CENTRO DE INVESTIGACIÓN CIENTÍFICA Y DE EDUCACIÓN SUPERIOR DE ENSENADA, BAJA CALIFORNIA

PROGRAMA DE POSGRADO EN CIENCIAS EN CIENCIAS DE LA COMPUTACIÓN

Diseño de compuertas lógicas mediante desplazamiento de cadenas de ADN

Tesis

para cubrir parcialmente los requisitos necesarios para obtener el grado de

Maestro en Ciencias

Presenta:

Ana Karen Velázquez Sánchez

Ensenada, Baja California, México

2014

Tesis defendida por

Ana Karen Velázquez Sánchez

y aprobada por el siguiente comité

Dr. Israel Marck Martínez Pérez
Director del Comité

Dr. Carlos Alberto Brizuela Rodríguez.
Miembro del Comité

Dr. Hugo Homero Hidalgo Silva
Miembro del Comité

Dr. Jorge Olmos Soto
Miembro del Comité

Dra. Ana Isabel Martínez García
Coordinador del Programa de Posgrado en Ciencias de la Computación

Dr. Jesús Favela Vara
Director de Estudios de Posgrado

Octubre, 2014

Resumen de la tesis que presenta Ana Karen Velázquez Sánchez como requisito parcial para la obtención del grado de Maestro en Ciencias en Ciencias de la Computación.

Diseño de compuertas lógicas mediante desplazamiento de cadenas de ADN

Resumen elaborado por:

Ana Karen Velázquez Sánchez

El Cómputo molecular es un área emergente y se refiere principalmente al uso de biomoléculas con propósitos computacionales. Actualmente, se considera al ADN como un material ideal para la ejecución de operaciones de cómputo, ya que admite la construcción de ciertos dispositivos como compuertas y circuitos lógicos. Uno de los paradigmas más importantes para el diseño de dispositivos de ADN son las reacciones de desplazamiento de cadenas de ADN, principalmente porque dicha técnica permite la implementación de dispositivos computacionales sin la necesidad de componentes adicionales (e.g. enzimas), lo que admite programar el cálculo únicamente en términos de secuencias de nucleótidos. En esta tesis se presenta una biblioteca de compuertas lógicas (AND, OR, NAND, NOT, XOR, NOR, XNOR) diseñadas a partir de este tipo de reacciones. Las estructuras se diseñaron utilizando el lenguaje DSD y se simularon estocásticamente mediante la herramienta Visual DSD. Para analizar algunas propiedades cuantitativas y termodinámicas de los modelos se utilizaron la herramientas PRISM y NUPACK, respectivamente. Las compuertas lógicas propuestas pueden ser aplicables a distintos ámbitos, como por ejemplo, el diagnóstico clínico de alguna patología o el diseño de circuitos combinatorios más complejos como un sumador completo.

Palabras Clave: **Cómputo Molecular, Biocomputación, desplazamiento de cadenas de ADN, compuertas lógicas.**

Abstract of the thesis presented by Ana Karen Velázquez Sánchez as a partial requirement to obtain the Master of Science degree in Master in Sciences in Computer Science.

Logic gates design based on DNA strand displacement

Abstract by:

Ana Karen Velázquez Sánchez

Molecular computing is an emerging area which mainly refers to the use of biomolecules with computational purposes. Currently, DNA is considered as an ideal material for the execution of computational operations, since it enables the construction of certain computational devices such as logic gates and circuits. The design of DNA devices is simplified by the predictability of Watson-Crick base pairing and based on this physical property, DNA strand displacement has arisen as one of the most important paradigms for performing computation with DNA molecules. Strand displacement reactions rely on the hybridization of complementary nucleotides to displace prehybridized partial double-stranded DNA. In this thesis we present a suite of basic logic gates (AND, OR, NAND, NOT, XOR, NOR, XNOR) based on DNA strand displacement reactions. All of the structures were designed using DSD language, stochastically simulated using the Gillespie algorithm incorporated in the Visual DSD software and analyzed using PRISM Model Checker and NUPACK. Potential applications for the proposed devices include the clinical diagnosis of diseases and the implementation of complex combinatorial circuits such as full adders.

Keywords: **DNA Computing, Biocomputing, DNA strand displacement, logic gates.**

Dedicatoria

A mi madre...

Agradecimientos

A mi madre, mi orgullo y ejemplo a seguir, gracias por todo el apoyo y cariño.

A mi director de tesis, el Dr. Israel Marck Martínez Pérez, por guiarme pacientemente durante el desarrollo de este trabajo.

A mi comité de tesis, por sus observaciones y cometarios para mejorar mi investigación.

A todos los investigadores, estudiantes y personal del Departamento de Ciencias de la Computación por apoyo durante mi estancia en esta institución y por su enseñanza académica.

Al Centro de Investigación Científica y de Educación Superior de Ensenada.

Al Consejo Nacional de Ciencia y Tecnología (CONACyT) por brindarme el apoyo económico para realizar mis estudios de maestría.

Tabla de contenido

Página

Tabla de contenido (continuación)

Tabla de contenido (continuación)

Lista de figuras

Figura Página

Lista de figuras (continuación)

Lista de figuras (continuación)

Lista de figuras (continuación)

Figura Página

Lista de tablas

Capítulo 1. Introducción

1.1. Antecedentes y motivación

El cómputo o computación molecular es un área emergente relativamente promisoria para el futuro, la cual está creciendo velozmente. Principalmente, el término se refiere al uso de biomoléculas con propósitos computacionales, y por otro lado, a comprender la naturaleza computacional de los procesos moleculares que se llevan a cabo en las células vivas. El ADN es la molécula donde se almacena el material genético de los seres vivos y éste se utiliza como un medio para la computación molecular (Adleman, 1998). Adicionalmente, se sabe que los sustratos moleculares pueden ser vistos como dispositivos computacionales que procesan entradas tanto físicas como químicas para generar productos basándose en un conjunto de operadores lógicos (Lakin *et al.*, 2012c). El ADN es un excelente material para construir circuitos bioquímicos porque su naturaleza biológica soporta aplicaciones tecnológicas *in vivo*, su síntesis química es sencilla y facilita experimentos prácticos *in vitro*, además de que su estructura combinatoria proporciona suficiente espacio para el diseño de las secuencias y el principio de complementariedad de Watson-Crick permite un comportamiento molecular predecible (Win y Smolke, 2008). Los dispositivos moleculares construidos usando ADN son competentes para aplicaciones en una amplia gama de áreas importantes, incluyendo bio-detección, manufactura biomimética molecular y administración de fármacos. Sin embargo, el diseño de dispositivos de ADN correctos y robustos implica un gran desafío, principalmente porque existe la posibilidad de que se incluya interferencia no deseada entre las moléculas del sistema (Qian y Winfree, 2011). Uno de los retos actuales consiste en reconocer e identificar la existencia de operaciones lógicas simples dentro de las reacciones químicas y fenómenos biológicos. El diseño de sistemas lógicos moleculares, aparte de ser reconfigurable, es aplicable en áreas de considerable importancia como ciencias de la vida o medicina, ya que su tamaño tan pequeño admite la detección y/o reconocimiento de micropartículas especialmente en el medio intracelular, lo que eventualmente permitiría la realización de diagnósticos inteligentes; con esto y las características que se mencionaron anteriormente, se reafirma que el ADN es un material ideal para la realización de operaciones de cómputo y admite la construcción de algunos dispositivos tales como compuertas lógicas

y circuitos.

Se han utilizado diversos enfoques para implementar circuitos de ADN, algunos dependen de agentes externos tales como enzimas de restricción o maquinaria de transcripción adicional para funcionar. En comparación con los otros enfoques, el desplazamiento de cadenas de ADN es una alternativa que tiene la ventaja de requerir solamente complejos de cadenas sencillas de ADN (que son relativamente baratos y fáciles de conseguir) sin la necesidad de que intervengan otro tipo de moléculas, ya que esta técnica se basa únicamente en la hibridación entre las secuencias de nucleótidos complementarias para llevar a cabo los pasos de cálculo (Lakin *et al.*, 2012c). Los circuitos construidos por medio de este enfoque son eficientes en cuanto a tiempo y energía (Chiniforooshan *et al.*, 2011) además de que presentan una amplia gama de aplicaciones tales como terapéutica médica *in vivo*, instrumentos moleculares *in situ* y diagnóstico biomédico *in vitro* (Qian y Winfree, 2011).

1.2. Definición del problema

Se busca diseñar configuraciones de complejos de ADN a partir de reacciones de desplazamiento de cadenas de tal forma que dichos complejos se comporten como compuertas lógicas. En trabajos del estado del arte, salvo aquel de (Kahan-Hanum *et al.*, 2013) que no utiliza desplazamiento de cadenas, comúnmente presentan una o dos compuertas (por lo general AND y OR) e implementan un circuito o una aplicación más compleja donde las utilizan y sugieren que es posible construir o emular las demás compuertas lógicas existentes a partir de las que presentan o implementaron; sin embargo, no especifican el procedimiento exacto para hacerlo. Además, por lo general, en la mayoría de los trabajos se comenta que es posible escalar los diseños para programar circuitos más complejos o generar otro tipo de compuertas partir de las que presentan en su enfoque; aunque, no ejemplifican ni proveen una idea clara de cómo hacerlo y probablemente se necesitan hacer modificaciones a los diseños para que esto sea posible.

Adicionalmente, no se ha reportado ningún trabajo que presente una biblioteca de compuertas lógicas a partir de reacciones de desplazamiento de cadenas de ADN. Debido a lo anterior, se propone el diseño de una biblioteca de compuertas lógicas básicas

así como una aplicación conceptual donde se compruebe su funcionamiento mediante simulación computacional.

1.3. Objetivos

1.3.1. Objetivo general

El objetivo de este trabajo de investigación es diseñar las compuertas lógicas básicas (AND, OR, NOT, NAND, XOR, NOR, XNOR) mediante reacciones de desplazamiento de cadenas de ADN.

1.3.2. Objetivos específicos

1. Analizar las diferentes tecnologías de desplazamiento de cadenas para determinar la más indicada para realizar el diseño de las compuertas lógicas.
2. Diseñar compuertas lógicas con moléculas de ADN utilizando las tecnologías de desplazamiento de cadenas seleccionadas.
3. Implementar y simular estocásticamente los modelos de las compuertas lógicas.
4. Evaluar la fiabilidad de los resultados de la simulación de las compuertas lógicas diseñadas, comparando los umbrales de 0s y 1s lógicos, de tal manera que exista una separación bien definida entre éstos.
5. Investigar una aplicación importante donde se pueda probar el funcionamiento de las compuertas lógicas, preferentemente en el ámbito del diagnóstico clínico.
6. Probar que el sistema generado es robusto mediante el diseño de una aplicación conceptual donde se utilicen las compuertas en cascada, preferentemente en el ámbito del diagnóstico médico.

1.4. Metodología

Con el fin de llevar a cabo los objetivos, primeramente se planteó el problema de investigación como un problema de diseño. Después del proceso de revisión de la literatura se

determinaron cuáles compuertas lógicas serían diseñadas: AND, OR, NOT, NAND, XOR, NOR, XNOR. Para ello, se seleccionaron los métodos y herramientas computacionales y de cómputo molecular ideales para realizar este trabajo, los cuales corresponden al modelo de desplazamiento de cadenas de ADN y a los softwares PRISM (Kwiatkowska *et al.*, 2011) y Visual DSD (Phillips y Cardelli, 2009), respectivamente.

El primer paso en el diseño del modelo propuesto fue determinar la forma de representación de las variables de entrada y salida del sistema para que éstas pudieran generar reacciones en cascada y fueran las mismas para todas las compuertas. Una vez hecho esto, fue necesario generar la estructura para reconocer las variables de salida (complejo detector) para después realizar los diseños conceptuales de los complejos de las siete compuertas lógicas.

Posteriormente, fue necesario probar la fiabilidad de los modelos diseñados; debido a que no fue posible realizar los experimentos en laboratorio, éstos se realizaron *in silico*, es decir, los modelos de las compuertas diseñadas se simularon estocásticamente mediante un reconocido software especializado para modelos de desplazamiento de cadenas (Microsoft Visual DSD). El simulador estocástico de Visual DSD utiliza el algoritmo de Gillespie (Gillespie, 1977). Además de las simulaciones estocásticas, se analizaron las redes de reacciones y el espacio de estados con esta misma herramienta.

