Aquí, todas las celdas son optimizadas para ocupar el menor área posible; a su vez que los algoritmos de ruteo se optimizan para instanciar las celdas según el pitch. Generalmente se definen dos o tres alturas, en función del pitch mínimo. En este caso también se aplica el ruteo over-the-cell. En la Figura 4.7 se observa un esquema de esta topología, donde se distinguen dos alturas diferentes, siendo una el pitch mínimo y la otra el doble de este.



Figura 4.7: Esquema de las celdas *single-and double-or multiple-height cells*.

#### 4.2.4. Grilla de ruteo

El diseño de las celdas estándar se define según una grilla de ruteo, la cual es una matriz que se utiliza para guiar a la herramienta de place & route. Luego, todas las celdas se instancian en el diseño según esta grilla, y todas las líneas de ruteo se colocan utilizando ésta como guía.

Para establecer una grilla de ruteo, se deben considerar las reglas de diseño, en particular las de spacing. En la Figura 4.8 se muestran las principales reglas que se deben considerar. La distancia 1 es el spacing *line-to-line*, la distancia 2 es el spacing *line-to-via* y la distancia 3 es el spacing *via-to-via*. Estas distancias se deben respetar principalmente para las capas M2 y M3, así como para las vías, que serán los utilizados para realizar el ruteo over the

cell. Finalmente, el peor caso se dará para el spacing via-to-via, y se establecen las distancias en el eje x y en el eje y de la grilla según las siguientes definiciones:

1. Via-to-Via horizontal ( $g_{x_1}$ ): Via width + M1 spacing
2. Via-to-Via vertical ( $g_y$ ): Via width + M2 spacing
3. Via2-to-Via2 horizontal ( $g_{x_2}$ ): Via2 width + M3 spacing

y se calculan para los procesos CMOS escalables de MOSIS

$$g_{x_1} = 4\lambda + 3\lambda = 7\lambda \quad (4.1)$$

$$g_y = 4\lambda + 3\lambda = 7\lambda \quad (4.2)$$

$$g_{x_2} = 6\lambda + 3\lambda = 9\lambda \quad (4.3)$$

Se discriminan las distancias en horizontal y vertical para aplicar ruteo Manhattan. De ésta forma, se puede toma el peor de los casos 1 y 3 para definir la grilla horizontal y el caso 2 para definir la grilla vertical. Para simplificar el diseño, se optó por hacer el entramado de la grilla cuadrado, es decir, la grilla tiene la misma dimensión para ambas direcciones. Para ésto se tomó el máximo de las Ecuaciones 4.1 y 4.3.

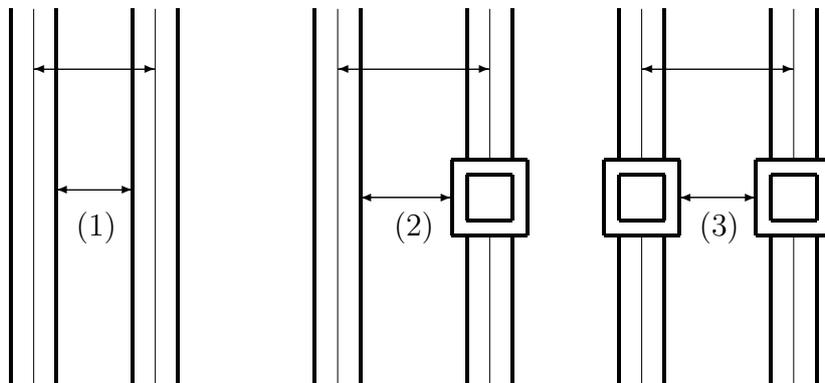


Figura 4.8: Spacing.

### 4.2.5. Parámetros adoptados en el diseño

Se estableció entonces una grilla de ruteo según las reglas escalables del proceso y en referencia a la medida lambda, según lo calculado en la Ecuación 4.3. Todas las celdas son del tipo *single height*, comprendidas entre los rieles de alimentación, ubicados en los extremos de las mismas. Se estableció el valor de pitch según la cantidad de tracks de ruteo que se desean en una celda estándar. Se definió la cantidad de 10 tracks de M3, resultando el pitch en  $90\lambda$ , es decir  $27\mu\text{m}$ . El valor del pitch, al definir una librería tipo *single height*, debe mantenerse constante.

En la Figura 4.9 se muestra el template de la grilla de ruteo, considerando el tamaño del NWELL y la ubicación esquemática de los rieles de alimentación. Las medidas  $g_x$  y  $g_y$  se definieron en valores absolutos según el proceso, es decir  $2.7\mu\text{m}$  ( $9\lambda$ ).  $H_W$ , la extensión de NWELL, se definió como  $54\lambda$ , es decir,  $16.2\mu\text{m}$ . Finalmente, el pitch, definido como H, es de  $27\mu\text{m}$ , es decir,  $90\lambda$  (los 10 tracks de ruteo de M3). Es importante destacar que la grilla está definida con un offset de 4.5 lambda, logrando de ésta forma evitar problemas de violación de reglas DRC al solapar las mismas cuando se realiza el place and routing. La variable  $O_x$  define el offset de la grilla en la figura. El rectángulo verde ejemplifica el límite del P&R boundary. El comienzo de la celda se establece en las coordenadas (0,0), que se detalla también en la figura.

Es fundamental resaltar que los contactos deben estar en grilla (ya que los algoritmos de ruteo así lo requieren) lo cual aumenta el grado de complejidad del diseño. También es importante que exista la menor cantidad de bloqueos en las celdas. Esto se refiere a que no debe haber metales de ruteo en la celda (M2 y M3), y a minimizar la cantidad de contactos alineados.

Los transistores nMOS se colocan en la parte inferior de la celda, con una altura de  $36\lambda$ , y los transistores pMOS se colocan en los superiores  $54\lambda$  (extensión del NWELL). Así, las celdas se pueden conectar superponiendo (*by abutment*) los rieles de alimentación y tierra y deben tener el NWELL coincidente. Los contactos de substrato y well son instanciados coincidentes con los rieles de alimentación. Las entradas y salidas se rutean en M2, que se extiende verticalmente. Cada celda es un múltiplo exacto de  $9\lambda$  de ancho, de modo que ofrece un número entero de pistas, *tracks*, de M2. Dentro de la celda, conviene rutear con POLY de forma vertical y con M1 de forma horizontal, aunque siempre no es simple seguir esta regla. En caso de ser necesario, es posible rutear dentro de las celdas con POLY en forma horizontal y M1 en forma vertical, en el caso de que no interfiera con otras conexiones.

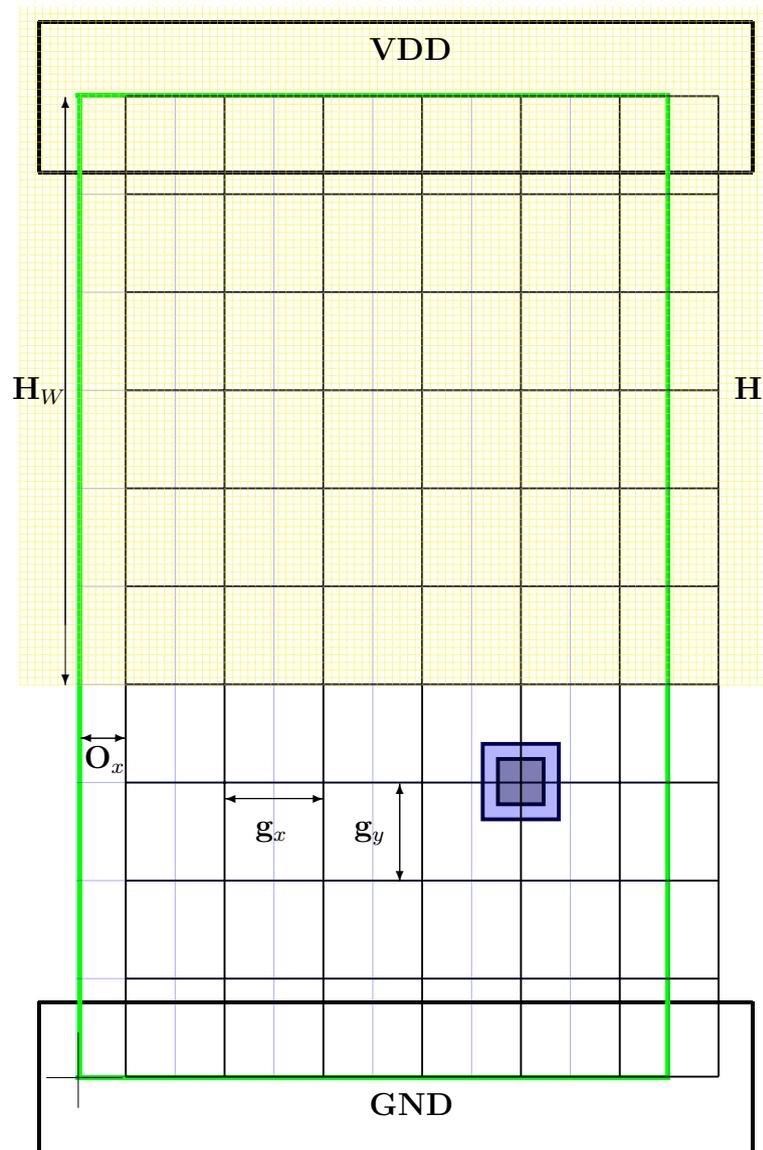


Figura 4.9: Grilla de ruteo.

#### 4.2.6. Dimensionamiento de los transistores

El dimensionamiento de los transistores es una etapa fundamental del diseño, puesto que esta característica definirá el desempeño en tiempo de los circuitos digitales. Idealmente, en diseños digitales, se desea tener siempre un *duty cycle* de 50%. Para ello, los tiempos de *rise* y *fall* deben ser iguales en cada celda. Estos tiempos dependen del fan-out de cada red complementaria

de cada celda y de la capacidad del nodo de salida. El modelo del transistor MOS presenta una ecuación para la corriente de drain donde la corriente es proporcional a  $\mu$  y  $W$ , como se muestra en la Ecuación 2.1. Como la movilidad de los electrones es mayor que la de los huecos en silicio, los transistores nMOS suelen tener mayor capacidad de corriente que los transistores pMOS. Por esta razón, es necesario aumentar el  $W$  de los transistores pMOS.

De esta forma, para dimensionar los transistores en las diferentes celdas digitales, se tomó como referencia el inversor mínimo. Así, se establece como

$$W_P^{min} = 2W_N^{min} = 2W^{min} \quad (4.4)$$

para reducir la diferencia en el *rise* y *fall time*, por las movilidad de los huecos y electrones.

#### 4.2.7. Inversor

Se define al inversor CMOS mínimo como el inversor que está compuesto por los transistores mínimos del proceso. La fotolitografía de los procesos escalables de MOSIS permiten diseñar transistores tipo dog bone, con un ancho de canal igual a  $3\lambda$ , el cual es el *minimum width* para el layer ACTIVE. Sin embargo, Para diseños analógicos o diseños digitales críticos, MOSIS recomienda, para el proceso SCN3ME\_SUBM, un ancho mínimo de canal de  $10\lambda$ . Si bien los dispositivos más estrechos son funcionales, sus características eléctricas no se pueden escalar hacia abajo, y su rendimiento no es predecible a partir de parámetros de SPICE que ofrece MOSIS [20].

Se estableció entonces el  $W^{min}$  en  $10\lambda$ , igual a  $3\mu m$ . Como se describió en la Sección 4.2.6, el ancho mínimo para el transistor nMOS del inversor mínimo se definió como

$$W_N^{min} = 10\lambda = 3\mu m \quad (4.5)$$

mientras que el transistor pMOS mínimo se definió como

$$W_P^{min} = 20\lambda = 6\mu m \quad (4.6)$$

En Figura 4.10 se observa la simulación de la transferencia de un inversor mínimo con las dimensiones de los transistores definidas (curva negra). Se observa que la respuesta es muy similar al comportamiento ideal, siendo  $V_M$  igual a  $\frac{V_{DD}}{2}$ . También se simuló los casos en donde los transistores nMOS (*bad nMOS*) y pMOS (*bad pMOS*) están sub-dimensionados.

Se diseñaron también diferentes inversores con mayor fan-out. Para la celda INVX2, se duplicaron los anchos de ambos transistores. Como el máximo espacio disponible para los transistores pMOS en grilla es  $54\lambda$ , no es posible instanciar transistores pMOS mayores a  $40\lambda$ , es decir  $12\mu\text{m}$ . Entonces, para los inversores de mayor capacidad de corriente que INVX2, se colocan transistores en paralelo.

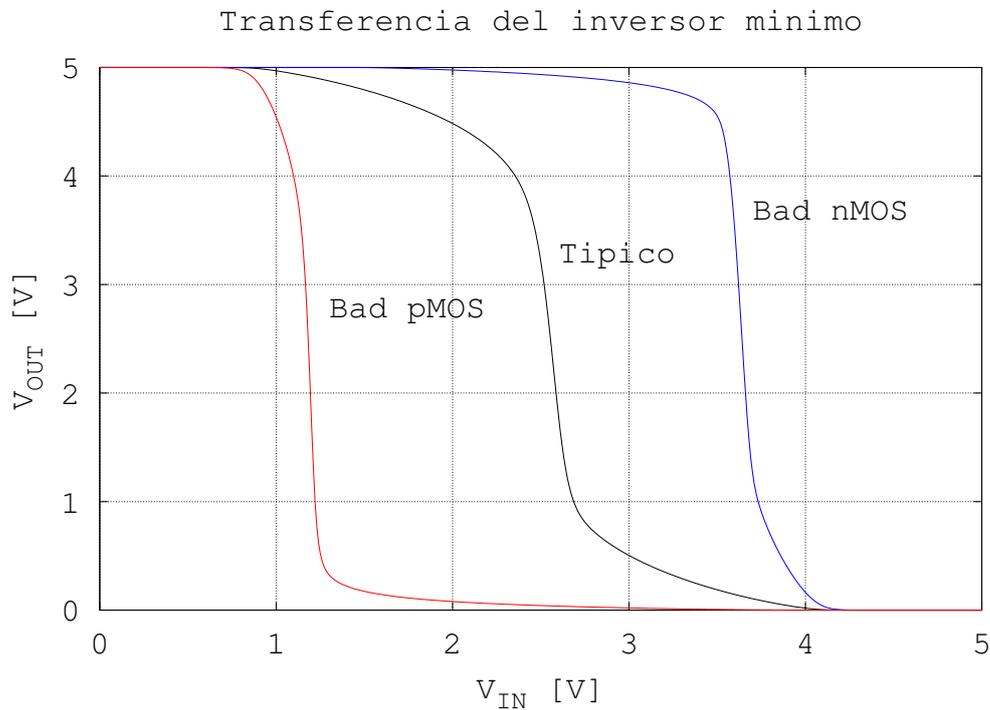


Figura 4.10: Simulación de la transferencia del inversor CMOS mínimo variando las características de los transistores.

Es fundamental conocer en detalle el comportamiento temporal y de polarización del inversor, para luego poder entender el comportamiento de otras estructuras de mayor complejidad, como se explicó en el Capítulo 2.

#### 4.2.8. Compuertas NAND y NOR

Se extendió el criterio presentando en la Sección 4.2.6 para dimensionar todas las redes *pull-up* y *pull-down* de cada celda. Por ejemplo, para una celda NAND de 2 entradas, el modelo considera la red *pull-down* como dos

resistores “tipo N” en serie, mientras que la red pull-up como dos resistores “tipo P” en paralelo. De esta forma,

$$W_N = 2W_N^{min} = 2W^{min} \quad (4.7)$$

y

$$W_P = W_P^{min} = 2W^{min} \quad (4.8)$$

Para una compuerta NOR de dos entradas, el modelo se constituye con dos resistores “tipo N” en paralelo y dos resistores “tipo P” en serie. Para este caso,

$$W_N = W_N^{min} = W^{min} \quad (4.9)$$

y

$$W_P = 2W_P^{min} = 4W^{min} \quad (4.10)$$

#### 4.2.9. Celdas compuestas

Una celda compuesta es una celda que realiza una función lógica compleja en una sola etapa, mediante la combinación de llaves en serie y paralelo. Por ejemplo, la derivación del circuito para la función

$$Y = (A \cdot B) + (C \cdot D) \quad (4.11)$$

se muestra en la Figura 4.11. Esta función se llama AND-OR-INVERT o AOI22, porque su salida es la inversión de un par de compuertas AND de 2 entradas. Para los nMOS de la red de pull-down, se toma la expresión no invertida  $(A \cdot B) + (C \cdot D)$  que indica cuando la salida debe ser cero. Las expresiones  $(A \cdot B)$  y  $(C \cdot D)$  deben ser implementadas por llaves conectadas en serie, como se muestra en la Figura 4.11 (a). Luego, se aplica una función OR al resultado, lo cual requiere la conexión en paralelo de estas dos estructuras, como se muestra en la Figura 4.11 (b).

Para la red de pull-up pMOS, debemos calcular la expresión complementaria utilizando llaves que se encienden con la polaridad invertida. Por ley de De Morgan, esto es equivalente a intercambiar las operaciones AND y OR. Por lo tanto, los transistores que aparecen en serie en la red pull-down deben aparecer en paralelo en la red pull-up. Los transistores que aparecen en paralelo en la red pull-down deben aparecer en serie en la red de pull-up.

En la red de pull-up, la combinación en paralelo de A y B se coloca en serie con la combinación en paralelo de C y D. Esta progresión es evidente en la Figura 4.11 (c) y en la Figura 4.11 (d). Al conectar las redes entre sí se genera el esquemático completo, como se observa en la Figura 4.11 (e). El símbolo se muestra en la Figura 4.11 (f) [10].

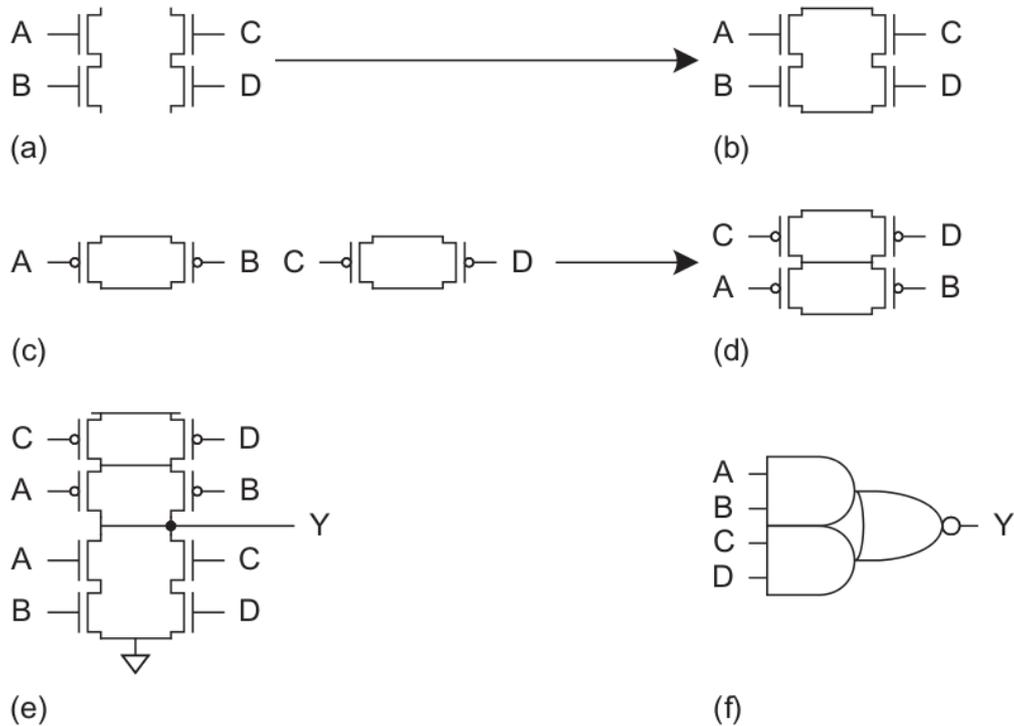


Figura 4.11: Celda CMOS compuesta para la función  $Y = (A \cdot B) + (C \cdot D)$  [10].

#### 4.2.10. Tri-states

La lógica *tri-state*, o lógica de tres estados, es aquella que permite tener una salida en donde existe un estado de alta impedancia además de los valores binarios tradicionales, el cual desconecta efectivamente la salida del resto del circuito. La Figura 4.12 muestra el símbolo de un buffer tri-state. Cuando la entrada de habilitación EN es igual a 1, la salida Y es igual a la entrada A, al igual que en un buffer ordinario. Cuando la entrada de habilitación es 0, la salida Y permanece flotante (alta impedancia).

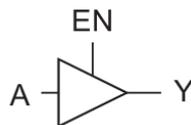


Figura 4.12: Buffer tri-state [10].

La Figura 4.13 (a) muestra un inversor tri-state. La salida es varía entre

VDD o GND como en un inversor tradicional. Cuando EN es 0, como se observa en la Figura 4.13 (b), ambos transistores de enable están apagados, dejando la salida flotante. Cuando EN es 1, como se detalla en la Figura 4.13 (c), ambos transistores de enable están encendidos. La Figura 4.13 (d) muestra los símbolos para el inversor de tres estados.

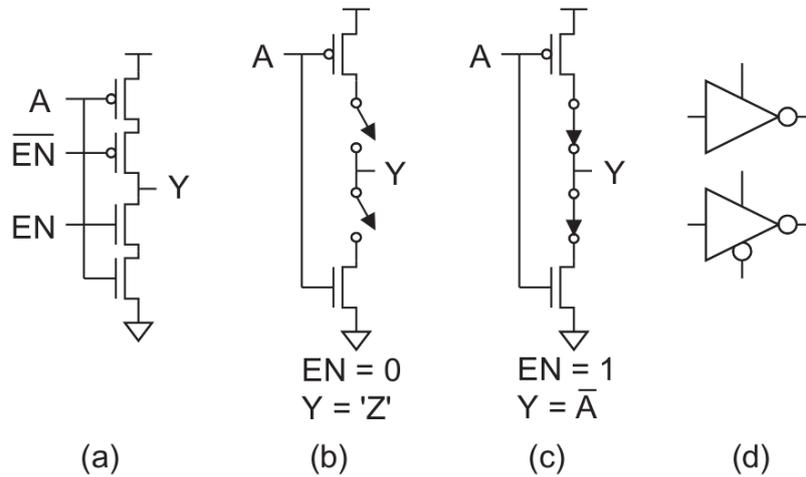


Figura 4.13: Inversor tri-state [10].

### 4.2.11. Multiplexores

Los multiplexores son circuitos combinatoriales con varias entradas y una única salida de datos, los cuales están dotados de señales de control capaces de seleccionar una, y sólo una, de las entradas de datos para permitir su transmisión desde la entrada seleccionada hacia dicha salida. Son componentes clave en elementos de memoria CMOS y en estructuras de manipulación de datos. En una celda “2-input MUX” o 2:1MUX, se elige la entrada D0 cuando S es igual a “0”; y la entrada D1 cuando S es igual a “1”, según la Figura 4.14. La función lógica se representa como

$$Y = S \cdot D0 + \overline{S} \cdot D1. \tag{4.12}$$

Dos llaves CMOS, denominadas *transmission gates*, pueden ser unidos entre sí para formar un multiplexor de área reducida de 2 entradas, como se muestra en la Figura 4.14 (a). La entrada de selección y su complemento habilitan una de las dos llaves en un momento dado.

La topología transmission gate produce un multiplexor *non-restoring*. Esto se puede corregir, añadiendo inversores, tanto en las entradas como en la

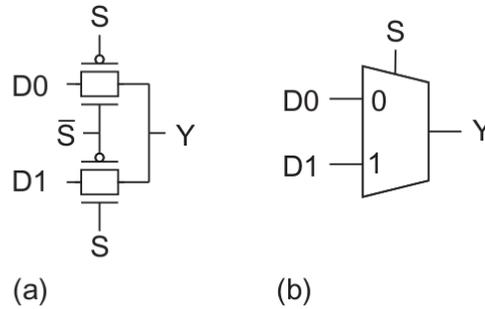


Figura 4.14: Multiplexor transmission gate [10].

salida común. Multiplexores de mayor cantidad de entradas pueden ser diseñados a partir de múltiples multiplexores de 2 entradas o interconectando entre sí varios inversores tri-state.