Para verificar algunas propiedades cuantitativas de los modelos se utilizó la herramienta PRISM, la cual junto con Visual DSD permite corroborar que el funcionamiento de los modelos propuestos fuera correcto y de acuerdo a como se esperaba. Esta herramienta computacional utiliza técnicas de verificación formal para analizar propiedades cuantitativas de modelos que exhiben un comportamiento estocástico o probabilístico.

Con el fin de verificar la estabilidad termodinámica de las compuertas se realizó un análisis de las estructuras con el software NUPACK (Zadeh *et al.*, 2011). El análisis de las propiedades termodinámicas consistió básicamente en obtener la energía libre de cada uno de los complejos.

Para ejemplificar el potencial y comprobar la robustez de los modelos de las compuertas lógicas diseñados también se investigó y adaptó una aplicación en medicina molecular

donde los modelos de las compuertas pueden ser utilizados. Por último, se propuso el diseño de un circuito sumador completo con las compuertas propuestas.

1.5. Organización de la tesis

Este trabajo se organiza en ocho capítulos: el Capítulo 2 corresponde a cuestiones de teoría en biología molecular, siendo éstas las bases para el cómputo molecular. El Capítulo 3 está dedicado a los fundamentos de computación, donde se presentan algunas definiciones básicas, las cuales serán útiles en los siguientes capítulos; algunas de estas definiciones incluyen tanto temas de compuertas y circuitos lógicos como de teoría de simulación estocástica. En el Capítulo 4 se presenta la teoría de cómputo molecular, que incluye definiciones importantes, antecedentes, modelos más representativos y algunas aplicaciones potenciales del área en cuestión.

Los siguientes tres capítulos abarcan el trabajo desarrollado en la presente tesis. En el Capítulo 5 se presentan los modelos conceptuales de cada una de las compuertas lógicas así como de las moléculas auxiliares y la representación de variables. El Capítulo 6 está dedicado a la simulación, pruebas y desempeño de las compuertas. El Capítulo 7 contiene dos aplicaciones conceptuales para probar la funcionalidad de la biblioteca de compuertas lógicas formadas con ADN y la robustez del sistema: una en el ámbito médico y otra en circuitos digitales. Finalmente, en el Capítulo 8 se concluye el trabajo con una breve discusión acerca de los resultados obtenidos y la descripción del trabajo a futuro.

Capítulo 2. Fundamentos de biología molecular

Las moléculas de la célula se dividen en dos clases: macromoléculas y micromoléculas; para este trabajo en particular, las de mayor interés son las primeras. Existen tres tipos de macromoléculas: ADN, ARN y proteínas (Waterman, 1995). En este capítulo se presentan las bases necesarias en biología molecular para el diseño de compuertas lógicas a partir del modelo de desplazamiento de cadenas de ADN.

2.1. Nucleótidos

Los nucleótidos son moléculas formadas por una base nitrogenada unida a un ázucar (ribosa o desoxirribosa) al que se une al menos un grupo fosfato (Figura 1). Las bases nitrogenadas son moléculas planas, aromáticas y heterocíclicas que derivan estructuralmente de la purina o pirimidina. Como se puede observar en la Figura 2(a) las purinas son adenina (A) y guanina (G), y las pirimidinas son la citosina (C), el uracilo (U) y la timina (T) Figura 2(b); las pirimidinas tienen un anillo mientras que las purinas tienen dos, los enlaces están representados por las líneas rectas. Los nucleótidos se unen entre sí para formar ácidos nucleicos. Los números del 1′ al 3′ se refieren a las localizaciones de los átomos de carbono en el ázucar. Los átomos de carbono 5′ y 3′ se utilizan para definir la orientación de la molécula.

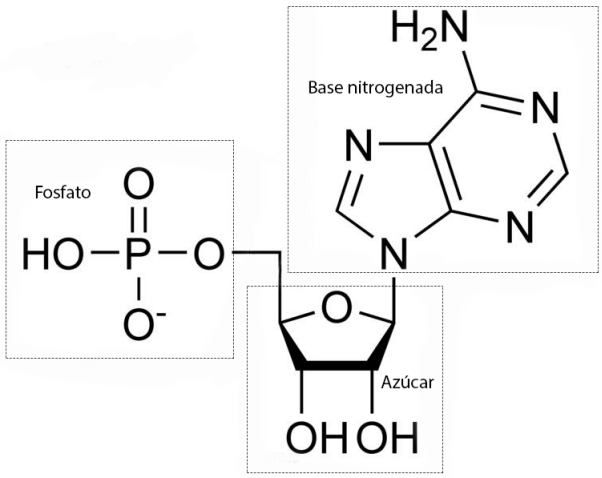


Figura 1: Estructura de un nucleótido.

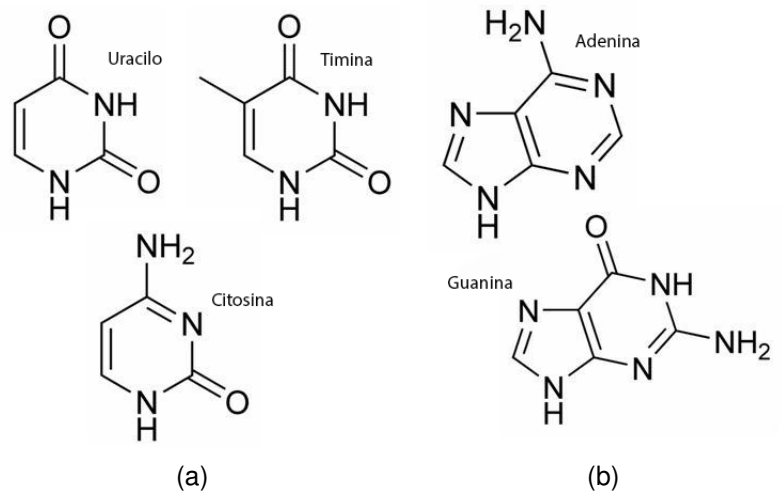


Figura 2: Bases nitrogenadas: (a) purinas; (b) pirimidinas.

2.2. Ácidos nucleicos

Los ácidos nucleicos son polinucleótidos, es decir, son cadenas de nucleótidos cuyos fosfatos se unen en las posiciones 3′ y 5′ de unidades de ribosa vecinas. A cada nucleótido que conforma el polinucleótido se le conoce como residuo nucleotídico. El residuo terminal cuyo C5′ no está unido a ningún otro nucleótido se le conoce como extremo 5′ y de manera análoga, al residuo terminal cuyo C3′ no está unido a otro nucleótido se le denomina extremo 3′. Por convención, la secuencia de nucleótidos en un ácido nucléico se escribe de izquierda a derecha, es decir, del extremo 5′ al 3′. Existen dos tipos básicos de ácidos nucléicos, el desoxirribonucleico (ADN) y el ribonucleico (ARN).

2.2.1. ADN

El ADN es un polímero formado a partir de nucleótidos, el cual contiene la información genética en todas las células y en algunos virus, ya que existen otros donde el material genético es ARN. El ADN es un ácido nucleico y es el medio por el cual los organismos trasfieren información genética a sus descendientes. En organismos eucariotas, el ADN se encuentra en el núcleo y en las procariotas se encuentra disperso en el citoplasma. De manera más simple, la molécula de ADN puede ser vista como una palabra formada por un alfabeto de 4 letras (A, G, C, T) correspondientes a los nucleótidos y su longitud se mide de acuerdo al número de pares de bases que contiene. Una de las características más importantes del ADN es la complementariedad de bases, esto es, la adenina es

complementaria a la timina y la guanina con la citosina y se consigue mediante la unión de puentes de hidrógeno. Tal como se puede obsevar en la Figura 2(a) y 2(b), la adenina se une a la timina por 2 puentes de hidrógeno (líneas punteadas) mientras que la citosina y la guanina se unen con 3. Los puentes de hidrógeno son enlaces débiles entre dos átomos de carga negativa (por lo general oxígeno o nitrógeno) y un átomo de hidrógeno (carga positiva) unido covalentemente a uno de los dos átomos electronegativos. Como se muestra en la Figura 4, una cadena sencilla de ADN está formada por secuencias sucesivas de un fosfato seguido de un azúcar, con el carbono 5′ del azúcar unido al fosfato y con el carbono 1′ unido a la base y recordando que por definición, la dirección de la cadena es $5' - 3'$. Una cadena sencilla de ADN ($5' - 3'$) se une a una cadena complementaria la cual tiene la dirección contraria ($3' - 5'$) mediante puentes de hidrógeno para formar una molécula completa de ADN, la cual presenta la estructura de doble hélice tal como se aprecia en la Figura 5.

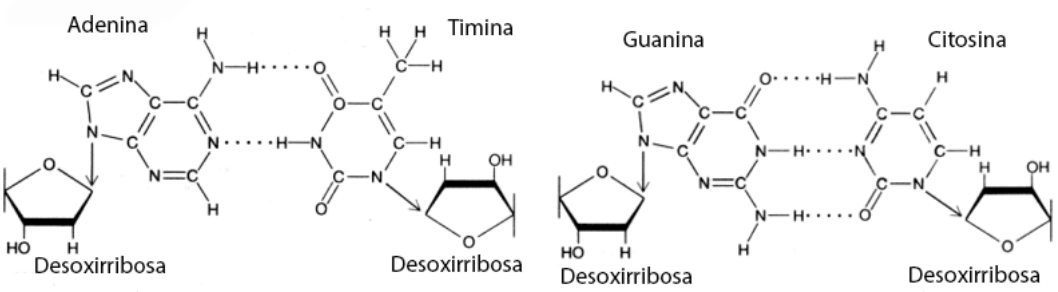


Figura 3: Complementariedad de bases: (a) Complementariedad de adenina con timina; (b) complementariedad de guanina con citosina.

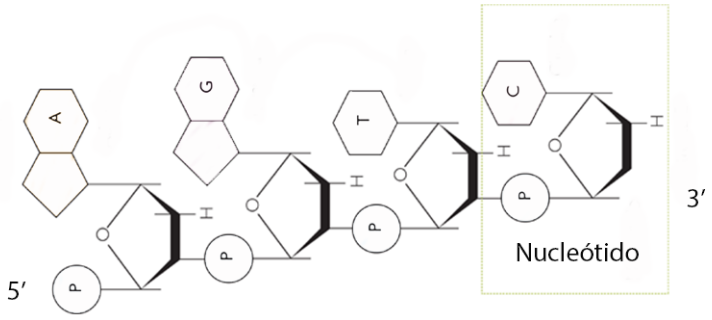


Figura 4: Molécula de ADN.

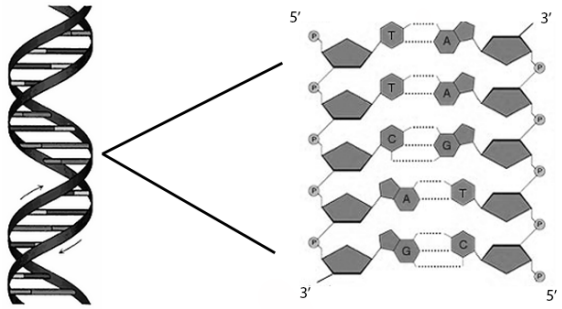


Figura 5: Estructura de la doble hélice del ADN.

2.2.2. ARN

Al igual que el ADN, la molécula de ARN puede ser vista como una palabra formada por un alfabeto de 4 letras (A, G, C, U) correspondientes a los ribonucleótidos. La complementariedad de bases es similar a la del ADN con excepción de que la timina es reemplazada por uracilo, el cual es complementario a la adenina. En el esqueleto del ARN el azúcar 2-desoxirribosa se reemplaza por ribosa (Figura 6).

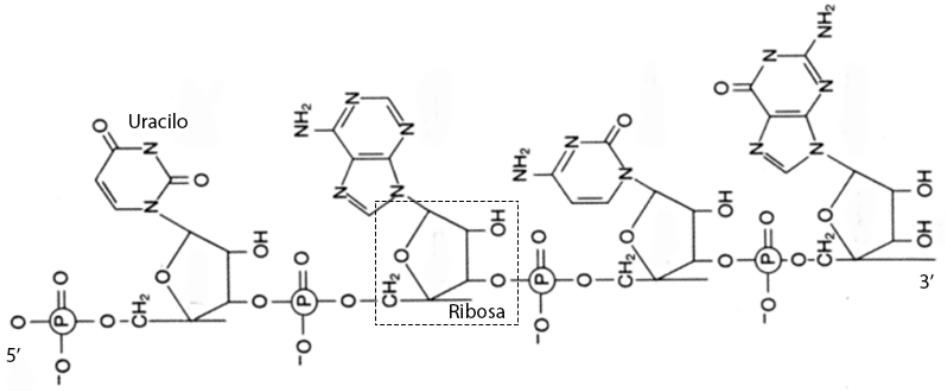


Figura 6: Estructura del ARN.