## 4.2.12. Circuitos secuenciales

### 4.2.12.1. Latches

Una topología posible para un latch D se muestra en la Figura 4.15 (a), construido a partir de un multiplexor de 2 entradas y dos inversores. El multiplexor puede ser construido a partir de un par de llaves MOS, como se muestra en la Figura 4.15 (b). Esta microarquitectura también produce una salida complementaria,  $\bar{Q}$ . Cuando el reloj está en estado alto, el latch es transparente y la entrada D se propaga directamente en la salida Q, como se ve en la Figura 4.15 (c). Cuando la señal de reloj cae a cero, la entrada D queda aislada del circuito. Se establece un camino de realimentación que contiene al par de inversores (Figura 4.15 (d)) para mantener el estado actual de Q indefinidamente.

El latch D también se conoce como “latch sensible a nivel” debido a que el estado de la salida es dependiente del nivel de la señal de reloj, como se muestra en la Figura 4.15 (e). El latch que se muestra es sensible al nivel positivo, representado por el símbolo en la Figura 4.15 (f). Invertiendo las conexiones de control del multiplexor, el latch se convierte a nivel sensible negativo.

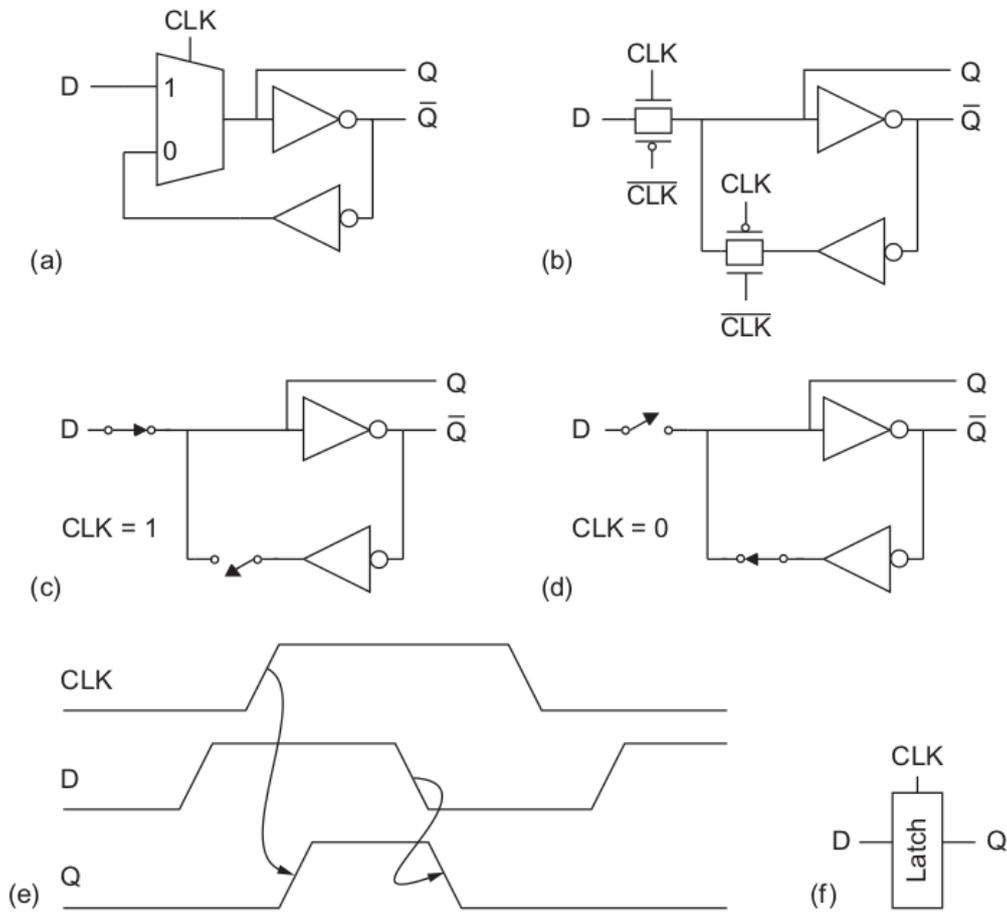


Figura 4.15: Latch D [10].

#### 4.2.12.2. Flip-flops

Al combinar dos latches D, uno con sensibilidad negativa y otro positiva, se construye el flip-flop D, disparado por flanco, como se muestra en la Figura 4.16 (a-b). Esta es la topología master-slave presentada en la Figura 2.17a. Mientras la señal de reloj se mantiene en cero, la salida del latch maestro (*negative-level-sensitive latch*),  $\bar{Q}\bar{M}$ , sigue a la entrada D, mientras que el latch esclavo (*positive-level-sensitive*) mantiene el valor previo (Figura 4.16 (c)). Cuando el reloj cambia de 0 a 1, el latch maestro se realimenta y mantiene el valor D en el momento de la transición de reloj. El latch esclavo a su vez se vuelve transparente, transfiriendo el valor almacenado QM a la salida Q del latch esclavo. La entrada D está bloqueada y no afecta a la salida porque el latch maestro se desconecta de la entrada (Figura 4.16 (d)).

Cuando el reloj pasa de 1 a 0, el latch esclavo mantiene su valor y el maestro empieza el muestreo de la entrada nuevamente.

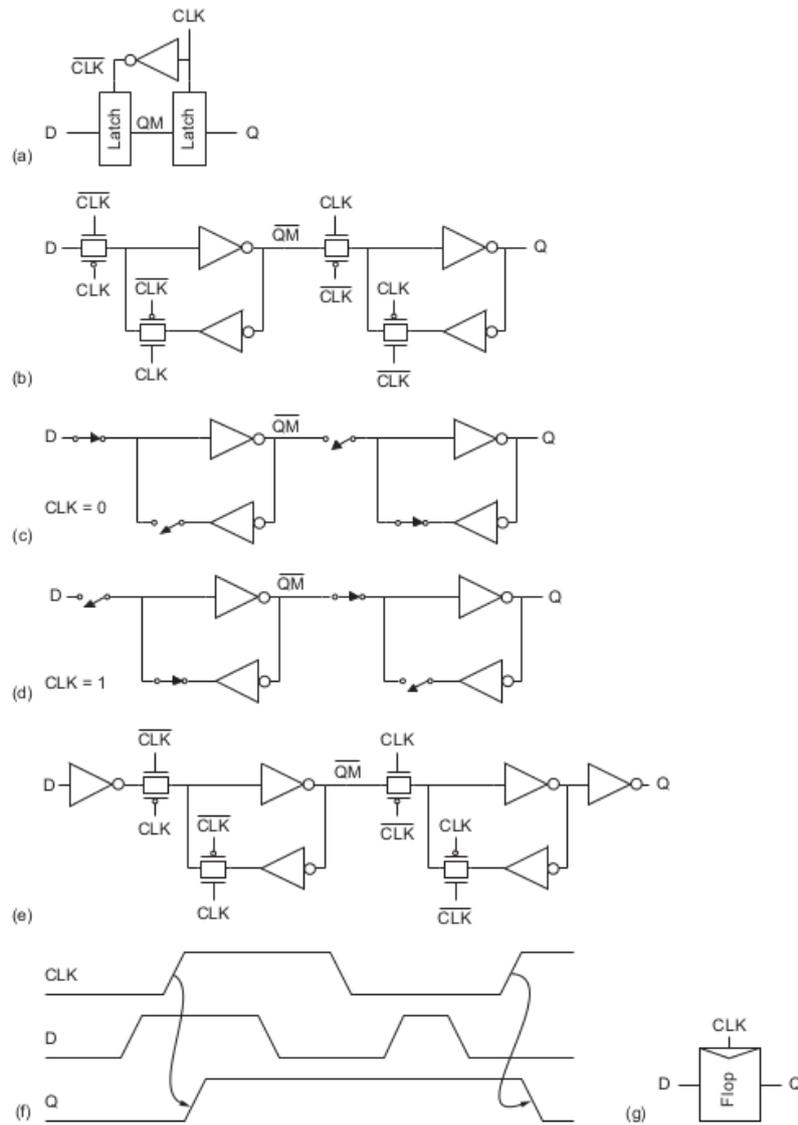


Figura 4.16: Flip-flop D [10].

Para el diseño del flip-flop D básico se utilizó la topología TGSM, con una variación para poder introducir el terminal de reset: se cambió el inversor de la realimentación del latch master y el inversor de la salida del latch esclavo por una compuerta NOR, donde una de las entradas es el terminal de reset,

en éste caso, asincrónico, como se observa en la Figura 4.17. La simulación se detalla en la Figura 4.18.

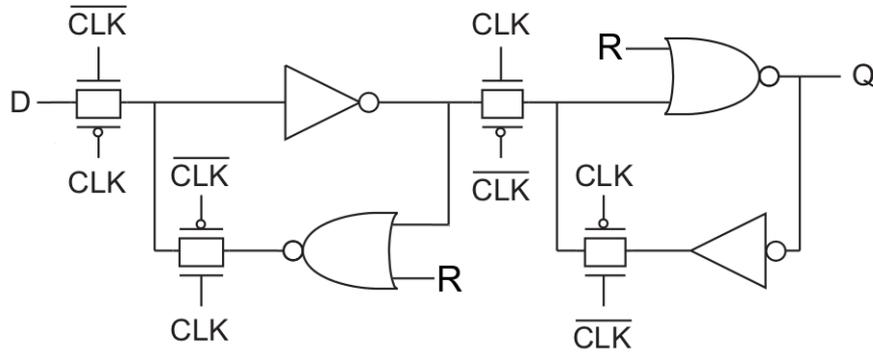


Figura 4.17: Esquemático de la celda DFFR.

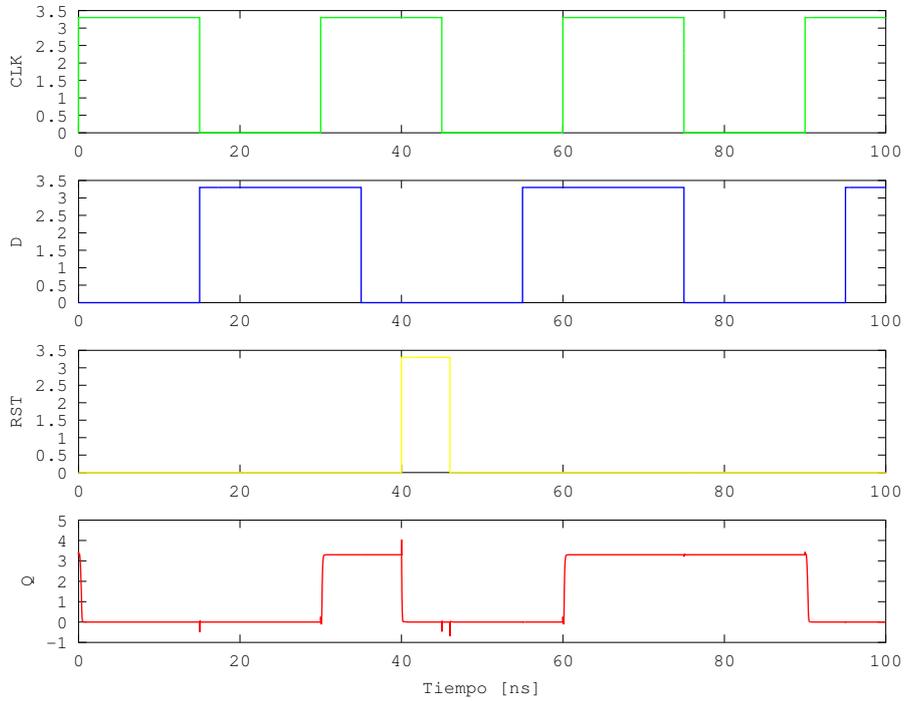


Figura 4.18: Simulación estática de la celda DFFR.