2.3. Proteínas

Las proteínas también son polímeros. Estas moléculas están formadas por aminoácidos unidos de forma lineal mediante enlaces peptídicos, los cuales son enlaces covalentes. Existen 20 aminoácidos esenciales, a partir de combinaciones entre ellos se obtienen las diferentes proteínas. Cada proteína tiene diferentes funciones dentro de la célula.

2.3.1. Enzimas

Las enzimas son proteínas complejas encargadas de catalizar todas las reacciones bioquímicas. Un catalizador es una sustancia que disminuye la energía de activación de una reacción química y cuando esto sucede, se incrementa la velocidad de la reacción. El sustrato es la sustancia sobre la cual actúa una enzima y ésta es específica para cada enzima.

El funcionamiento de las enzimas puede describirse de la siguiente forma: la enzima (E) y el sustrato (S) se combinan para formar un complejo intermedio enzima sustrato $(E - S)$, el cual se descompone formando un producto y regenerando la enzima. Debido a su importancia como catalizadores biológicos, tienen muchos usos médicos y comerciales.

2.4. Dogma central de la biología molecular

El dogma central de la biología molecular fue articulado por primera vez en (Crick, 1958), donde se describe el proceso mediante el cual la información genética pasa a formar proteínas; dicho proceso se compone de dos pasos: transcripción y traducción. En la Figura 7 se aprecia el esquema que detalla de manera general el dogma central, donde se puede observar que una vez que la información es codificada a proteínas es imposible revertir el proceso; sin embargo, la transferencia de información de ácidos nucleicos a ácidos nucleicos o de ácidos nucleicos a proteínas es posible. Con el término información nos referimos a una secuencia determinada de nucleótidos en el ácido nucleico o de residuos de aminoácidos en el caso de las proteínas. El ciclo de ADN a ADN representa la replicación, esto es, que una molécula de ADN puede copiarse, por lo que es posible crear una molécula de ADN a partir de otra molécula existente. La transcripción está representada por la segunda línea que va de ADN a ARN, aquí la información genética del ADN pasa a ARN mensajero (ARNm), este es el primer paso de la síntesis de proteínas. La segunda línea de color negro que va de ARN a proteína representa el proceso de traducción, este es el segundo paso de la síntesis proteica donde se convierte una secuencia de ARNm en una cadena de aminoácidos para formar una proteína. Los retrovirus pueden sintetizar ADN a partir de ARN, esto lo hacen mediante la transcripta-

sa inversa, la cual es una polimerasa. Lo anterior supone una evasión al dogma, ya que originalmente sólo permite el flujo de la información en un sentido.

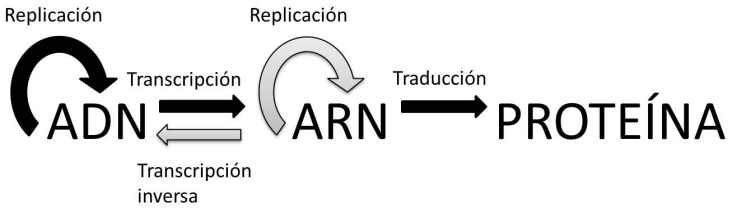


Figura 7: Dogma central de la biología molecular.

2.5. Termodinámica

2.5.0.1. Leyes de termodinámica

Las leyes de la termodinámica presentan una gran importancia ya que son principios generales aplicables a todos los procesos biológicos y físicos; éstos principios determinan las condiciones bajo las cuales algún proceso específico puede o no ocurrir. Primeramente, en las leyes de termodinámica existen dos conceptos básicos: el sistema y su entorno; el sistema consiste en la materia incluida en una región definida del espacio y el entorno corresponde a la materia restante.

La Primera Ley de la Termodinámica establece que la energía total de un sistema y de su entorno permanece constante, es decir, el contenido energético del universo es constante, ya que la energía no puede ser creada ni destruída; no obstante, ésta puede tomar diferentes formas, por ejemplo, el calor es una manifestación de energía cinética la cual se asocia con movimiento aleatorio de moléculas. Otra forma de representar la energía es como energía potencial, esto hace referencia a la capacidad de que se libere energía a partir de la ocurrencia de ciertos sucesos. En sistemas químicos, la energía potencial está relacionada con la probabilidad de que los átomos reaccionen entre sí. En resumen, esta ley implica que cualquier energía liberada en la formación de enlaces químicos puede liberarse como calor, puede usarse para romper otros enlaces o simplemente almacenarse de alguna forma.

Otros conceptos importantes en termodinámica son la entropía y entalpía. La entropía es una medida de desorden o nivel de libertad del sistema, mientras que la entalpía mide

la facilidad con la que la energía se absorbe o libera en el sistema, o dicho de otra forma, la cantidad de calor en el sistema.

La Segunda Ley de la Termodinámica establece que la entropía total de un sistema y de su entorno siempre aumenta. Es posible que la entropía disminuya localmente siempre y cuando en otras partes del universo aumente en una magnitud igual o mayor. Esta disminución local por lo general conduce a una liberación de calor lo que aumenta la entropía del entorno.

La entropía (S) del sistema puede cambiar el curso de una reacción química por un total $\Delta_{sistema}$. La entalpía (H) del sistema se verá reducida en un total $\Delta H_{sistema}$ si fluye calor del sistema al entorno. Es necesario deteminar el cambio de la entropía del entono para poder aplicar la Segunda Ley, y este cambio depende de la temperatura porque el cambio en la entropía es mayor cuando el calor se pasa a un entorno parcialmente frío que cuando se encuentra a una temperatura alta y con un alto grado de desorden. Entonces, el cambio en la entropía del entorno es proporcional a la cantidad de calor transferido por el sistema e inversamente proporcional a la temperatura (T) del entorno. En los sistemas biológicos la temperatura absoluta (T), la cual se mide en grados kelvin (K), se considera constante por lo que el cambio en la entropía del entorno está dado por

$$\Delta S_{entorno} = -\Delta H_{sistema}/T. \tag{1}$$

El cambio total de entropía está dado por

$$\Delta S_{total} = \Delta S_{entorno} + \Delta S_{sistema}. \tag{2}$$

Sustituyendo la ecuación 1 en 2, se tiene que

$$\Delta S_{total} = \Delta S_{sistema} - \Delta H_{sistema}/T. \tag{3}$$

Al multiplicar por -T se obtiene

$$-T\Delta S_{total} = \Delta H_{sistema} - T\Delta S_{sistema}. \tag{4}$$

A la función de la Ecuación 4 se le conoce como energía libre y se utiliza para describir la energética de las reacciones bioquímicas tomando en cuenta tanto la entropía del sistema como del entorno. De manera más formal, la energía libre o energía libre de Gibbs (en honor a Josiah W. Gibbs quien desarrolló dicha función en 1878) está dada por la función

$$\Delta G = \Delta H_{sistema} - T\Delta S_{sistema}. \tag{5}$$

Como se mencionó, la Segunda Ley establece que debe aumentar la entropía del universo para que una reacción ocurra de forma espontánea y en la Ecuación 3 se muestra que la entropía total aumentará únicamente si se cumple que

$$\Delta S_{sistema} > \Delta H_{sistema}/T. \tag{6}$$

Dicho lo anterior, se tiene que $T\Delta S_{sistema} > \Delta H$, lo que indica que la entropía sólo aumentará si

$$\Delta G = \Delta H_{sistema} - T\Delta S_{sistema} < 0. \tag{7}$$

Esto implica que el cambio de energía libre debe ser negativo para que una reacción ocurra de forma espontánea y esto incluye un aumento en la entropía total del universo (Berg *et al.*, 2008).

2.5.0.2. Termodinámica del ADN

La hibridación del ADN consiste en la unión de dos cadenas sencillas de ADN complementarias, produciendo así ADN de cadena doble o dúplex. Es posible que una cadena sencilla interactúe con otra a pesar de no ser completamente complementarias, sin embargo, esto creará una molécula de ADN de doble cadena imperfecta. En el proceso de hibridación, la temperatura es importante ya que si ésta disminuye también lo hace el rigor de hibridación. Un concepto importante es el rigor de hibridación, éste consiste en el grado de interacción entre nucleótidos y es posible medirlo con la energía libre de Gibbs. Si el rigor de hibridación disminuye, las cadenas de ADN presentan mayor probabilidad de hidridarse con pares de bases que no sean complementarias y cuando aumenta ocurre la desnaturalización del ADN. La desnaturalización es cuando ocurre la separación de una cadena doble de ADN en dos cadenas sencillas. La temperatura de fusión $T_m(\mathrm{C}^\circ)$ es

la temperatura donde se consigue que más de la mitad de la concentración de moléculas de ADN de doble cadena se desnaturalicen.

Existen diferentes métodos para predecir las reacciones de hidridación según el tipo de complejo, por ejemplo, en el caso de complejos de ADN/ADN el más utilizado es el modelo del vecino más cercano (SantaLucia y Hicks, 2004) principalmente debido a que es bastante exacto y sencillo de calcular; en el caso de complejos ARN/ARN se puede utilizar el modelo propuesto por Xia *et al.* (1998) y para ADN/ARN el de Sugimoto *et al.* (1995) o de Watkins y colaboradores (2011).

El modelo del vecino más cercano indica que la estabilidad de una molécula de doble cadena de ADN depende del par de bases que constituyen el vecino más cercano. Existen 10 posibles interacciones, las cuales se muestran en la Tabla 1 donde la diagonal indica que las secuencias se encuentran en orientación antiparalela; la corrección de simetría sólo se aplica a los dúplex auto-complementarios, además se aplica una penalización a cada dúplex que termine con la secuencia AT y una penalización de $+1.0\,kcal/mol$ para ΔG°. Las unidades para ΔH° y ΔG° son en $kcal/mol$ y las unidades de ΔS° son cal/K por mol de interacción.

Un ejemplo de aplicación de los parámetros del modelo del vecino más cercano puede ser calcular la energía libre para una molécula de doble cadena de ADN (SantaLucia y Hicks, 2004), por ejemplo, dada:

$$5'-\ \text{CGTTGA}\ -3'$$
$$3'-\ \text{GCAACT}\ -5'$$

El cálculo de su energía libre a 37° está dado por

$$\Delta G_{37}(total) = \Delta G_{37\,i} + \Delta G_{37\,s} + \Delta G_{37\,stack} + \Delta G_{37\,terminal\,AT}, \tag{8}$$

donde $\Delta G_{37\,i}$ corresponde a energía de iniciación de la hélice, $\Delta G_{37\,s}$ es la energía de corrección por simetría, $\Delta G_{37\,stack}$ es la energía generada por el dúplex y $\Delta G_{37\,terminal\,AT}$

es la penalización por terminar en una secuencia AT, entonces

$$\Delta G_{37}(total) = \Delta G_{37\,i} + \Delta G_{37\,s} + \Delta G_{37}(CG/GC) + \Delta G_{37}(GT/CA) +$$
$$\Delta G_{37}(TT/AA) + \Delta G_{37}(TG/AC) + \Delta G_{37}(GA/CT) + \Delta G_{37\,terminal\,AT}.$$

Los valores de los parámetros se obtienen a partir de la Tabla 1, por lo que

$$\Delta G_{37}(predicho) = 1.96 + 0 - 2.17 - 1.44 + 1.0 - 1.45 - 1.30 + 0.05.$$

Finalmente, se tiene que

$$\Delta G_{37}(predicho) = -5.35\,kcal\,mol^{-1}.$$

Tabla 1: Parámetros termodinámicos del vecino más cercano para pares de bases en 1 M de NaCl a 37° C. Adaptado de (Santa Lucía y Hicks, 2004).