### 4.3. Extracción de parásitos

La extracción de parásitos es un proceso en donde se calculan los efectos parásitos dentro de un circuito integrado, tanto en los dispositivos diseñados como en las interconexiones. El propósito principal de la extracción es crear un modelo analógico preciso del circuito, para poder realimentar esta información para realizar simulaciones minuciosas, que puedan emular las respuesta de los circuitos digitales y analógicos reales.

Idealmente, se consideran a las interconexiones como cables que solo conectan a los diferentes elementos funcionales (sean dispositivos, compuertas o bloques funcionales) y que no afectan el rendimiento del diseño. Este supuesto es prácticamente cierto para la mayoría de los circuitos de “grandes dimensiones” (o de bajas frecuencias), pero es inaceptable para el diseño de circuitos integrados. En la realidad, todas éstas interconexiones presentan resistencias, capacidades e inductancias, según la geometría del diseño. Estos parásitos impactan en los tiempos de propagación, consumo y generan pérdidas de potencia, al mismo tiempo que se convierten en fuentes de ruido, lo que afecta la confiabilidad del diseño.

### 4.4. Conclusiones

En el presente capítulo se presentaron las generalidades del proceso de diseño de las celdas estándar, comenzando por la definición del diseño VLSI de celdas estándar e introduciendo conceptos de layout y de diseño digital. Se lograron cumplir con éxito todas las etapas de diseño: definición de la topología, dimensionamiento de dispositivos, simulación, diseño de layout, verificación del diseño y extracción de parásitos.

# Capítulo 5

## Resultados

En este capítulo se presentan los diferentes resultados obtenidos a lo largo del trabajo de tesis. En primera instancia, se muestran los resultados que se desprenden del trabajo de diseño del PDK, como son las celdas paramétricas, los archivos de tecnología y las reglas de diseño para verificación. Luego, se presentan los resultados obtenidos a partir del diseño de las celdas estándar. Para ésto, se desarrollaron diferentes estructuras de testeo, las cuales fueron fabricadas en un chip a través de MOSIS.

### 5.1. PDK

Se diseñó un juego de celdas paramétricas, escritas en python. En ésta primera etapa, se desarrollaron los transistores MOS, necesarios para el diseño de las celdas estándar. A su vez, se diseñó también una resistencia, con múltiples parámetros, ampliando así el iPDK para uso tanto digital como analógico. En las Figuras 5.1a y 5.1b se muestran los layout de las pycells de los transistores.

A su vez, en la Figura 5.2 se presenta la pycell de la resistencia HR mínima, mientras que en las Figuras 5.3a y 5.3b se muestran resistencias con tres *fingers*, conectados en serie y paralelo, respectivamente.

La resistencia diseñada se construye con POLY2, y luego con la máscara HR (del inglés *high resistance implant*) se bloquea un implante para bajar la resistencia.

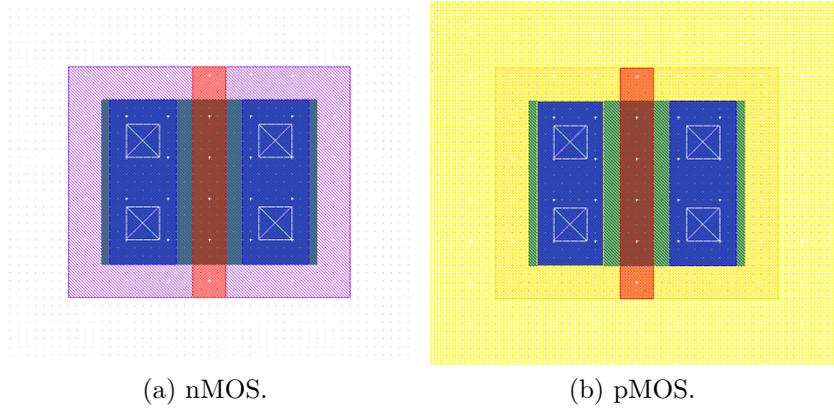


Figura 5.1: Layout de las PyCells de los transistores MOS.

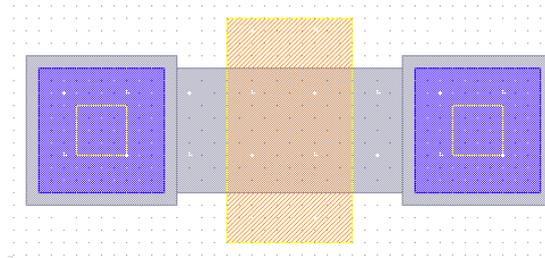


Figura 5.2: Layout de la PyCell básica de la resistencia HR.

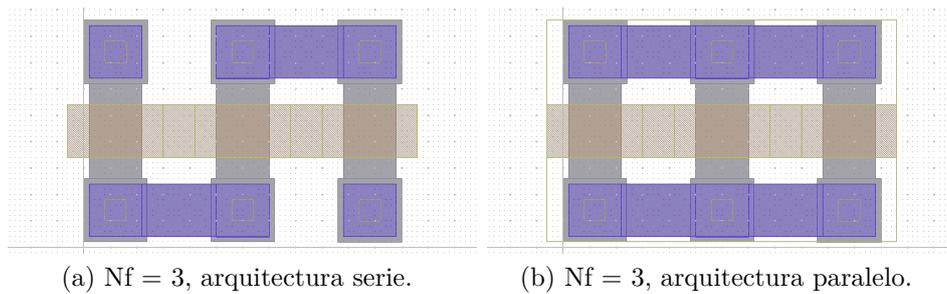


Figura 5.3: Layout de la PyCell de la resistencia HR con algunos parámetros modificados respecto de los parámetros por defecto.

## 5.2. Celdas Estándar

Como resultado de la etapa de diseño de celdas estándar, se diseñaron 28 celdas. El layout de las mismas se muestra en el Anexo A.

PyCells			
	Parámetro	Valor por defecto	Descripción
nMOS	model	N	Modelo LVS
	spice_model	ami06n	Modelo SPICE
	w	0.9 $\mu m$	Width
	l	0.6 $\mu m$	Length
pMOS	model	P	Modelo LVS
	spice_model	ami06p	Modelo SPICE
	w	0.9 $\mu m$	Width
	l	0.6 $\mu m$	Length
HR_res	model	3.63 m	Modelo SPICE
	specBy	L & H	Parámetros que se utilizan para setear el valor de la resistencia
	L	1.5 $\mu m$	Largo del finger
	H	1.5 $\mu m$	Alto del finger
	nf	1	Número de fingers
	rsq	1000 $\Omega$	Resistividad
	arch	Serie	Arquitectura
	R	1000 $\Omega$	Resistencia

Tabla 5.1: Parámetros de las pycells diseñadas.

### 5.2.1. Nomenclatura

Para nombrar las diferentes celdas estándar, se estableció una nomenclatura, de forma de relacionar el nombre de las mismas con su función, como así también con ciertas características eléctricas. Para esto, se dividieron las celdas en grupos funcionales.

**Funciones lógicas** <name>\_<number\_of\_inputs>X<driving\_factor>  
**Flip-flops** <flip-flop type>FF<special char>  
**Celdas especiales** <name><special char>

### 5.2.2. Parámetros globales

En esta biblioteca, todos los pines están situados en la intersección de las grillas horizontales y verticales. La mayoría de las herramientas de place and

route trabajan más eficientemente con todos los pines en grilla, mientras que algunas herramientas aun así lo demandan.

Largo del canal [ $\mu m$ ]	0.6
Layers de metal (interno)	M1
Layers de metal (ruteo)	M2 y M3
Grilla de layout [ $\mu m$ ]	2.7
Pitch total [ $\mu m$ ]	27
Ancho de rieles de alimentación y tierra [ $\mu m$ ]	1.5

Tabla 5.2: Especificaciones físicas.

## 5.3. Resultados experimentales

Para corroborar el funcionamiento de las celdas generadas se diseñaron algunas estructuras de prueba. Se fabricó un *tiny chip* a través de MOSIS con éstas diferentes estructuras, con el objetivo de comprobar el funcionamiento lógico de las celdas e inferir el retardo de propagación y otras características temporales.

### 5.3.1. Estructuras de testeo

Como se observa en la Figura 5.4, se diseñaron dos grandes estructuras para poder medir algunas de las características del PDK y de las celdas. En primer lugar, se diseñó un chip que cumplió con las reglas de DRC y LVS escritas durante éste trabajo, así como también cumplió con las reglas de diseño de MOSIS, posibilitando su fabricación. En la Figura 5.4a se observa el diagrama en bloques de la matriz de celdas estándar que se fabricó, con el fin de corroborar el funcionamiento de las mismas. Debido a la capacidad de carga que presentan los pines y el bonding, no se pueden realizar de forma directa mediciones dinámicas ni análisis temporales.

Por su parte, como se muestra en la Figura 5.4b, se diseñaron cuatro osciladores en anillo de diferentes largos, para poder estimar el retardo de un inversor. Los cuatro osciladores se diseñaron con inversores mínimos, y tienen 257, 193, 127 y 65 etapas cada uno. Los mismos son multiplexados y su frecuencia se divide en una cadena de ocho flip-flops, para facilitar la medición y el diseño del buffer de salida.

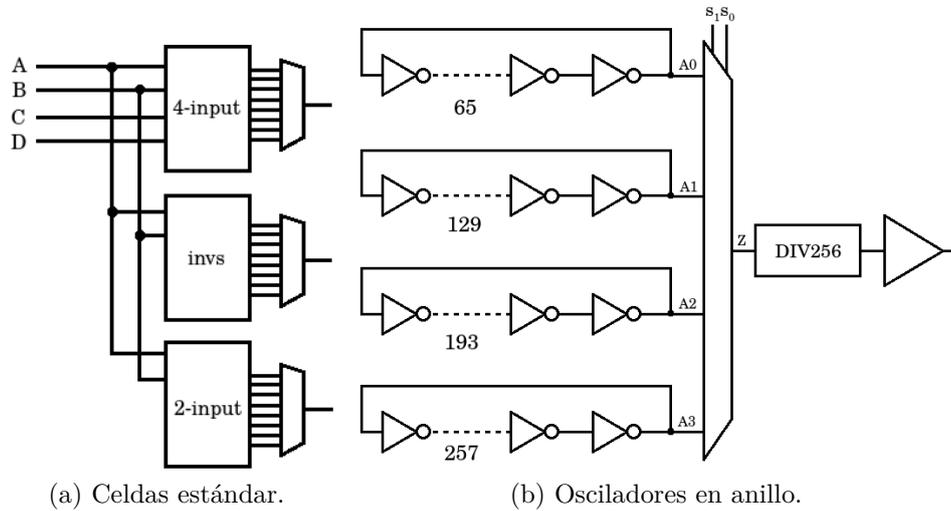


Figura 5.4: Diagrama en bloques de las estructuras de testeo diseñadas.

En la Figura 5.5 se puede observar el *top level* del chip diseñado. En la Figura 5.5a se muestra el diseño a nivel layout, mientras que en la Figura 5.5b se muestra el diseño físico visto en un microscopio óptico.

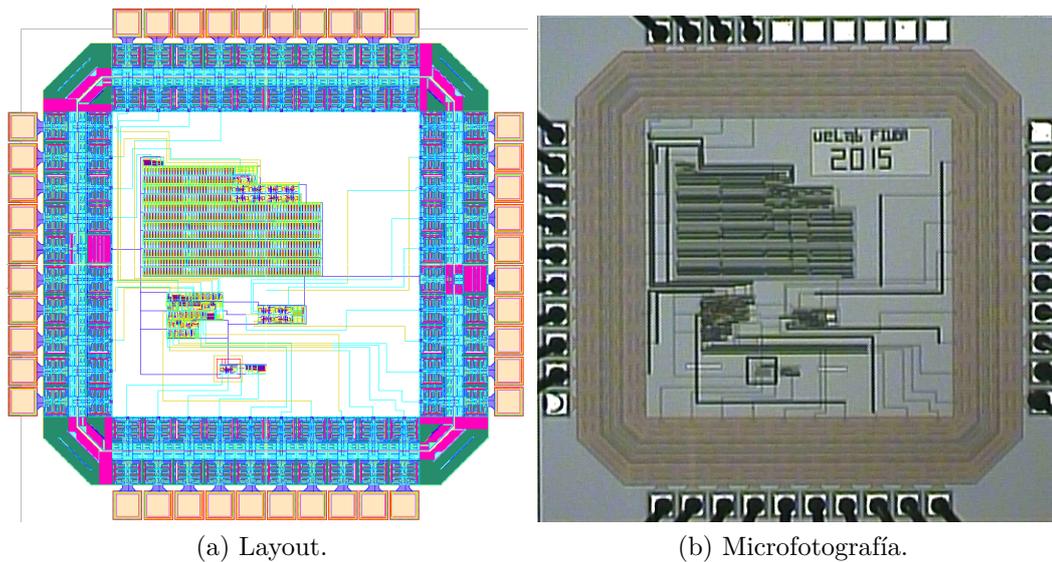


Figura 5.5: *Top level* del tiny chip fabricado por MOSIS.