Interacción	ΔH°	ΔS°	ΔG°
AA/TT	-7.6	-21.3	-1.00
AT/TA	-7.2	-20.4	-0.88
TA/AT	-7.2	-21.3	-0.58
CA/GT	-8.5	-22.7	-1.45
GT/CA	-8.4	-22.4	-1.44
CT/GA	-7.8	-21.0	-1.28
GA/CT	-8.2	-22.2	-1.30
CG/GC	-10.6	-27.2	-2.17
GC/CG	-9.8	-24.4	-2.24
GG/CC	-8.0	-19.9	-1.84
Iniciación	+0.2	-5.7	+1.96
Penalización terminal AT	+2.2	+6.9	+0.05
Correción por simetría	0.0	-1.4	+0.43

2.5.0.3. Cinética química estocástica

La cinética química estudia la velocidad con la que ocurren las reacciones, los factores que intervienen en ella y el mecanismo a través del cual los reactivos se transforman en productos; resumiendo, la cinética estudia la manera en que la concentración de algunas

especies moleculares varía en función del tiempo. La velocidad de una reacción química se define como la derivada de la concentración de un reactivo o producto con respecto al tiempo y convertida siempre en un valor positivo. De manera más formal, la velocidad de una reacción se define como el límite al que tiende el cociente entre la variación de la concentración de un reactivo o producto y el intervalo del tiempo cuando éste tiende a cero:

$$v = \lim_{\Delta t \to 0} \frac{\Delta[concentración]}{\Delta t} = \frac{d[concentración]}{dt}. \tag{9}$$

La velocidad se expresa en unidades de concentración/tiempo, por ejemplo Ms^{-1}, μMmin^{-1}, nMs^{-1}, etc. Las unidades de la constante de velocidad siempre incluyen el inverso de una unidad de tiempo (seg^{-1}, min^{-1}, etc.) y también pueden contener el inverso de una o más concentraciones.

Las ecuaciones o leyes de velocidad indican la velocidad como función matemática de una constante cinética o de velocidad (la cual se representa por medio de la letra k) y por las concentraciones de cada una de las especies moleculares que intervienen en la reacción. Para una reacción del tipo $aA + bB \rightarrow cC + dD$, lo anterior se expresa como

$$v = k[A]^m[B]^n, \tag{10}$$

donde v es la velocidad de la reacción, k la constante de velocidad cuyo valor depende de cada reacción, $[A]$ y $[B]$ indican la concentración molar de A y B, respectivamente, y los valores de los exponentes m y n se calculan de manera experimental. El orden cinético global de una reacción se determina por la suma de los exponentes a los que se encuentran elevadas las concentraciones de los reactivos en la ecuación de velocidad ($m+n$). El orden parcial de la reacción o el orden con respecto a un reactivo corresponde al exponente al que se eleva la concentración de dicho reactivo en la ecuación de velocidad (la reacción sería de orden m con respecto al reactivo A y de orden n con respecto al reactivo B). El orden cero significa que la velocidad de la reacción no varía cuando cambia la concentración de una especie, cabe mencionar que no existen reacciones bioquímicas de este orden. En los procesos de primer orden las unidades de la constante de velocidad son de tiempo^{-1} y en los de segundo orden son de M^{-1}tiempo^{-1}. Existen 4 factores que influyen en la velocidad de las reacciones: el estado físico de los reactivos, la concentra-

ción, la temperatura y los catalizadores. La velocidad de las reacciones se incrementa al aumentar la concentración de los reactivos, ya que se produce un aumento en el número de colisiones entre partículas.

En el enfoque determinístico las ecuaciones de las reacciones químicas se resuelven de manera matemática, es decir, las reacciones se convierten a ecuaciones diferenciales ordinarias. Para el caso del enfoque estocástico, el cual es más apropiado para modelar la naturaleza de las reacciones en biología molecular (Polanski y Kimmel, 2007), los procesos químicos se describen mediante una sola ecuación diferencial a la que se le llama la ecuación maestra. En los modelos estocásticos, a diferencia de los determinísticos donde las concentraciones de reactantes se miden en moles por litro (M), las concentraciones se representan por números enteros de cada especie de moléculas en el sistema. Para realizar esta conversión, es necesario conocer el volumen del contenedor V (en litros), entonces, para una concentración del reactante X de $[X]M$ en un volumen de V litros existen $[X]V$ moles de X, es decir, $n_A[X]V$ moléculas, donde $n_A \simeq 6.023 \times 10^{23}$ indica el número de moléculas en un mol y se denomina la constante de Avogadro. A continuación se detalla la conversión de concentraciones a número de moléculas de acuerdo al tipo de reacción:

Reacciones de orden cero. Considerando la reacción $\emptyset \longrightarrow X$, dado que la constante de velocidad es $k\text{M}s^{-1}$, para un volumen V, el producto X se genera a una velocidad de n_AkV moléculas por segundo. La constante de velocidad estocástica es c moléculas por segundo por lo que $c = n_AkV$.

Reacciones de primer orden. Considerando la reacción $X \longrightarrow ?$, debido a que la constante de velocidad es $k[X]\text{M}s^{-1}$, para un volumen V de $[X]$ se tiene que la concentración $[X]$ es igual a $x = n_A[X]V$ moléculas, la cual decrece a una velocidad de $n_A[X]V = kx$ moléculas por segundo. Dado que la constante de velocidad estocástica es cx moléculas por segundo se tiene que $c = k$, lo que indica que las constantes de velocidad estocásticas y determinísticas son iguales para las recciones de este tipo.

Reacciones de segundo orden. Considerando la reacción $X + Y \longrightarrow ?$ y siendo $k[X][Y]\text{M}s^{-1}$ la constante de velocidad, para un volumen V los reactivos se consumen a una velocidad de $n_A[X][Y]V = kxy/(n_aV)$ moléculas por segundo. La constante de velocidad estocástica corresponde a cxy moléculas por segundo por lo que $c = k/n_AV$.

Para las reacciones de dimerización $2X \longrightarrow ?$ la constante de velocidad es $k[X]^2$, por lo que la reacción se lleva a cabo a una velocidad de $n_A2k[X]^2V = 2kx^2/(n_AV)$ moléculas por segundo, siendo $c = 2k/n_AV$ la velocidad de reacción estocástica.

2.6. Diagnóstico molecular

A diferencia de sus contrapartes de silicio, las compuertas o circuitos basados en ADN al presentar interfaces naturales con los sistemas químicos y biológicos se pueden utilizar en una amplia gama de aplicaciones que incluyen el diagnótico molecular de alguna enfermedad a partir de marcadores biológicos como entradas. Básicamente, el término diagnóstico molecular se refiere a la aplicación de tecnologías moleculares para ciertas cuestiones clínicas como lo son el análisis de biomoléculas para diagnosticar, monitorear y entender el origen de las enfermedades humanas. El término en sí abarca múltiples tecnologías (PCR, secuenciación de ADN, biochips, microarreglos, etc.) que toman como base el ADN, ARN, genes o proteínas para realizar exámenes de diagnóstico.

2.6.1. Marcadores biológicos

Los biomarcadores o marcadores biológicos son una característica que puede ser medida de manera objetiva y que funciona como indicador para evaluar algún proceso patológico o una respuesta farmacológica a una intervención terapéutica. Los biomarcadores clásicos corresponden a ciertas alteraciones físicas medibles, por ejemplo, niveles de lactato en sangre después de ejercitarse o niveles de glucosa en sangre cuando se tiene *diabetes mellitus* o anomalías en la presión sanguínea. Cualquier alteración en una célula, en ADN, ARN, metabolitos o proteínas puede considerarse como un marcador biológico. Actualmente, los biomarcadores por lo general se refieren a marcadores biológicos y de acuerdo a su función se dividen en las siguientes categorías (Jain, 2010):

- Los que monitorean el progreso de una enfermedad a través del tiempo y están

relacionados con medidas clínicas conocidas.

- Los que ayudan a detectar efectos de drogas o medicamentos.
- Los que funcionan como criterios de valoración en ensayos clínicos.

Los marcadores biológicos no sólo permiten detectar respuestas biológicas a medicamentos experimentales sino que también son útiles para descubrir nuevos blancos para intervención terapéutica por lo que compañías farmacéuticas y de biotecnología los están empleando para el descubrimiento de nuevos medicamentos. Los biomarcadores se clasifican en:

- Biomarcadores de enfermedad: tipo 0.
- Biomarcadores de diagnóstico.
- Biomarcadores para descubrimiento de medicamentos.
- Biomarcadores de predicción de enfermedades.
- Biomarcadores para detectar efectos de medicamentos o drogas: tipo I.
- Biomarcadores de translación (clínicos y pre-clínicos).
- Biomarcadores como criterios de valoración en ensayos clínicos.
- Biomarcadores válidos (validados en ensayos clínicos).

Los marcadores moleculares pueden ser moléculas simples como metabolitos, carbohidratos (e.g. glucosa), esteroides y lípidos; también pueden serlo moléculas no tan simples como los péptidos y proteínas (e.g. insulina, hemoglobina A y C, entre otros). Otros biomarcadores más complejos son células como plaquetas, células T o anticuerpos. Como se mencionó anteriormente, existen biomarcadores de ADN y ARN, siendo de interés para este trabajo los de ARN, particularmente los miARNs.

2.6.1.1. MicroARN (miARN)

Los micro (mi)ARNs son cadenas sencillas de ARN no codificante de aproximadamente 20-23 nucleótidos de longitud los cuales se encargan de regular la expresión génica en numerosos procesos celulares. Generalmente, estas moléculas pueden modificar la expresión del ARN mensajero (mARN) reduciendo su traducción y estabilidad. Lo anterior incluye el mARN de los genes mediadores de procesos de tumorigénesis (e.g. inflamación, regulación del ciclo celular, apoptosis e invasión, entre otros) por lo que se ha observado que tienen la propiedad de suprimir tumores o de promover su crecimiento (Schmitz *et al*., 2013). Un ejemplo de lo anterior es el miR-21 que es un típico oncogen de microARN cuya expresión ha sido confirmada en varios tipos cáncer, tales como cáncer de mama, de pulmón, de esófago, pancreático, colorrectal y carcinoma hepato-celular (Hamano *et al.*, 2011).

Capítulo 3. Fundamentos de computación

El cómputo basado en silicón se lleva a cabo mediante lógica binaria, donde las entradas y salidas se representan en formato binario. Una variable binaria es toda variable que sólo puede tomar dos valores, los cuales corresponden a estados distintos y exclusivos; dichos estados se representan con los símbolos 1 y 0 e indican el estado electrónico de encendido (circula corriente) y apagado (no circula corriente), respectivamente. En la actualidad existen aparatos electrónicos capaces de realizar operaciones muy complejas, éstos están basados en circuitos que a su vez están formados por elementos más básicos llamados compuertas, que llevan a cabo una serie de operaciones lógicas. En este capítulo se presentan los fundamentos básicos en computación para el diseño de compuertas lógicas a partir de ADN.

3.1. Circuitos lógicos

Los circuitos digitales están diseñados para responder con salidas dentro de los niveles de voltaje de 0 y 1 según las entradas correspondientes. La lógica de un circuito es la forma en que un circuito digital responde a la entrada. Cada circuito está condicionado a ciertas reglas lógicas, por este motivo también se les conoce con el nombre de circuitos lógicos (Tocci y Widmer, 2003).

3.2. Compuertas lógicas

Son el circuito más elemental, emiten una salida de acuerdo a sus variables de entrada según su tabla de verdad. La tabla de verdad de una compuerta lógica nos indica el estado de salida de acuerdo a la combinación de los valores de sus variables de entrada. Un conjunto de compuertas lógicas es completo si se puede implementar cualquier función lógica a partir de éste y se pueden implementar todas las demás compuertas de otro conjunto completo, por ejemplo:

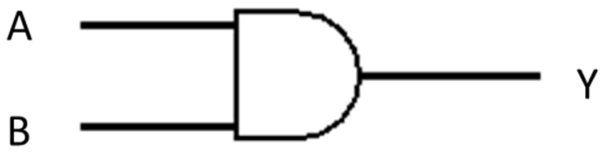
- Compuertas AND, OR y NOT.
- Compuertas AND y NOT.
- Compuertas OR y NOT.

- Compuertas NAND.
- Compuertas NOR.

Cualquier expresión booleana se compone de algunas combinaciones entre las operaciones básicas AND, OR y NOT. No obstante, también es posible ejecutar cualquier operación booleana usando solamente compuertas NAND y NOR, ya que estas compuertas combinándolas por sí solas realizan las tres operaciones booleanas básicas (AND, OR, NOT) por lo que se le llama universales (Tocci y Widmer, 2003).

3.2.1. Compuerta AND

Esta compuerta representa la conjunción lógica, por lo que dicha compuerta solamente emite una salida positiva cuando todas las entradas son positivas. En la Figura 8(a) se puede observar la representación simbólica de la compuerta AND. Su tabla de verdad se muestra en la Figura 8(b), donde es posible observar que el único caso en que la compuerta AND emite un 1 lógico es cuando ambas entradas corresponden a un 1.



(a)

Entradas		Salida
A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

(b)

Figura 8: Compuerta AND: (a) representación simbólica y (b) tabla de verdad.