### 5.3.2. Análisis estático

Para probar el funcionamiento estático de las celdas, es decir, que cumplan su función booleana, se diseñó una matriz de celdas estándar, como se muestra en la Figura 5.4a, donde se describe el diagrama en bloques del diseño. Las entradas A, B, C y D están compartidas por todas las celdas, dividiéndose en tres grupos de compuertas: de cuatro entradas, de dos entradas e inversores. En el primer grupo hay 8 celdas, y en el segundo y el tercero hay 6. Las salidas se multiplexan, utilizándose 9 entradas de control y tres salidas.



Figura 5.6: Banco de mediciones estáticas.

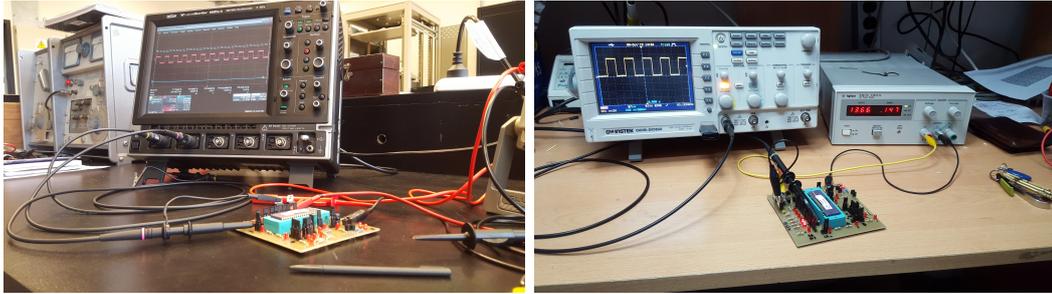
Para realizar las pruebas, se construyó un PCB con leds para verificar rápidamente que se cumpla la tabla de la verdad de las diferentes compuertas, a la vez que las entradas también tienen un led que indica su estado. Se comprobó así el correcto funcionamiento estático de todas las celdas.

En la Figura 5.6a se muestra una foto con los cinco chips fabricados por MOSIS. En la Figura 5.6b se muestra el chip número uno en el PCB de pruebas.

### 5.3.3. Análisis dinámico

Con el fin de estimar el tiempo de propagación del inversor mínimo, se diseñaron cuatro osciladores en anillo de diferente cantidad de etapas, según se detalla en el diagrama en bloques de la Figura 5.4b. Las entradas de selección s1 y s2 corresponden al multiplexor de salida, que selecciona cada uno de los diferentes osciladores. A la salida de éste, se colocó una cadena de ocho flip-flops con el fin de dividir la frecuencia de salida, para simplificar el buffer de salida y facilitar la medición.

La Figura 5.7 muestra el banco de medición donde se observa la placa con el chip, la fuente de alimentación y el osciloscopio con la señal de uno de los osciladores en anillo.



(a) Chips.

(b) Medición.

Figura 5.7: Banco de mediciones dinámicas.

Para calcular el tiempo de propagación se midió la frecuencia de cada oscilador, sabiendo que este resultado es en cada caso 256 veces menor a la frecuencia de cada oscilador

$$f_{real} = f_{medida} \times 256 \quad (5.1)$$

Luego,

$$t_d = \frac{1}{2 \times f_{real} \times m}, m = \begin{cases} 65, & s = 00 \\ 129, & s = 01 \\ 193, & s = 10 \\ 257, & s = 11 \end{cases} \quad (5.2)$$

La Tabla 5.3 muestra el resultado de las mediciones de los osciladores en anillo. Las mismas se realizaron utilizando los osciloscopios “INSTEK GDS-2062” y “LeCroy waveSurfer 64MXs-A” y luego procesando los datos en Octave.

### 5.3.3.1. Análisis de los resultados

Se puede afirmar que el tiempo de propagación medio para el inversor mínimo, definido con un  $W_P$  de  $6\mu\text{m}$  y con un  $W_N$  de  $3\mu\text{m}$ , es 120ps. Para la corrida v53v, MOSIS no proveyó modelos de SPICE. De ésta forma, se toma como referencia los valores medidos por MOSIS de los osciladores en anillo. En el Anexo D se presenta el *test data* de la corrida. MOSIS midió dos

s1	s0	Z	Etapas	Ref.	Frec. medida	$t_d$ estimado
0	0	A0	65	osc1	250kHz	119.23ps
0	1	A1	129	osc2	125kHz	120.15ps
1	0	A2	193	osc3	83kHz	120.47ps
1	1	A3	257	osc4	63kHz	120.60ps

Tabla 5.3: Resultados de las mediciones de los osciladores en anillo.

osciladores en anillo, con dos tamaños de transistores diferentes. En la Tabla 5.4 se presenta los resultados y el cálculo del tiempo de propagación. Según lo presentando por MOSIS, las mediciones se realizaron también con 5V. Es importante destacar que no se puede hacer una comparación directa entre las mediciones efectuadas y lo medido por MOSIS, ya que las dimensiones de los transistores son diferentes.

Para el inversor mínimo, con  $W$  igual a  $3\mu\text{m}$  para ambos transistores, el tiempo de propagación es 163ps, 35% mayor a la medición realizada. Para el inversor *WIDE*, con  $W$  igual a  $20\mu\text{m}$  para ambos transistores, el tiempo de propagación es 103ps, 14% menor a la medición realizada. El dimensionamiento de los transistores afecta el fan-out de salida del inversor, lo que genera las variaciones en los tiempos de propagación.

Ref.	W/L [ $\mu\text{m}$ ]	Etapas	Frecuencia	$t_d$
MINIMUM	3.0/0.6	31	98.91 MHz	163ps
WIDE	20.0/0.6	31	155.97 MHz	103ps

Tabla 5.4: Resultados de las mediciones de los osciladores en anillo realizadas por MOSIS.

## 5.4. Conclusiones

En éste capítulo se presentaron los resultados más importantes del trabajo realizado, la fabricación un chip funcional a través de MOSIS utilizando el PDK y las celdas estándar desarrolladas. Luego se midieron y verificaron los resultados obtenidos.

En resumen, se verificó el correcto funcionamiento de:

- iPDK

- Celdas paramétricas
  - Archivos de tecnología
  - Archivos de verificación
  - Archivos iCDF
  - Menú de configuración del PDK
- Celdas estándar: se diseñaron un total de 28 celdas
  - Estructuras de testeo
    - Análisis estático
    - Análisis dinámico



# Capítulo 6

## Resumen y Conclusiones

En el presente trabajo se ha llevado a cabo el flujo completo de diseño de un kit de diseño interoperable, desde el diseño de los transistores, callbacks, reglas de diseño, hasta el estudio, diseño, simulación, fabricación y testeado de estructuras de interés para la caracterización del PDK. A su vez, utilizando el kit, se diseñaron un conjunto de celdas estándar digitales, las cuales también fueron fabricadas para su verificación.

A lo largo de todo el trabajo de tesis hubo que sortear diferentes obstáculos hasta llegar al diseño final y funcional, tanto de las celdas como del PDK. El trabajo comenzó con el estudio a fondo de las reglas de diseño de los procesos provistos por MOSIS. En primera instancia, se debió optar por un proceso propietario o un proceso escalable. Luego, se dedicó una gran cantidad de tiempo a estudiar cada una de las reglas.

Posteriormente, con un acabado conocimiento de las reglas del proceso, se continuó con el diseño de las PCells básicas de los transistores MOS, escritas en Python. Para esto, se debió analizar el entorno de diseño del PyCell Studio y escribir el archivo de tecnología correspondiente.

Al finalizar el proceso de diseño de las celdas paramétricas en Python, se debió comprender el flujo de diseño analógico. Para esto, se utilizaron los procesos SAED90 y SAED32<sup>1</sup>. A partir de aquí, se comenzó el proceso de diseño de las celdas estándar, en una continua realimentación con el diseño de callbacks y archivos de verificación.

Se diseñaron 28 celdas estándar utilizando el iPDK desarrollado, entre las que se incluyen funciones lógicas básicas (como NAND, NOR, XNOR, XOR), celdas secuenciales (flip-flop D), buffers, inversores y *fillers*.

---

<sup>1</sup>Los procesos SAED son procesos ficticios generados por Synopsys con fines educativos

Se diseñaron estructuras de testeo, las cuales fueron fabricadas a través de MOSIS y verificadas.

El resultado de todo este trabajo fue un PDK funcional, interoperable y siguiendo los lineamientos de la IPL Alliance y la comunidad OpenAccess y un conjunto de celdas estándar.

## 6.1. Trabajos a futuro

- Continuar el desarrollo de las pycells para agregar más dispositivos, como ser:
  - Capacitores.
  - Inductores.
  - TBJ.
- Continuar el desarrollo de las pycells para agregar más características, como ser:
  - AutoAbutment.
  - Fingers.
- Continuar el desarrollo de las celdas estándar para aumentar la cantidad de éstas.
- Caracterizar la librería digital.
- Dejar el material libre disponible para la comunidad.
- Escribir el manual del PDK.
- Escribir el manual de las celdas estándar.

# Apéndice A

## Layout de las Celdas Estándar

En esta sección se presentan los layout físicos de las celdas estándar desarrolladas durante la tesis, agrupadas según su función lógica.

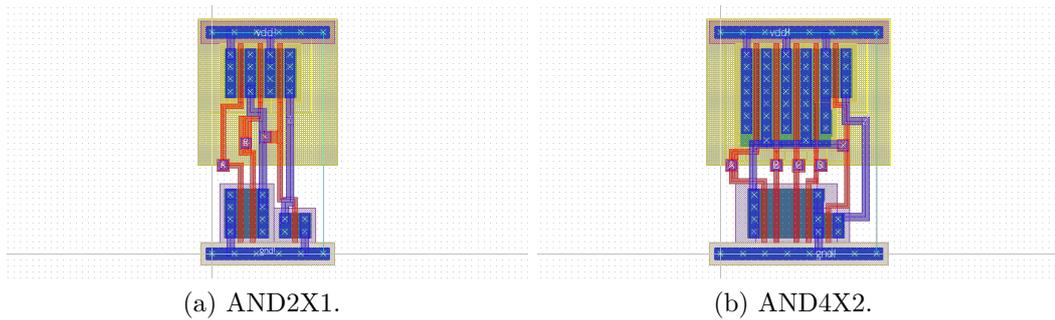


Figura A.1: Celdas AND.

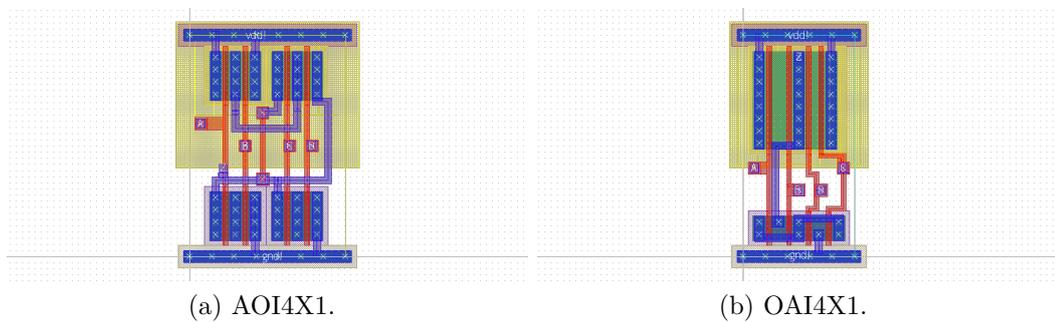


Figura A.2: Celdas compuestas.