3.2.2. Compuerta OR

La compuerta OR corresponde a la operación de adición lógica OR. Tal y como se observa en la Figura 9(b), la operación OR produce un 1 cuando por lo menos una de las entradas tiene un valor lógico de 1 y la única manera de que se produzca un 0 como salida es si todas las entradas presentan el valor de 0. Se representa con el símbolo de la Figura 9(a).

(a)

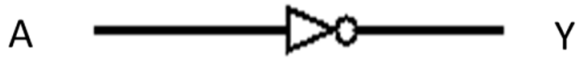
Entradas		Salida
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

(b)

Figura 9: Compuerta OR: (a) representación simbólica y (b) tabla de verdad.

3.2.3. Compuerta NOT

A diferencia de las anteriores, la compuerta NOT requiere una sola variable y comúnmente se le conoce como inversor ya que su nivel de salida siempre es contrario al nivel de entrada, tal como se puede observar en la Figura 10(b). La representación simbólica de la compuerta inversora se encuentra en la Figura 10(a).



(a)

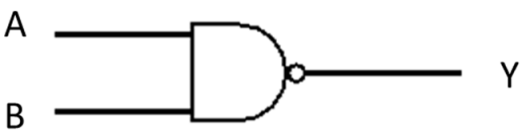
Entradas	Salida
A	Y
0	1
1	0

(b)

Figura 10: Compuerta NOT: (a) representación simbólica y (b) tabla de verdad.

3.2.4. Compuerta NAND

La compuerta NAND opera de la misma forma que una compuerta AND seguida de un inversor. La Figura 11(b) corresponde a la tabla de verdad de la operación NAND la cual muestra que la salida es exactamente el inverso de la salida de una compuerta AND. La compuerta AND sólo devuelve un 1 cuando ambas entradas son 1, mientras que la NAND solamente regresa un 0 si ambas entradas son 1. La representación gráfica de una compuerta NAND de dos entradas se muestra en la Figura 11(a). Como se puede observar, es el mismo símbolo de la compuerta AND seguido de un pequeño círculo que indica inversión. Es la compuerta más utilizada debido a que con ella es posible realizar circuitos complejos con pocos componentes.



(a)

Entradas		Salida
A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

(b)

Figura 11: Compuerta NAND: (a) representación simbólica y (b) tabla de verdad.

3.2.5. Compuerta XOR

También se le denomina anticoincidencia y comparador, la compuerta de OR exclusivo es muy utilizada en dispositivos electrónicos digitales. Funciona únicamente para dos entradas y como se muestra en la Figura 12(b), su salida es 1 solamente cuando una de sus dos entradas es 1; en todos los demás casos la salida permanece en 0. En la Figura 12(a) se encuentra la representación gráfica de una compuerta XOR. Es posible obtener una compuerta XOR de varias maneras, por ejemplo, al combinar las compuertas AND, OR y NOT; empleando solamente compuertas NAND; o empleando únicamente compuertas NOR.

(a)

Entradas		Salida
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

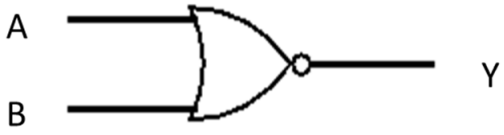
(b)

Figura 12: Compuerta XOR: (a) representación simbólica y (b) tabla de verdad.

3.2.6. Compuerta NOR

La compuerta NOR opera de la misma forma que una compuerta OR seguida de un inversor. La Figura 13(b) corresponde a la tabla de verdad de la operación NOR, la cual muestra que la salida es exactamente el inverso de la salida de una compuerta NOR. La compuerta OR sólo devuelve un 1 cuando cualquiera de las entradas tiene valor de 1,

mientras que la NOR solamente regresa un 1 si ambas entradas son 0. La representación gráfica de una compuerta NOR de dos entradas se muestra en la Figura 13(a). Como se puede observar, es el mismo símbolo de la compuerta OR seguido de un pequeño círculo que indica inversión.



(a)

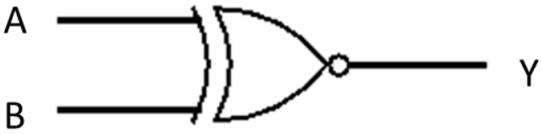
Entradas		Salida
A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

(b)

Figura 13: Compuerta NOR: (a) representación simbólica y (b) tabla de verdad.

3.2.7. Compuerta XNOR

La compuerta XNOR opera de la misma forma que una compuerta XOR seguida de un inversor y sólo tiene dos entradas, también se le conoce como operador de equivalencia. La Figura 14(b) corresponde a la tabla de verdad de la operación XNOR, la cual muestra que la salida es exactamente el inverso de la salida de una compuerta XOR. La compuerta XNOR devuelve un 1 cuando ambas entradas tienen exactamente el mismo valor (ya sea 0 ó 1), mientras que en los demás casos regresa 0. La representación gráfica de una compuerta XNOR se muestra en la Figura 14(a). Como se puede observar, es el mismo símbolo de la compuerta XOR seguido de un pequeño círculo que indica inversión.



(a)

Entradas		Salida
A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1

(b)

Figura 14: Compuerta XNOR: (a) representación simbólica y (b) tabla de verdad.

3.3. Simulación estocástica

El término simulación se refiere al procesamiento de un problema real a través de un experimento en un ambiente controlado. El ambiente usualmente se obtiene mediante equipo de cómputo, lo que permite que se pueda reproducir el sistema de estudio a menor escala. Existen algunos sistemas donde algunos o todos los componentes están sujetos a variaciones aleatorias, por lo que no pueden describirse por una regla matemática exacta, sino que la probabilidad es necesaria. La simulación estocástica se encarga de la simulación de este tipo de sistemas y tiene como objetivo principal el reproducir dichas variables aleatorias en un ambiente controlado a pesar de la complejidad que ello implica (Gamerman y Lopes, 2006). Algunas aplicaciones donde la simulación estocástica puede resultar útil son (Nelson, 2002):

- Manufactura (control de inventarios, evaluación de calidad de procesos).
- Cuidado de la salud (toma de decisiones medicas, dotación de personal en hospitales).
- Computación (diseño de configuraciones de hardware, protocolos de sistemas operativos).
- Comunicaciones (avaluación de fiabilidad de una red).
- Negocios (comportamiento del consumidor, logística de distribución de productos).
- Defensa (estrategias de combate).
- Biología (genética de poblaciones, esparcimiento de epidemias).

Para simular la naturaleza aleatoria de las reacciones químicas es necesario el uso de modelos estocásticos discretos, los cuales en su mayoría están basados en el trabajo de Gillespie (1997), quien propuso un algoritmo de simulación estocástica bien fundamentado en cuanto a cuestiones físicas, matemáticas y teoría cinética (Barbuti *et al.*, 2005).

3.3.1. Algoritmo de Gillespie

Gillespie (1997) propuso un método para generar una trayectoria estadísticamente correcta, o dicho de otra manera, una posible solución a una ecuación estocástica. El algoritmo simula la evolución en el tiempo de un sistema de reacciones químicas detemi-nando cuándo y de qué tipo será la siguiente reacción que ocurrirá. Esto se calcula con base en una constante de reacción estocástica, la cual por lo general es desconocida y predicha de manera aproximada a partir de una constante cinética determinista. En el algoritmo, primero se inicializan el número de moléculas del sistema, las constantes de reacción y el generador de números aleatorios. Después viene el paso de Monte Carlo, donde se generan dos números aleatorios que determinan la siguiente reacción y el inter-valo de tiempo; la probabilidad de que una reacción sea elegida es proporcional al número de moléculas del sustrato. Posteriormente, el tiempo se incrementa por un valor generado aleatoriamente en el paso de Monte Carlo y se actualizan las moléculas basadas en la reacción ocurrida; el algoritmo continúa iterando hasta que el número de moléculas sea igual a cero o se exceda el tiempo de simulación. El Algoritmo 1 corresponde al algoritmo de Gillespie clásico.

Algoritmo 1 Algoritmo de Gillespie

Entrada: un medio m donde existen especies $x_1, \ldots, x_n$ sometidas a posibles reacciones químicas $r_1, \ldots, r_q$ con sus correspondientes constantes de reacción $p_1, \ldots, p_n$; tiempo máximo de simulación T_{max}.

Salida: reacción química a ejecutar y el tiempo de espera correspondiente para encontrar el siguiente estado.

1: **mientras** $t < T_{max}$ **hacer**
2: Calcular $a_0 = \sum_{j=1}^{q} p_j$, siendo p_j la constante estocástica de la reacción j.
3: Generar dos números aleatorios b_1 y b_2 uniformemente distribuidos sobre el intervalo $(0, 1)$.
4: Calcular el tiempo de espera para la siguiente reacción $\tau_m = \frac{1}{a_0} \ln(\frac{1}{b_1})$.
5: Elegir el índice j_0 de la siguiente reacción química que verifica $\sum_{k=1}^{j_0-1} p_k < b_2 * a_0 \leq \sum_{k=1}^{j_0} p_k$.
6: Actualizar el tiempo $t = t + \tau_m$, y el número de moléculas de acuerdo al evento ocurrido.
7: **fin mientras**
8: **regresar** (τ_m, j_0, m).

3.4. Cadenas de Markov en tiempo continuo

Para modelar un sistema de reacciones a nivel molecular, donde se obedecen las leyes de probabilidad en lugar de las determinísticas, es posible determinar la probabilidad de que dicho sistema se encuentre en un estado dado en un tiempo dado t_2; esto se deduce a partir del conocimiento del estado del sistema en un tiempo anterior t_1 y es independiente del pasado del sistema, es decir, todos los estados anteriores al estado inmediatamente anterior al estado actual son irrelevantes para la predicción. A los procesos estocásticos que representan sistemas que satisfacen esta condición se les conoce como procesos de Markov. Las cadenas de Markov son un tipo especial de proceso de Markov, son modelos probabilísticos que permiten la predicción de la evolución y el comportamiento a corto y a largo plazo de ciertos sistemas. Propiamente, se definen como un proceso estocástico con propiedad Markoviana con un número finito de estados (M) y con probabilidades de transición estacionarias. Como se mencionó anteriormente, una cadena de Markov es un proceso estocástico el cual es visto como una serie transiciones entre ciertos valores llamados estados del proceso, el conjunto de estados alcanzables puede ser finito o infinito contable; una cadena de Markov tiene la propiedad de que al encontrarse en un estado dado, la ley de probabilidad del futuro desarrollo del proceso se determina únicamente a partir del estado, es decir, no depende de cómo es que el proceso llegó a dicho estado. Las cadenas de Markov se clasifican como discretas o continuas según la naturaleza de su parámetro (Parzen, 1999). Para este trabajo, las de interés son las cadenas de Markov en tiempo continuo.

3.4.1. Definición formal de proceso de Markov

Un proceso estocástico de parámetro discreto $\{X(t),\, t = 0, 1, 2, \ldots\}$ o un proceso estocástico de parámetro continuo $\{X(t),\, t \geq 0\}$ es un proceso de Markov si para cualquier conjunto de n puntos de tiempo $t_1 < t_2 < \ldots < t_n$ en el conjunto de índices del proceso, la distribución condicional de $X(t_n)$ para valores dados de $X(t_1), \ldots, X(t_{n-1})$ depende solamente del valor conocido más reciente, es decir, de $X(t_{n-1})$. En otras palabras, para cualquiera números reales $x_1, \ldots, x_n$, se cumple que

$$P[X(t_n) \leq x_n | X(t_1) = x_1, \ldots, X(t_{n-1}) = x_{n-1}] = P[X(t_n) \leq x_n | X(t_{n-1}) = x_{n-1}]. \quad (11)$$

A lo que la Ecuación 11 se refiere es que dado el presente del proceso, el futuro es independiente del pasado. Los procesos de Markov se clasifican de acuerdo a la naturaleza del conjunto de índices del proceso (si son discretos o continuos) y de acuerdo a la naturaleza del espacio de estados del proceso.

Un número real x es un posible valor o un estado de un proceso estocástico $\{X(t),\, t \in T\}$ si existe un tiempo t en T tal que la probabilidad $P[x-h < X(t) < x+h]$ sea positiva para toda $h > 0$; al conjunto de posibles valores de un proceso estocástico se le conoce como el espacio de estados. El espacio de estados es discreto si contiene un número finito o contable de estados, si el espacio no es discreto, se le llama continuo.

Un proceso de Markov se describe por una función de probabilidad de transición, usualmente denotada por $P(x,\, t_0; E, t)$ ó $P(E, t \,|\, x, t_0)$, la cual representa la probabilidad condicional de que el estado del sistema en un tiempo t pertenezca al conjunto E dado que en un tiempo $t_0 < t$ el sistema se encuentra en el estado x. Se dice que un proceso de Markov tiene probabilidades de transición estacionarias o que es homogéneo en el tiempo si $P(x, t_0; E, t)$ depende de t y t_0 solamente a través de la diferencia $(t - t_0)$.

Una cadena de Markov es un proceso de Markov cuyo espacio de estados es discreto, por lo que generalmente se utiliza un conjunto de enteros $\{0, 1, \ldots\}$ como el espacio de estados.

3.4.2. Cálculo de probabilidad de transición

Suponiendo que las transiciones entre estados discretos $1, 2, \ldots, N$ pueden ocurrir en cualquier tiempo t, donde t es un número real; el proceso estocástico resultante se denota por $X(t)$ y se introduce una matriz de transiciones $P(t-s)$ de $N \times N$ con entradas

$$p_{ij}(t-s) = P[X(t) = j \,|\, X(s) = i]. \tag{12}$$

La propiedad Markoviana de $X(t)$ es equivalente a la ecuación de Chapman-Kolmogorov

$$p_{ij}(s+t) = \sum_{n=1}^{N} p_{in}(s) p_{nj}(t). \tag{13}$$

Utilizando la notación matricial $P(t)$, la Ecuación 13 puede reescribirse como

$$P(s+t) = P(s)P(t), \tag{14}$$

donde $s \geq 0$, $t \geq 0$ y

$$P(0) = I, \tag{15}$$

donde I es la matriz identidad. $P(t)$ es diferenciable y al calcular la derivada a partir de la Ecuación 14 se obtiene

$$\frac{d}{dt}P(t) = P'(t) = QP(t), \tag{16}$$

donde la matriz Q es llamada la matriz de intensidad de la cadena de Markov en tiempo continuo $X(t)$ y está dada por el límite de la derivada en cero,

$$Q = \lim_{t \to 0^+} \frac{dP(t)}{dt}.$$

Los procesos de Markov $X(t)$ utilizados en aplicaciones prácticas como por ejemplo, para modelar sistemas de reacciones a nivel molecular, comienzan con la definición de la matriz de intensidad Q, ya que dicho enfoque es más acertado (Polanski y Kimmel, 2007). Dada la matriz de intensidad Q es posible obtener la matriz de transiciones $P(t)$ a partir de la solución de 16 con la condición inicial 15, dando como resultado

$$P(t) = \exp(Qt) = \sum_{m=1}^{\infty} \frac{(Qt)^m}{m!}. \tag{17}$$

Para cada $t \geq 0$, $P(t)$ es una matriz estocástica y dada una distribución de probabilidad inicial $\pi(0)$ de los estados $1, 2, \ldots, N$, es posible calcular la distribución en el tiempo t a partir de

$$\pi(t) = \pi(0)P(t). \tag{18}$$

La construcción del proceso utilizando intensidades implica que la probabilidad de una transición $i \to j$ en el intervalo $(t, t+\Delta t)$ para cualquier estado i es igual a $q_{ij}\Delta t + o(\Delta t)$, por ejemplo

$$P[X(t+\Delta t) = j | X(t) = i] = q_{ij}\Delta t + o(\Delta t). \tag{19}$$

Para los elementos de la matriz de intensidad Q se define

$$q_{ii} = -\sum_{j \neq i} q_{ij}. \tag{20}$$

Por último, es posible derivar 16, 17 usando 19 y 20.

Capítulo 4. Cómputo molecular

El cómputo molecular es un campo multidisciplinario que involucra ciencias como la biología molecular, física, matemáticas, ciencias computacionales, nanotecnología y biotecnología. En computación molecular, a diferencia del cómputo convencional, la información se guarda en biomoléculas en lugar de registros y el procesamiento de dicha información se realiza mediante la manipulación de estas biomoléculas en lugar de circuitos digitales. El área se puede dividir en dos categorías, la primera busca la forma de implementar mecanismos moleculares que emulen los componentes de cómputo tradicionales (por ejemplo, transistores a partir de semiconductores basados en carbón o compuertas lógicas moleculares); la segunda trata de la investigación de nuevos paradigmas de cómputo basados en características específicas de las moléculas (Ignatova *et al.*, 2008).

4.1. Orígenes y antecedentes

Formalmente, el ADN se utiliza como elemento de cómputo a partir del trabajo de Adleman (1994), donde se demostró por primera vez una instancia del problema del camino Hamiltoniano con 7 vértices. Sin embargo, Head (1987) provee las bases para comprender el modelo computacional de operaciones biomoleculares importantes que ocurren en la naturaleza; dicho trabajo ha influenciado tanto investigaciones en computación molecular como en teoría del lenguaje formal y dio origen a los sistemas de empalme, mejor conocidos como sistemas H, que fueron nombrados así en su honor.

4.1.1. Experimento de Adleman

Leonard Adleman (1994) propuso que el ADN podría tener aplicaciones en las técnicas de investigación computacional, esto gracias a su composición, principalmente por la gran cantidad de posibles combinaciones entre nuclétidos que lo conforman. En este trabajo, Adleman puso a prueba su teoría con el problema del vendedor viajero, donde se utilizaron cadenas de ADN de 20 bases de longitud de manera aleatoria para representar químicamente cada una de las ciudades y una cadena complementaria de 20 bases longitud que se superpone a cada ciudad hasta la mitad para representar cada camino o

arista. Mediante unos cuantos gramos de ADN de cada una de las ciudades y caminos en un tubo de ensayo y gracias a las propiedades de hibridación del ADN se consiguieron más de 10^9 respuestas en unas cuantas horas. Las soluciones, es decir, los caminos válidos debían comenzar en la ciudad A y terminar en la ciudad G y pasar por las 7 ciudades al menos una vez. Cabe mencionar que el problema del camino de hamiltoniano (HPP) pertenece a la clase NP-Completa por lo que hasta la fecha no es posible solucionarlo mediante computadoras digitales convencionales utilizando algoritmos de fuerza bruta.

En resumen, el algoritmo de Adleman resuelve un caso del HPP de 7 vértices en tiempo lineal mediante la codificación del problema en cadenas de ADN y mediante operaciones biológicas. Cada vértice v_i del grafo se representa por una cadena de ADN y cada arista $e_{i,j}$ que conecta los vértices v_i con v_j corresponde a una cadena de ADN complementaria a la segunda mitad del extremo $3'$ de la secuencia del vértice v_i y a la primera mitad del extremo $5'$ de v_j. Entonces, dado un grafo con n vértices, los pasos son los siguientes:

1. Generar un conjunto de caminos aleatorios a través del grafo.
2. Para cada camino en el conjunto: a) Verificar si el camino comienza en el vértice inicial y termina con el último vértice. Si no cumple con esta condición, el camino se elimina del conjunto. b) Verificar que el camino pase exactamente por los n vértices, de no ser así, remover el camino del conjunto. c) Para cada vértice, verificar si el camino pasa solamente una vez a través de dicho vértice (esto es para eliminar los caminos que tengan vértices repetidos), si no remover el camino del conjunto.
3. Si el conjunto no es vacío, reportar la existencia de un camino Hamiltoniano. Si el conjunto es vacío, reportar que no existe dicho camino.

En cuestión de operaciones biológicas, el conjunto de los caminos se crea a partir de la hibridación de las cadenas de ADN que codifican vértices y aristas del grafo. Los caminos que tienen un inicio y fin correcto, es decir, que inician y terminan con el vértice inicial y final correspondientes, se amplifican con PCR. Los caminos que contienen exactamente los n vértices de grafo se extraen por medio de electroforesis en gel. Para eliminar los

caminos con vértices repetidos se utiliza un ciclo de filtrado lineal y los caminos finales se amplifican con PCR para detectar las soluciones (si es que las hay) por medio de electroforesis en gel.

4.2. Modelos clásicos de cómputo con ADN

En esta sección se presentan los modelos clásicos de cómputo biomolecular, los cuales se consideran no autónomos, es decir, es necesario que una persona monitoree el proceso y que realice las operaciones de laboratorio correspondientes. Inicialmente, el cómputo molecular se empleó para resolver problemas de tipo NP-Completo (hasta ahora imposibles de resolver con cómputo convencional) pero esto no resulta del todo eficiente debido a que las operaciones son lentas y suelen ser suceptibles a errores y el volumen de ADN que se requiere para codificar un problema crece exponencialmente de acuerdo con su tamaño.

4.2.1. Modelo de filtrado

Básicamente, los modelos de filtrado buscan generar bibliotecas de moléculas de ADN conteniendo varias posibles soluciones a un problema y después filtrarlas para encontrar la respuesta si es que ésta existe. Ejemplos de problemas abordados con este modelo son el problema de satisfacibilidad (Lipton, 1995) (Gibbons *et al.*, 1996) y el del máximo clique (Ouyang *et al.*, 1997).

4.2.2. Modelo de concatenación

También conocidos como *splicing systems* por su nombre en inglés, son modelos basados en las operaciones biomoleculares propuestas por Head (1987), las cuales utilizan técnicas de biología molecular como primitivas de un lenguaje de programación para diseñar algoritmos. Está demostrado que los modelos de concatenación pueden emular una máquina de Turing (Harju y Margenstern, 2005). El modelo básico consiste en una población inicial de moléculas de ADN de cadena doble, varias enzimas de restricción y ligasa contenidos en un tubo de ensayo. Es posible representar el modelo de manera matemática mediante un conjunto de cadenas (lenguaje inicial), un conjunto de operaciones de corte y otro conjunto de operaciones de concatenación.

4.2.3. Modelo de etiquetas

El modelo de etiquetas o en inglés *sticker model* es un modelo de filtrado clásico que utiliza operaciones de separación/filtrado para realizar el procesamiento y es una implementación de una máquina de registros; cada registro consiste en una molécula de ssADN de longitud fija para representar información binaria, la cual se divide en subcadenas o bits (de longitud constante de nucleótidos). Las etiquetas se utilizan para representar el estado de encedido o apagado de los bits y son cadenas complementarias a las subcadenas de los registros. Al hibridarse una etiqueta con su correspondiente subcadena complementaria se enciende el bit, de otro modo, éste se encuentra apagado. A los complejos formados por un registro y etiquetas se les llama complejos de memoria y mediante ellos es posible representar cualquier número binario. Los tubos son colecciones de complejos de memoria que pueden contener varias copias de la misma cadena. Un algoritmo de etiquetas (*sticker algorithm*) recibe un tubo incial como parámetro de entrada y se define como una secuencia finita de operaciones utilizadas para manipular los tubos, las cuales consisten en: combinar dos tubos para formar un nuevo tubo (unión), separar un tubo en dos nuevos tubos (separación), y (encender) o (apagar) un determinado bit de cada registro en el tubo. La complejidad de estos algoritmos se obtiene mediante el número de pasos de operaciones de laboratorio. Comúnmente, el tubo inicial contiene un conjunto basto de posibles soluciones para después remover las que no son válidas aunque también existe la opción de que el tubo inicial contenga soluciones aproximadas y construir soluciones completamente válidas durante la ejecución del algoritmo. Cuando éste finaliza, pueden encontrarse uno o más tubos con la solución correcta si es que existe (Roweis *et al.*, 1998).

4.3. Modelos autónomos de cómputo con ADN

Debido a que los modelos clásicos de cómputo molecular presentaban algunos inconvenientes tales como el volumen exponencial de ADN conforme se incrementaba la dificultad del problema, operaciones lentas y la necesidad de personal calificado para monitorear y llevar a cabo los experimentos, surgió el cómputo autónomo como la segunda generación del cómputo molecular. Éste se basa en las propiedades de autoensamblado de las moléculas de ADN (en ocasiones se modula mediante enzimas) lo que le confiere

autonomía y la posibilidad de ser parcialmente programable. En esta sección se mencionan algunos de los modelos de cómputo autónomo básicos.

4.3.1. Autoensamblado de ADN

Este modelo se basa en las propiedades de autoensamblado del ADN y fue propuesto por Winfree *et al.* (1996). Con este modelo es posible realizar cómputo universal por medio de estructuras capaces de autoensamblarse para formar un plano bidimensional. Los cálculos se realizan a través de la formación de configuraciones bidimensionales conocidas como mosaicos que son capaces de unirse a otras estructuras mediante extremos pegajosos (*sticky ends*), los cuales son segmentos de nucleótidos en una cadena sencilla de ADN.