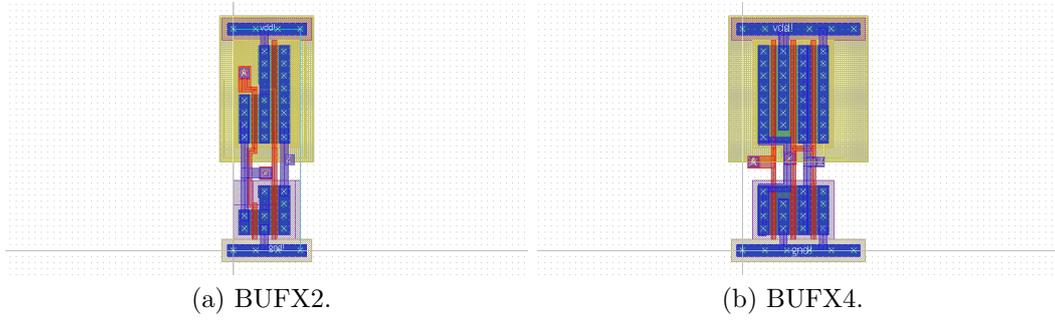


Figura A.3: Buffers.

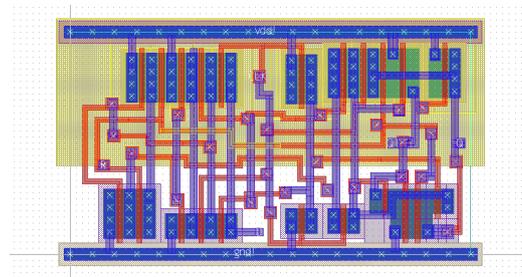


Figura A.4: Layout del DFFR.

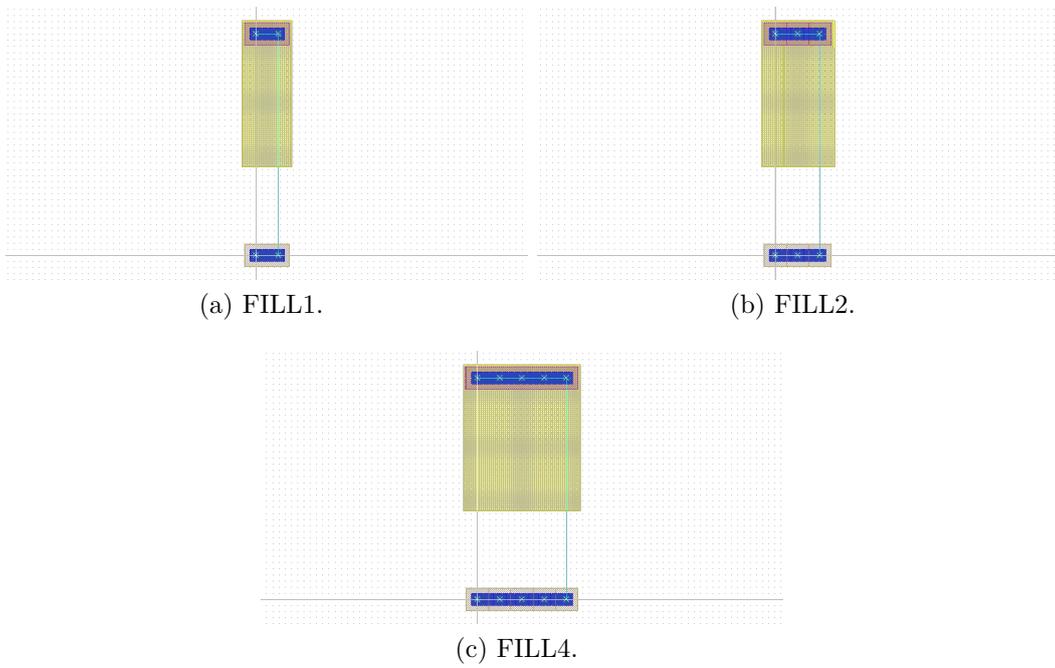


Figura A.5: Fillers.

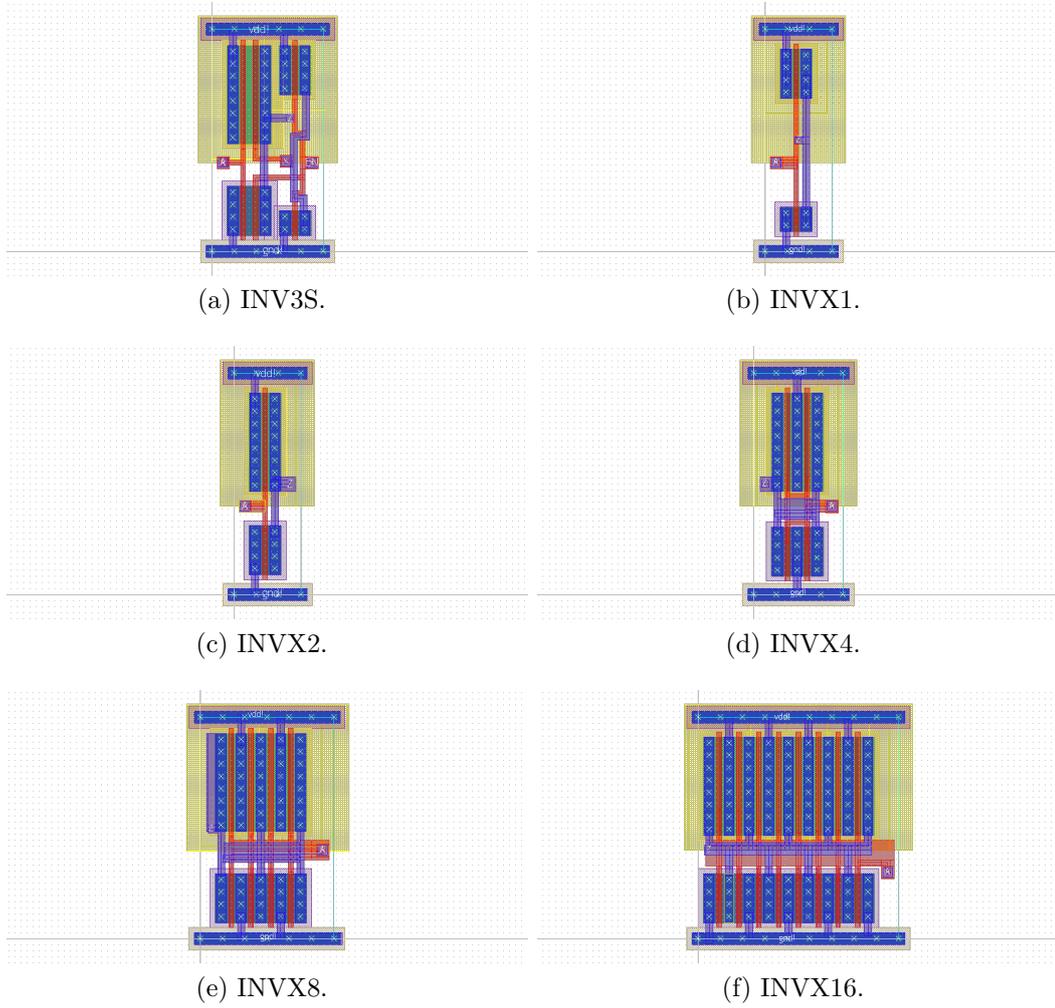


Figura A.6: Inversores.

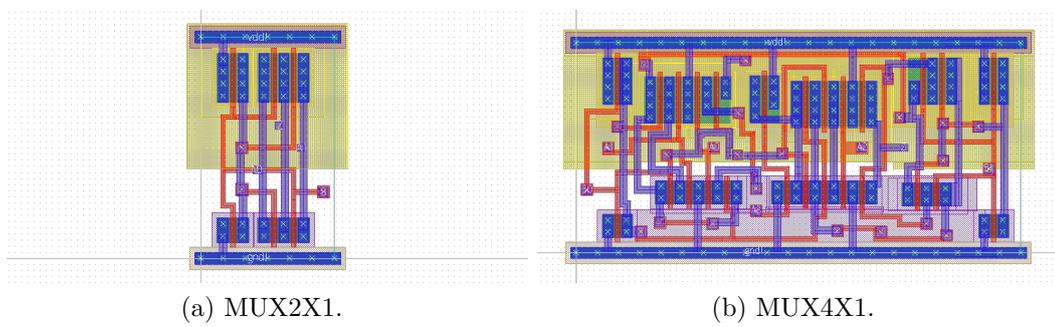


Figura A.7: Multiplexores.

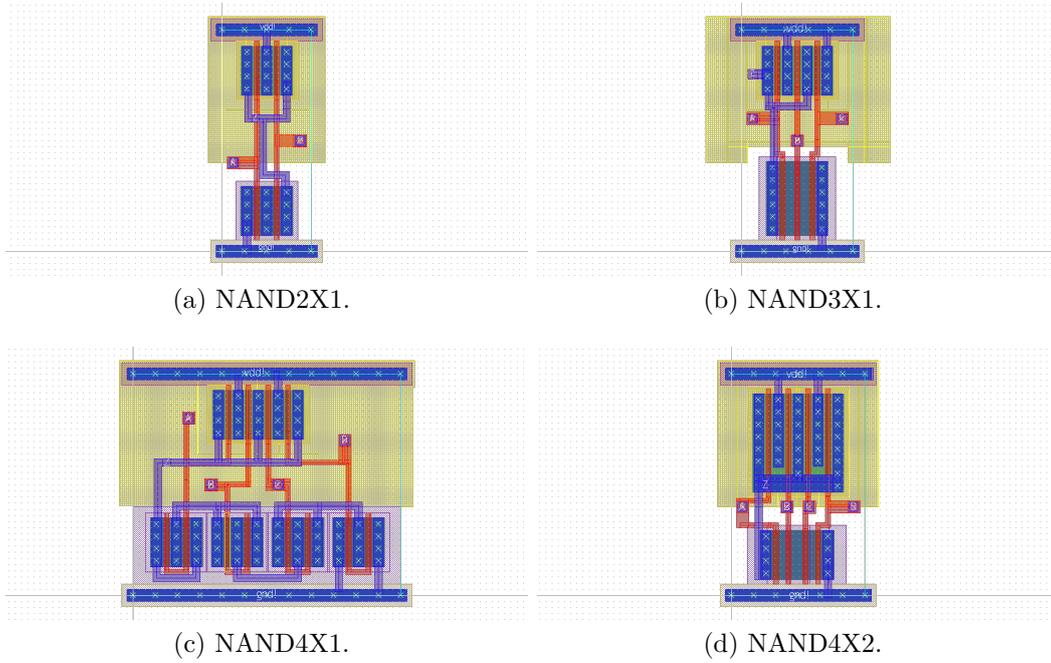


Figura A.8: Celdas NAND.

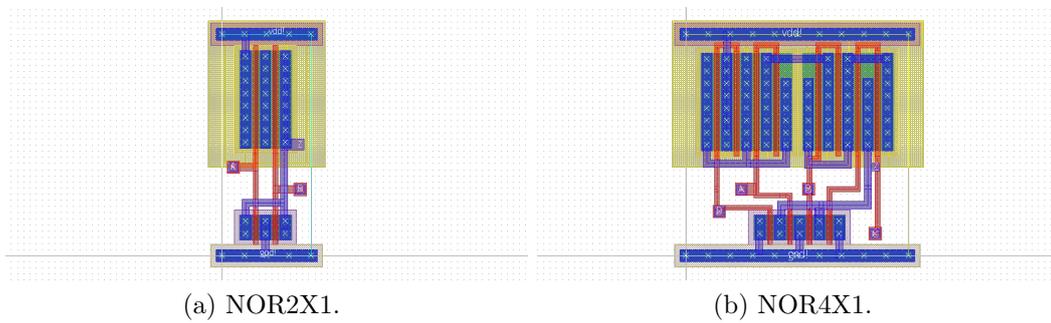


Figura A.9: Celdas NOR.