4.3.2. Cómputo con estructuras secundarias de ADN

El ADN monocatenario o de cadena sencilla (ssADN) puede formar una estructura secundaria (en inglés *hairpin*), donde la cadena forma un bucle o lazo alrededor de sí misma y se producen puentes de hidrógeno entre las bases que la conforman. Al cómputo con *hairpins* se le conoce como *Whiplash* PCR (del inglés *polymerase chain reaction*). El modelo fue introducido por Hagiya *et al.* (1997), el cual permitía evaluar una fórmula Booleana en un solo tubo en un tiempo relativamente corto y posteriormente desarrollado experimentalmente por Winfree (1998) para generar una solución para el CIRCUIT-SAT. El *Whiplash* PCR es una reacción donde el extremo 3′ de una cadena sencilla de ADN que toma forma de *hairpin* se extiende mediante una subsecuencia de sí misma como plantilla para la polimerización.

4.3.3. Autómatas moleculares

Benenson *et al.* (2001) fueron los primeros en proponer un modelo de autómata de estado finito basado en ADN tomando como referencia el modelo de Rothemund (1996) de una máquina de Turing. El modelo del autómata está formado por tres partes: la entrada, el hardware y el software. La entrada se codifica como ADN de doble cadena, el hardware consiste en las enzimas ligasa y endonucleasa *FokI* y el software corresponde a las reglas de transición que también se codifican como dsADN. El autómata procesa la

entrada por medio de cortes generados por la enzima de restricción *FokI* y mediante los procesos de hibridación y ligación por la enzima ligasa. De cada corte resulta una cadena con el estado actual y un extremo pegajoso (que codifica la siguiente entrada) expuestos, dicha cadena se hibrida con la molécula correspondiente a la regla de transición, repitiendo el proceso hasta que no existan más transiciones o se encuentre un terminador. Por último, se produce una molécula que codifica el estado final. A pesar de que el modelo es completamente autónomo y cada molécula realiza un cálculo independiente en paralelo, éste presenta los inconvenientes de que sólo se pueden utilizar dos símbolos y que el estado inicial y el final deben ser el mismo, por lo que Kuramochi y Sakakibara (2006) proponen un modelo más general para autómatas de estado finito que lleva como nombre *length-encoding*; donde la longitud de la cadena de entrada depende del número de estados del autómata pero no se soluciona el problema del estado final e inicial. Más tarde, Martínez-Pérez *et al.* (2009) proponen un modelo para resolver dicho problema.

4.4. Modelo de desplazamiento de cadenas de ADN

En cómputo molecular, el proceso de desplazamiento de cadenas usualmente está mediado por enzimas pero el mecanismo al que se refiere este trabajo se guía por la termodinámica de ADN y es independiente de enzimas. El desplazamiento de cadenas libre de enzimas se estudia desde 1970, por ejemplo en (Lee *et al.*, 1970); sin embargo, no es sino hasta el año 2000 con (Reynaldo *et al.*, 2000) que se aplica al campo de la nanotecnología y con ello se da pie a que se use para la construcción de dispositivos de ADN. Una reacción de desplazamiento de cadenas se describe como el proceso mediante el cual dos hebras con complementariedad parcial o completa hibridan entre sí, desplazando en el proceso una o más hebras pre-hibridadas (Figura 15). Usualmente, en los diseños de desplazamiento de cadenas de ADN se abstraen las secuencias de ADN en dominios, esto quiere decir que se representa un segmento de la secuencia por medio de un número o letra y el dominio complementario se representa con la misma letra o número acompañado de un asterisco. La cinética del desplazamiento de cadenas se controla mediante ciertos mecanismos llamados puntos de apoyo (o en inglés *toehold*s), que no son más que secuencias cortas de ADN encargadas de iniciar la reacción (Figura 16). Las reacciones de desplazamiento de cadenas son programables gracias al diseño

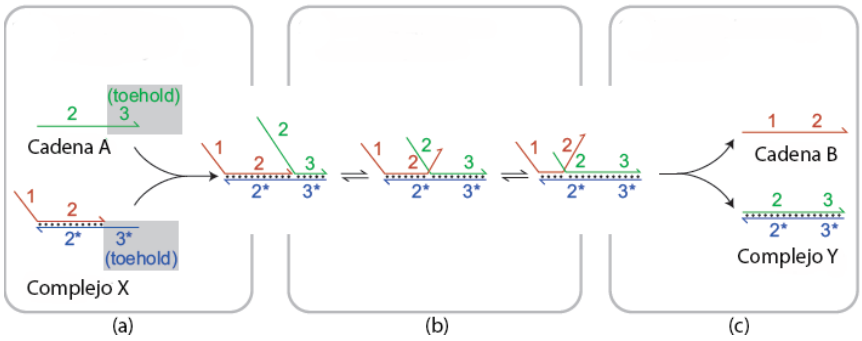


Figura 15: Ejemplo de reacción de desplazamiento de cadena: (a) inicialmente se cuenta con un complejo X que exhibe el *toehold* 3*, al sistema se agrega la cadena sencilla A que contiene la secuencia complementaria al *toehold* 3*; (b) por acción termodinámica, la cadena A desplaza a la cadena pre-hibridada; (c) se genera un nuevo complejo Y y se libera la cadena sencilla B. Adaptado de la referencia (Zhang y Seelig, 2011).

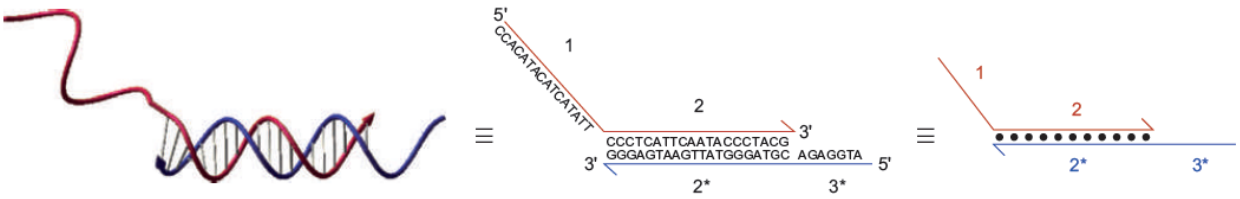


Figura 16: Representación de cadenas de ADN en el diseño de desplazamiento de cadenas. Aquí se muestran 3 dominios (1, 2, 3), los dominios complementarios se representan con el mismo número y un asterisco, por ejemplo el dominio complementario del dominio 2 es 2*. El dominio 3* corresponde al *toehold*. Tomado de la referencia (Zhang y Seelig, 2011).

meticuloso de las secuencias que se utilizan, por ejemplo, para obtener una reacción rápida y fiable, los dominios de los *toehold*s deben ser cortos (aproximadamente 6 nucleótidos), mientras que los dominios que se desplazan deben ser más largos (alrededor de 20 nucleótidos) (Soloveichik *et al.*, 2010). El uso de desplazamiento de cadenas de ADN como herramienta para construir mecanismos útiles en nanoingeniería fue introducido por Yurke *et al.* (2000). La construcción de circuitos por este medio fue propuesta inicialmente por Seelig *et al.* (2006); más tarde, otros trabajos como el de (Zhang *et al.*, 2007) o el de (Qian y Winfree, 2009) se encargaron de mejorar y simplificar la técnica. El desplazamiento de cadenas mediado por *toehold*s requiere la adición de secuencias de cadena sencilla externas para su funcionamiento; para evitar esto, las reacciones de desplazamiento de cadenas se pueden realizar en cascada. En las reacciones de desplazamiento de cadenas en cascada, la salida de la reacción previa sirve como entrada para la siguiente reacción (Figura 17); con ello se elimina la necesidad de añadir las cadenas

sencillas que desencadenan las reacciones en cada paso, por lo que es posible construir sistemas autónomos complejos.

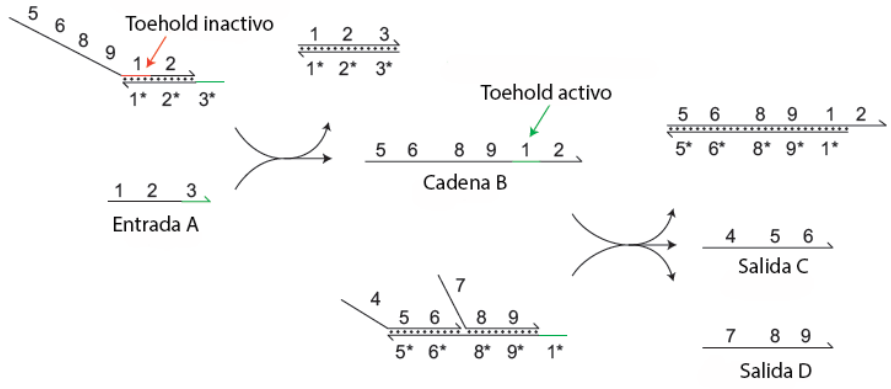


Figura 17: Desplazamiento de cadenas en cascada. La entrada A reacciona con el complejo X mediante el desplazamiento de cadena liberando la cadena B. Una vez que esto sucede, el *toehold* (dominio 1) se activa y como es complementario a 1* inicia una reacción de desplazamiento de cadenas con el complejo Y liberando las dos cadenas de salida C y D. Tomado de la referencia (Zhang y Seelig, 2011).

4.5. Compuertas y circuitos lógicos construidos con ADN

Existen numerosos trabajos que utilizan el ADN como material para construcción de dispositivos lógicos, elementos de circuitos lógicos o circuitos lógicos en sí, como ejemplos se tienen (Modi *et al.*, 2009), (Qian y Winfree, 2009) y (Seelig *et al.*, 2006), sólo por mencionar algunos. En (Modi *et al.*, 2009) se realiza la primera demostración de una compuerta AND basada en ADN dentro de células vivas; los autores mencionan que gracias al desempeño de este dispositivo, se ilustra rotundamente que los nanodispositivos de ADN funcionan dentro de los sistemas y es posible emplearlos para llevar a cabo detecciones más complejas tales como el diagnóstico y terapias específicas en sistemas vivos. En lo que respecta a implementaciones de circuitos con ADN, Qian y Winfree (2009) proponen una arquitectura sencilla para compuertas de ADN que parece adecuada para formar circuitos a gran escala que incluyan varias compuertas. Esta arquitectura hace uso de reacciones de desplazamiento de cadenas de ADN en cascada. En (Seelig *et al.*, 2006) se presenta un diseño modular y la implementación experimental de circuitos lógicos digitales basados en ADN; en ella se demuestra el funcionamiento de compuertas AND, OR y NOT. A partir de los ejemplos anteriores se puede constatar que el ADN es cada vez

es más utilizado como material para la construcción de circuitos, estructuras y motores a nano-escala; muchas de estas construcciones se realizan mediante reacciones de desplazamiento de cadenas, como es el caso del último ejemplo (Seelig *et al.*, 2006), debido a que es un mecanismo sencillo pero robusto.

En el estado del arte se pueden encontrar diversas estrategias para la construcción de dipositivos lógicos con ADN, tal como se observa en la Figura 18, los trabajos existentes se pueden agrupar de acuerdo al tipo de reacción y moléculas utilizadas o según el tipo de circuito, ya sea basado en solución o de hibridación localizada (Gonzalez *et al.*, 2014). Los circuitos basados en solución son aquellos donde las moléculas se dispersan a través de una solución para encontrar otra molécula dentro de ese subconjunto como en el caso de (Qian y Winfree, 2011). Los de hibridación localizada son aquellos donde las moléculas de ADN se encuentran adheridos a una superficie y solamente pueden interactuar con otras moléculas que se encuentren dentro de su alcance, algunos trabajos que corresponden a este tipo son (Genot *et al.*, 2011) y (Chandran *et al.*, 2011). A continuación se enlistan algunas estrategias de diseño de circuitos lógicos a partir de ADN:

- **Enzimas de ADN.** Las *DNAzymes* son moléculas de ADN de cadena simple capaces de realizar alguna reacción química (acción catalítica) al reaccionar con un sustrato adecuado. Las compuertas diseñadas con este enfoque cambian su estructura cuando hibridan con un oligonucleótido complementario para romper un sutrato, el cual posee un fluoróforo y un *quencher* en sus extremos opuestos, de este modo emite fluorescencia e indica un resultado positivo. Stojanovic y Stefanovic (2003) y Macdonald *et al.* (2006) usaron las enzimas de ADN E6 y 8-17 para construir las máquinas MAYA I y MAYA II, respectivamente; en dichos trabajos se proponen las compuertas YES, AND y AND-NOT para implementar el clásico juego del gato. Otro ejemplo se puede encontrar en (Kahan-Hanum *et al.*, 2013), donde a partir de la enzima Dz13 se diseñan las compuertas AND, NOT, OR, NAND, AND-NOT, XOR, NOR y una AND de tres entradas.
- **Deoxy-oligonucleótidos.** Son oligonucleótidos compuestos por deoxynucleótidos, esto es, cualquier nucleótido que contenga una deoxy azúcar. En (Prokup *et al.*, 2012) se utiliza este enfoque y se propone la construcción de una compuerta AND

basada en ADN controlada de manera fotoquímica.