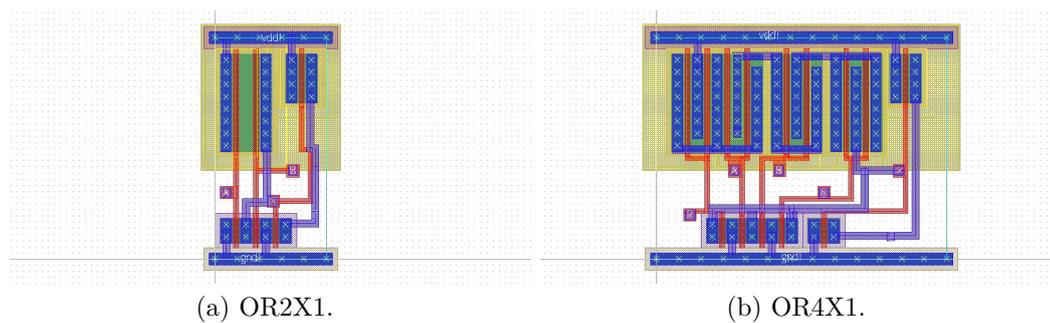


Figura A.10: Celdas OR.

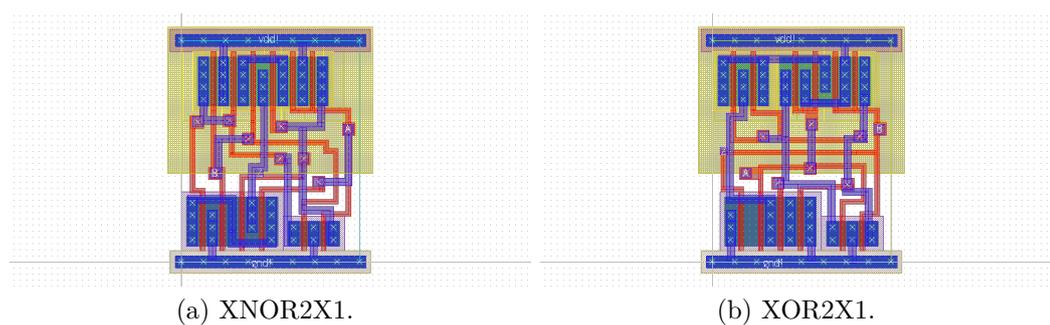


Figura A.11: Celdas XOR.



# Apéndice B

## Mediciones

### B.1. Osciladores en anillo

Se sabe que la frecuencia de oscilación de un oscilador en anillo responde a la siguiente ecuación:

$$f_{osc} = \frac{1}{2 \times t_d \times m} \quad (\text{B.1})$$

A su vez, a la salida de los osciladores hay un divisor de frecuencia. Así, la frecuencia real difiere de la frecuencia medida en un factor de 256 veces. Para obtener la frecuencia real

$$f_{real} = f_{measured} \times 256 \quad (\text{B.2})$$

luego, se estima el tiempo de propagación dependiendo de la cantidad de etapas de cada oscilador

$$t_d = \frac{1}{2 \times f_{real} \times m}, m = \begin{cases} 65, & s = 00 \\ 129, & s = 01 \\ 193, & s = 10 \\ 257, & s = 11 \end{cases} \quad (\text{B.3})$$

siendo  $t_d$  el tiempo de propagación y  $m$  la cantidad de etapas.

Se realizaron 60 mediciones de frecuencias para los osciladores, siendo una por cada oscilador de cada chip, para cada uno de los cinco chips, repitiendo esto tres veces. Se prefirió medir así para eliminar los errores sistemáticos al medir un mismo banco en reiteradas oportunidades y en poco tiempo. Se tomaron como válidos los valores que se agrupan en el valor modal, con un error menor al 10%. Las mediciones se presentan en la Tabla B.1. La

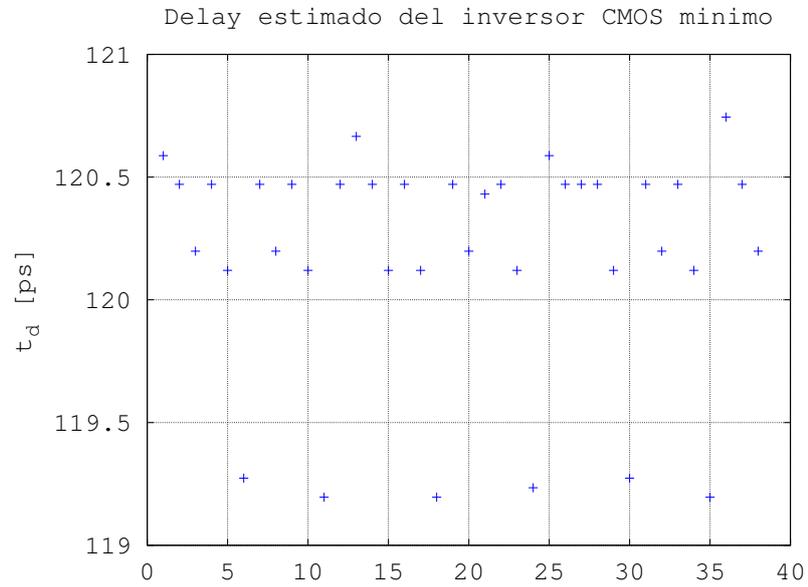


Figura B.1: Tiempo de propagación estimado a partir de las mediciones de frecuencia de los diferentes osciladores en anillo.

frecuencia que se muestra corresponde a la frecuencia real, no la medida y la cantidad de etapas se corresponden con cada uno de los cuatro osciladores. Los valores  $i$  y  $j$  son la referencia para las mediciones realizadas.

El promedio de todas estas mediciones es 120.19ps, con un desvío estándar de 0.4552ps.

<b>i</b>	<b>j</b>	<b>Frecuencia [MHz]</b>	<b>Etapas</b>	<b><math>t_d</math> [ps]</b>
1	0	16,1337	257	120,587
1	1	21,5047	193	120,47
1	2	32,2465	129	120,198
1	5	21,5047	193	120,47
1	6	32,2675	129	120,12
1	7	64,493	65	119,273
1	9	21,5047	193	120,47
1	10	32,2465	129	120,198
1	13	21,5047	193	120,47
1	14	32,2675	129	120,12
1	15	64,535	65	119,196
1	17	21,5047	193	120,47
2	0	16,1233	257	120,666
2	1	21,5047	193	120,47
2	2	32,2675	129	120,12
2	5	21,5047	193	120,47
2	6	32,2675	129	120,12
2	7	64,535	65	119,196
2	9	21,5047	193	120,47
2	10	32,2465	129	120,198
2	12	16,1547	257	120,431
2	13	21,5047	193	120,47
2	14	32,2675	129	120,12
2	15	64,514	65	119,235
2	16	16,1337	257	120,587
2	17	21,5047	193	120,47
3	1	21,5047	193	120,47
3	5	21,5047	193	120,47
3	6	32,2675	129	120,12
3	7	64,493	65	119,273
3	9	21,5047	193	120,47
3	10	32,2465	129	120,198
3	13	21,5047	193	120,47
3	14	32,2675	129	120,12
3	15	64,535	65	119,196
3	16	16,1128	257	120,744
3	17	21,5047	193	120,47
3	18	32,2465	129	120,198

Tabla B.1: Resultados de las mediciones de los diferentes osciladores en anillo.



# Apéndice C

## Capacidades parásitas

En el presente anexo se explican brevemente las capacidades parásitas del transistor MOS. Este análisis es de suma importancia para entender el comportamiento temporal de la lógica CMOS. Se extrae de [15 Ch. 5].

### C.1. Computando capacidades

Las capacidades parásitas del par en cascada de inversores CMOS se observan en la Figura 2.15. Se utiliza ese esquemático como referencia para describir los parásitos.

#### C.1.1. Capacidad gate-drain $C_{gd12}$

Los transistores M1 y M2 se pueden encontrar en corte o saturación durante la primera mitad (hasta el 50%) del transitorio de salida. Bajo estas circunstancias, las únicas contribuciones a  $C_{gd12}$  son las capacidades de *overlap* de ambos M1 y M2. La capacidad del canal de los transistores MOS no juega un papel preponderante aquí, ya que se encuentra o por completo entre gate y bulk (en corte) o gate y source (e saturación).

El modelo de capacidad agrupada requiere que este capacitor gate-drain flotante sea sustituido por un capacitor a tierra. Esto se logra por efecto Miller. Durante una transición bajo-alto o alto-bajo, los terminales del capacitor de gate-drain se mueven en direcciones opuestas, como se muestra en al Figura C.1. El cambio de voltaje sobre el capacitor flotante es, por tanto, el doble de la oscilación de voltaje de salida real. Para presentar una carga

idéntica a la del nodo de salida, el capacitor a tierra debe tener un valor que sea el doble de la capacidad flotante.

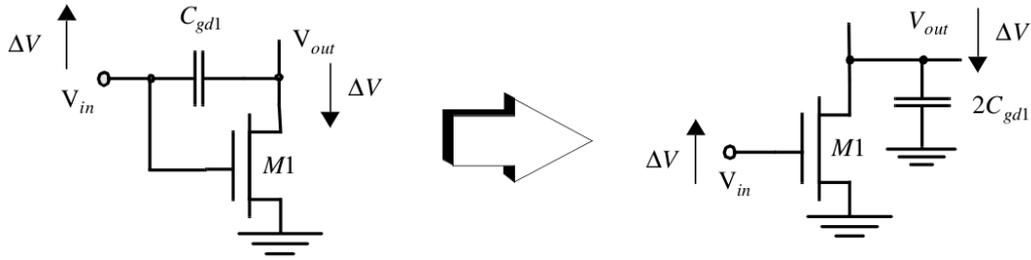


Figura C.1: Efecto Miller.

### C.1.2. Capacidad de difusión $C_{db1}$ y $C_{db2}$

La capacidad parásita entre drain y bulk se debe a la polarización inversa de la juntura pn parásita. Una capacidad de este tipo es, por desgracia, bastante alineal y depende en gran medida de la tensión aplicada. Para linealizar este efecto, es posible reemplazar el capacitor no lineal por un capacitor lineal con la misma sensibilidad de carga según la tensión aplicada en el rango de interés. Se introduce un factor de multiplicación  $K_{eq}$  para relacionar el capacitor linealizado con el valor de la capacidad de juntura bajo condiciones de polarización cero.

$$C_{eq} = K_{eq} \times C_{j0} \quad (C.1)$$

siendo  $C_{j0}$  la capacidad de juntura por unidad de área en condiciones de polarización cero. Utilizando este enfoque, la capacidad de juntura puede ser reemplazada por un componente lineal. El resultado de la linealización presenta una distorsión menor de las formas de onda de tensión. Los retardos lógicos no se ven influenciados significativamente por esta simplificación.

## C.2. Capacidad de ruteo $C_w$

La capacidad debida al ruteo depende de la longitud y el ancho de las interconexiones, y es función de la distancia del fan-out de salida desde el gate; y el número de fan-out.

### C.3. Capacidad de gate $C_{g3}$ y $C_{g4}$

Suponemos que la capacidad de salida es igual a la capacidad total del gate de las cargas M3 y M4. Por lo tanto,

$$\begin{aligned} C_{fanout} &= C_{gate}(nMOS) + C_{gate}(pMOS) \\ &= (C_{GSO_n} + C_{GDO_n} + W_n L_n C_{ox}) + (C_{GSO_p} + C_{GDO_p} + W_p L_p C_{ox}) \end{aligned} \quad (C.2)$$

Esta expresión simplifica la situación real de dos maneras:

- Se supone que todos los componentes de la capacidad de gate están conectados entre  $V_{OUT}$  y GND (o VDD), e ignora el efecto Miller en las capacidades de gate-drain. Esto tiene un efecto relativamente menor sobre la exactitud, ya que podemos asumir con seguridad que los gates no se enciende antes de alcanzar el punto de 50%, y  $V_{OUT2}$ , por lo tanto, se mantiene constante en el intervalo de interés.
- Una segunda aproximación es que la capacidad del canal es constante durante el intervalo de interés. Sin embargo, la capacidad total del canal es una función del modo de funcionamiento del dispositivo, y varía aproximadamente desde  $1/3WLC_{ox}$  (corte) hasta  $2/3WLC_{ox}$  (saturación). Durante la primera mitad del transitorio, se puede suponer que uno de los dispositivos de carga está siempre en modo lineal, mientras que el otro transistor evoluciona desde corte a saturación. Haciendo caso omiso de los resultados de variación de capacidad, en una estimación pesimista con un error de aproximadamente 10%, lo que es aceptable para un primer análisis de orden.



# Apéndice D

## MOSIS Test Data

Se presenta a continuación los resultados del test data de MOSIS para la corrida v53v en la cual se fabricaron los chips.

## MOSIS WAFER ELECTRICAL TESTS

RUN: V53V  
 TECHNOLOGY: SCN05  
 microns

VENDOR: ON-SEMI  
 FEATURE SIZE: 0.5

Run type: SHR

INTRODUCTION: This report contains the lot average results obtained by MOSIS from measurements of MOSIS test structures on each wafer of this fabrication lot. SPICE parameters obtained from similar measurements on a selected wafer are also attached.

COMMENTS: SMSCN3ME06\_ON-

TRANSISTOR PARAMETERS	W/L	N-CHANNEL	P-CHANNEL	UNITS
MINIMUM	3.0/0.6			
Vth		0.80	-0.93	volts
SHORT	20.0/0.6			
Idss		475	-255	uA/um
Vth		0.68	-0.91	volts
Vpt		12.5	-12.2	volts
WIDE	20.0/0.6			
Ids0		< 2.5	< 2.5	pA/um
LARGE	50/50			
Vth		0.72	-0.96	volts
Vjblkd		11.1	-11.8	volts
Ijlk		147.0	<50.0	pA
Gamma		0.47	0.58	V^0.5
K' (Uo*Cox/2)		58.4	-19.2	uA/V^2

COMMENTS: Poly bias varies with design technology. To account for mask bias use the appropriate value for the parameter XL in your SPICE model card.

Design Technology	XL (um)	XW (um)
SCMOS_SUBM (lambda=0.30)	0.10	0.00
SCMOS (lambda=0.35)	0.00	0.20

FOX TRANSISTORS	GATE	N+ACTIVE	P+ACTIVE	UNITS
Vth	Poly	>15.0	<-15.0	volts

COMMENTS:

PROCESS PARAMETERS	N+	P+	N_W	_U	POLY	PLY2_HR	POLY2	M1	UNITS
Sheet Resistance	84.2	110.6	807.8	23.4	1089		40.9	0.09	ohms/sq
Contact Resistance	60.3	144.9		17.2			27.1		ohms
Gate Oxide Thickness	138 angstrom								

PROCESS PARAMETERS	M2	M3	N_W	UNITS
Sheet Resistance	0.09	0.05	803	ohms/sq
Contact Resistance	0.85	0.83		ohms

## COMMENTS:

CAPACITANCE PARAMETERS	N+	P+	POLY	POLY2	M1	M2	M3	N_W	UNITS
Area (substrate)	415	726	89		27	12	8	86	
aF/um <sup>2</sup>									
Area (N+active)			2504		35	16	11		
aF/um <sup>2</sup>									
Area (P+active)			2423						
aF/um <sup>2</sup>									
Area (poly)				885	59	16	9		
aF/um <sup>2</sup>									
Area (poly2)					52				
aF/um <sup>2</sup>									
Area (metall)						31	12		
aF/um <sup>2</sup>									
Area (metal2)							32		
aF/um <sup>2</sup>									
Fringe (substrate)	334	213			50	34	26		aF/um
Fringe (poly)					64	38	27		aF/um
Fringe (metall)						52	32		aF/um
Fringe (metal2)							47		aF/um
Overlap (N+active)			186						aF/um
Overlap (P+active)			234						aF/um

## COMMENTS:

CIRCUIT PARAMETERS			UNITS
Inverters	K		
Vinv	1.0	2.01	volts
Vinv	1.5	2.27	volts
Vol (100 uA)	2.0	0.47	volts
Voh (100 uA)	2.0	4.48	volts
Vinv	2.0	2.45	volts
Gain	2.0	-18.91	
Ring Oscillator Freq.			
DIV256 (31-stg,5.0V)		98.91	MHz
D256_WIDE (31-stg,5.0V)		155.97	MHz
Ring Oscillator Power			
DIV256 (31-stg,5.0V)		0.47	uW/MHz/gate
D256_WIDE (31-stg,5.0V)		0.99	uW/MHz/gate

## COMMENTS: SUBMICRON

SPICE BSIM parameters not available.



# Apéndice E

## Trabajos publicados

G. A. Sanca, O. H. Alpago, M. García Inza, “Desarrollo de un Kit de Diseño Interoperable y un conjunto de celdas estándar abiertos para un proceso CMOS escalable” VI Congreso de Microelectrónica Aplicada, 2015



# Glosario

## Acrónimos

<b>ASIC</b>	Application Specific Integrated Circuit
<b>BSIM</b>	Berkeley Short Channel IGFET Model
<b>CAD</b>	Computer Aided Design
<b>CMOS</b>	Complementary Metal Oxide Semiconductor
<b>DRC</b>	Design Rule Check
<b>EDA</b>	Electronic Design Automation
<b>FIUBA</b>	Facultad de Ingeniería de la Universidad de Buenos Aires
<b>FPGA</b>	Field Programmable Gate Array
<b>GDS</b>	Graphic Database System
<b>HDL</b>	Hardware Description Language
<b>IC</b>	Integrated Circuit
<b>IGFET</b>	Insulated Gate Field Effect Transistor
<b>IO</b>	Input-Output
<b>iPDK</b>	Interoperable Process Design Kit
<b>LM</b>	Laboratorio de Microelectrónica de la FIUBA
<b>LPP</b>	Layer Purpose Pairs
<b>*.lib</b>	Liberty
<b>LVS</b>	Layout Versus Schematic
<b>MEP</b>	MOSIS Educational Program
<b>MOS</b>	Metal Oxide Semiconductor
<b>MOSFET</b>	Metal Oxide Semiconductor Field Effect Transistor
<b>PDK</b>	Process Design Kit
<b>P&amp;R</b>	Place and Route

<b>SPICE</b>	Simulation Program with Integrated Circuit Emphasis
<b>VHDL</b>	Very High Speed Integrated Circuit Hardware Description Language
<b>VLSI</b>	Very large scale integration
<b>OA</b>	OpenAccess
<b>Tcl</b>	Tool Command Language. Se pronuncia tí.quel

## Layers

<b>M1</b>	Metal 1
<b>M2</b>	Metal 2
<b>M3</b>	Metal 3
<b>NWELL</b>	Difusión tipo N que define el bulk de los transistores pMOS
<b>POLY</b>	Silicio muy dopado (polisilicio)
<b>NACTIVE</b>	Zona activa tipo N. Difusión de drain y source
<b>PACTIVE</b>	Zona activa tipo P. Difusión de drain y source
<b>N_SELECT</b>	Layer lógico que identifica a la zona activa como tipo N
<b>P_SELECT</b>	Layer lógico que identifica a la zona activa como tipo P
<b>HR</b>	Implante de alta resistividad
<b>ELEC</b>	" <i>Electrode</i> ". Segunda capa de polisilicio. A.k.a POLY2
<b>CC</b>	Contacto
<b>CA</b>	Contacto entre NACTIVE/PACTIVE y M1
<b>CP</b>	Contacto entre POLY y M1
<b>CE</b>	Contacto entre ELEC y M1
<b>VIA1</b>	Contacto entre M1 y M2
<b>VIA2</b>	Contacto entre M2 y M3

## Parámetros

$A_v$	Ganancia
<b>CLK</b>	Reloj
$C_{IN}$	Capacidad de entrada
$C_{OUT}$	Capacidad de salida
$C_{ox}$	Capacidad del óxido de gate
$C_{gd12}$	Capacidad gate-drain
$C_{db1}$	Capacidad de difusión
$C_w$	Capacidad de ruteo (interconexiones)
$C_g$	Capacidad de gate
$g_x$	Horizontal grid
$g_y$	Vertical grid

<b>h</b>	Fan-out
<b>p</b>	Parasitic delay
<b>RST, R</b>	Reset
$t_d, t_{pd}, t_p$	Propagation delay
$V_T$	Threshold voltage



# Referencias

- [1] N. Williams, J. Miller, “PDKs for Analog/Mixed-Signal (AMS) Design and Verification”, Marketing de Productos, Mentor Graphics
- [2] “IPL Now”, <https://www.iplnow.com>.
- [3] G. E. Moore, “Cramming More Components onto Integrated Circuits”, Electronics, 1965
- [4] “Excerpts from A Conversation with Gordon Moore: Moore’s Law”, 2005
- [5] G. E. Moore, “Progress in Digital Integrated Electronics”, IEEE. Reprinted, with permission, from Technical Digest 1975. International Electron Devices Meeting, IEEE, 1975, pp. 11-13.
- [6] N. A. Sherwani, “Algorithms for VLSI physical design automation”, 3ra Ed., Kluwer Academic Publishers, 2002.
- [7] Dr. Wanlass, Frank Marion, “Low stand-by power complementary field effect circuitry”, U.S Patent #3356858
- [8] Fairchild Semiconductor, “CMOS, the Ideal Logic Family”, nota de aplicación número 77, Enero 1983
- [9] Pedro Julián, “Introducción a los dispositivos semiconductores: principios y modelos”, EdiUNS, 2011.
- [10] N. H. E. Weste, D. M. Harris, “CMOS VLSI Design. A Circuits and Systems Perspective”, 4ta Ed., Addison-Wesley, 2011.
- [11] A. Bukowski “Czochralski-Grown Silicon Crystals for Microelectronics”, Acta Physica Polonica, 2013
- [12] N. Van den Bogaert, F. Dupret, “Dynamic global simulation of the Czochralski process I. Principles of the method” Journal of Crystal Growth 171 (1997) 65-76

- [13] D. J. Walkey, "ELEC 3908, Physical Electronics: Basic IC Processing"
- [14] E. Chen, Applied Physics 298r, 2004
- [15] J.M Rabaey, A. Chandrakasan, B. Nikolic, "Digital Integrated Circuits", Editorial Prentice Hall, 2003.
- [16] H. Wu, W. Luo, H. Zhou, M. Si, J. Zhang and P. D. Ye "First Experimental Demonstration of Ge 3D FinFET CMOS Circuits", Symposium on VLSI Technology Digest of Technical Papers, 2015
- [17] A. Ricci, "Dry Etch Process Application Note", Pall Microelectronics, 2005
- [18] "PowerPC", Wikipedia.
- [19] S. Tahmasbi Oskuii, A. Alvandpour, "Comparative study on low-power high-performance standard-cell flip-flops" Electronic Devices, Linkoping University, Linkoping, Sweden
- [20] Vendor-independent, scalable rules (MOSIS SCMOS Rules), <https://www.mosis.com/files/scmos/scmos.pdf>.
- [21] Carver A. Mead, Lynn A. Conway, "Introduction to VLSI Systems", Ed. 2, Addison-Wesley, 1980.
- [22] "Design for manufacturability", Wikipedia.
- [23] "Galaxy Custom Designer ® iPDK Developer Guide" Synopsys, Inc, 2013
- [24] "Interoperable Process Design Kit Developer's Guide" Versión 1.0, IPL Alliance
- [25] L. Li, "45nm Extraction and verification flow with SPACE" Tesis de magister, Delft University of Technology, 2009
- [26] Naveed A. Sherwani, "Algorithms for VLSI Physical Design Automation", 3ra Ed., Kluwer Academic Publishers, 2002.
- [27] C. Saint, J. Saint, "IC Mask Design: Essential Layout Techniques", 2da Ed., McGraw-Hill, 2004.
- [28] "Documentación OpenAccess", <http://www.si2.org/?page=621>.

---

SANCA, GABRIEL ANDRÉS