- **Enzimas.** Los dispositivos construidos por medio de este enfoque presentan la forma de una máquina de Turing, un ejemplo de esto es el trabajo de Benenson *et al.* (2001), el cual se discutió en la sección 4.3.3. Este método requiere algunas moléculas de transición y de la enzima *FokI*.
- **Autoensamblado de ADN.** Rothemund *et al.* (2004) propone el uso de este enfoque para realizar la operación XOR e implementar un autómata que genera un fractal llamado “triángulo de Sierpinski”.

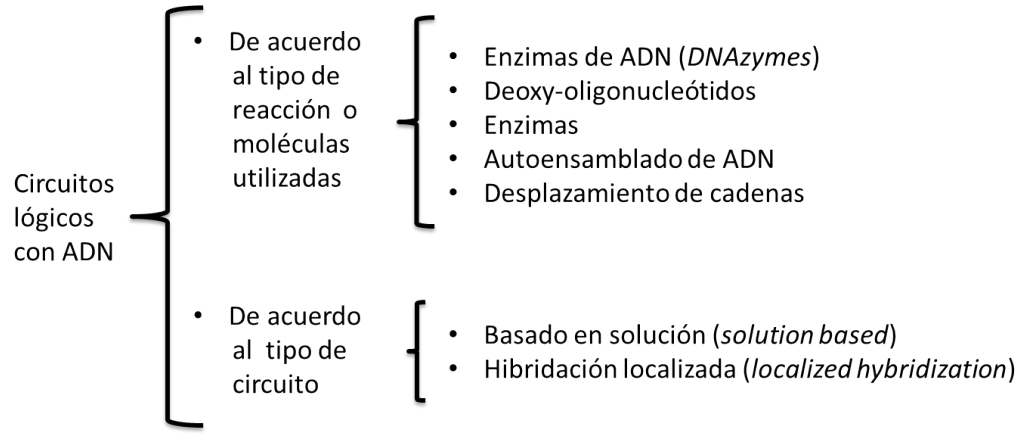


Figura 18: Estrategias de diseño de circuitos construidos con ADN.

4.5.1. Compuertas y circuitos lógicos construidos con el modelo de desplazamiento de cadenas de ADN

Recientemente, se han utilizado las reacciones de desplazamiento de cadenas de ADN para construir numerosos dispositivos, incluyendo circuitos, amplificadores catalíticos, motores moleculares autónomos y nanoestructuras reconfigurables (Zhang y Seelig, 2011). Un ejemplo de esto se puede ver en la publicación de Seelig *et al.* (2006) en Science, en la cual se presenta un diseño modular y la implementación experimental de circuitos lógicos digitales basados en ADN; en ella se demuestra el funcionamiento de compuertas AND, OR y NOT, restauración de señales, amplificación, regeneración y conexiones en cascada. Las compuertas utilizan ácidos nucleicos de cadena sencilla como entradas y salidas, el mecanismo se basa exclusivamente en el reconocimiento de la secuencia y en el desplazamiento de cadenas. El mayor inconveniente de este trabajo es

la enorme cantidad de tiempo que se requiere para generar resultados, ya que el circuito con restauración de señal tarda por lo menos 10 horas para activarse. Sería conveniente optimizar la respuesta de las compuertas lógicas individualmente para mejorar el desempeño de circuito.

A partir del 2009 se desarrolla la herramienta Visual DSD por el Biological Computation group de Microsoft Research y otros colaboradores. Ésta es una aplicación del lenguaje de programación DSD y del compilador descrito en (Phillips y Cardelli, 2009), la cual se revisó en el Capítulo 3. Existen varias publicaciones donde investigadores pertenecientes a dicho grupo aprovechan los recursos de Visual DSD para realizar aportes novedosos al área en cuestión. Por ejemplo, Chandran *et al.* (2011) presentan circuitos libres de enzimas que actúan mediante reacciones de cascadas de hibridación de ADN; en (Lakin y Phillips, 2011) se demuestra cómo se puede utilizar Visual DSD para diseñar una máquina de pila de ADN y analizar su comportamiento; en (Lakin *et al.*, 2012b) se propone una máquina abstracta genérica que puede ser instanciada para simular una serie de cálculos usando una gama de métodos de simulación; en (Lakin *et al.*, 2012a) se demuestra, por primera vez, el uso de técnicas de verificación probabilística para analizar la exactitud, fiabilidad y rendimiento de dispositivos de ADN durante la fase de diseño.

En (Zhang *et al.*, 2010) se presenta un enfoque novedoso, el cual utiliza ADN circular (moléculas de ADN covalentemente cerradas que se encuentran en bacterias, virus, mitocondrias y plásmidos) para construir compuertas lógicas. Aprovechando la estructura circular del ADN, la mayor parte de las regiones de reconocimiento específico fueron diseñadas en un solo plásmido de ADN. Los autores reportaron que dependiendo de la secuencia de reconocimiento y de reacciones de desplazamiento que sean altamente eficaces, las compuertas lógicas proporcionan resultados correctos. Sin embargo, aunque la mayoría de los resultados de las compuertas son correctos, el modelo presenta algunas desventajas como lo son fugas o interferencias en las compuertas y algunos problemas con respecto a uno de los materiales que utilizan, por ejemplo, las nanopartículas de oro que forman clústers.

En (Chiniforooshan *et al.*, 2011) se presenta un diseño teórico cuyos autores conjeturan como funcional para implementar cualquier circuito booleano usando el mecanismo

de desplazamiento de cadenas de ADN, pero puede existir diafonía entre los buffers si no se tiene cuidado con el diseño de los *toeholds*, que deben ser diferentes para los *linkers* y *buffers*, lo que hace que se requieran más secuencias en el sistema. Por otro lado, Qian y Winfree (2011) publican en Science la construcción de un circuito que calcula el piso de la raíz cuadrada de un número binario de cuatro bits; cabe mencionar que éste no es un circuito lógico digital optimizado y básicamente está diseñado para exhibir las compuertas AND, OR, NOT, NAND, NOR que lo forman, así como el *fan-in / fan-out* de las compuertas lógicas y el *fan-out* de las señales de entrada. La construcción del circuito se llevó a cabo con un mecanismo de reacción simple basado en el proceso de desplazamiento de cadenas reversible. Su diseño incorpora otros elementos cruciales para el desarrollo de circuitos a gran escala, tales como herramientas generales de depuración, la preparación de circuito en paralelo, y la utilización de un compilador de circuitos automatizado; particularmente, se maneja la aplicación Visual DSD para realizar las pruebas correspondientes del modelo. Durante el mismo año, Quian y Winfree hacen otra contribución al área utilizando reacciones de desplazamiento de cadenas de ADN en cascada y bloques de construcción elementales para la construcción de los circuitos (Qian y Winfree, 2009). Cada bloque consiste en una compuerta sencilla que está configurada de tal manera que una entrada convierta catalíticamente el combustible en una salida si la concentración de la entrada excede el nivel de umbral. Las señales de entrada, salida y el combustible son cadenas sencillas de ADN mientras que el nodo de la compuerta corresponde a moléculas parcialmente dobles de ADN. A pesar de que se sugiere que la arquitectura presentada puede ser escalable para circuitos de miles de compuertas, son sólo suposiciones, ya que uno de los inconvenientes es que al construir circuitos más complicados con este enfoque sería necesario generar más secuencias diferentes y con ello se requeriría agregar un procedimiento de purificación, además de que los circuitos serían bastante lentos y no garantizarían resultados correctos.

En (Li *et al.*, 2013) se tiene el diseño y la construcción experimental de una compuerta lógica de mayoría de tres entradas mediante el mecanismo de desplazamiento de cadenas. Una de las características clave de una compuerta de mayoría de tres de entradas, es que las tres entradas tienen la misma prioridad, y la salida será verdadera si por lo menos dos entradas son verdaderas. El diseño de la compuerta consiste en una cadena

central de ADN circular con tres dominios únicos entre los cuales secuencias idénticas se empalman. Además, en este trabajo se presenta una compuerta lógica compleja de cinco entradas en base a la compuerta de mayoría que se describió con anterioridad. Mediante el control de dos de las cinco entradas de la compuerta compleja se puede realizar cualquier combinación de compuertas OR y AND de las otras tres entradas. Según lo que afirman los autores, es posible utilizar varias compuertas de mayoría para construir circuitos más complejos sin necesidad de alterar la longitud de las secuencias originales de las compuertas y además, sugieren que es posible combinarlas con otras compuertas lógicas existentes debido a la naturaleza de su diseño. Sin embargo, no mencionan un ejemplo claro de cómo se podría realizar exactamente.

4.6. Herramientas de simulación para modelos con desplazamiento de cadenas de ADN

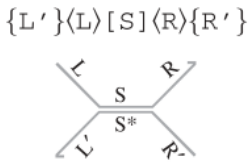
4.6.1. Lenguaje DSD

El lenguaje de programación para desplazamiento de cadenas de ADN (DSD) se desarrolló para facilitar el diseño, simulación y análisis de mecanismos formados a partir del modelo de desaplazamiento de cadenas (Phillips y Cardelli, 2009). Este lenguaje provee una sintaxis textual para expresar la estructura de especies de ADN como las que se muestran gráficamente en la Figura 19. La semántica del lenguaje traduce formalmente una colección de especies de ADN en un sistema de reacciones químicas mostrando todas las posibles interacciones entre dichas especies. También incluye abreviaciones gráficas y sintácticas que permiten representar una clase de moléculas de ADN en particular. Esta clase de moléculas no contiene estructuras secundarias, es decir, del complejo principal de doble cadena solamente pueden quedar colgando secuencias de ADN de cadena simple.

A continuación se muestran ejemplos básicos de la sintaxis textual del lenguaje y la representación gráfica correspondiente de la molécula; dicha sintaxis se define en términos de las secuencias S, L, R, la hebra A, compuerta G_i y el sistema D_j. Una secuencia S está compuesta por uno o varios dominios, los cuales pueden ser dominios largos N o dominios cortos $N^\wedge$ (*toeholds*). Una cadena sencilla superior de ADN $< S >$ denota una

secuencia S con orientación de izquierda a derecha ($5' - 3'$), mientras que la cadena $\{S\}$ representa una cadena sencilla inferior orientada de derecha a izquierda ($3' - 5'$):

Una cadena doble $[S]$ denota la cadena superior $< S >$ unida a su cadena inferior complementaria $\{S^*\}$. Una compuerta G está compuesta de segmentos de cadenas dobles de ADN de la forma $\{L^{'}\} < L > [S] < R > \{R^{'}\}$, lo que es lo mismo que una cadena superior $< L \quad S \quad R >$ (llamémosle G1) unida por una región de doble cadena $[S]$ a una cadena inferior $\{L^{'} \quad S^* \quad R^{'}\}$ (llamémosle G2), entonces esto se escribe $G1 : G2$. En el caso de que se unan por una cadena superior en común se denota por $G1 : G2$. Con lo anterior básicamente se obtiene un complejo de doble cadena $[S]$ con las cadenas sencillas $< L^{'} >$ y $< R^{'} >$ sobresalientes:



Una especie de ADN puede ser una cadena superior $< S >$, una cadena inferior $\{S\}$ o una compuerta G. Un sistema con tales especies puede ser representado por la variable D. La barra $'|'$ se utiliza para representar múltiples sistemas en paralelo, por ejemplo $D1 \mid D2$. Un dominio N puede restringirse a un sólo tipo de molécula D, esto se escribe $new\, N\, D$, lo que significa que el dominio N y su complemento no puede ser utilizado por ninguna otra molécula que no pertenezca a D. El lenguaje DSD también permite definiciones en módulos de la forma $X(\tilde{n}) = D$, donde $\tilde{n}$ representa los parámetros del módulo, por lo que si se quiere representar una instancia del módulo D con parámetros $\tilde{m}$ entonces se escribe $X(\tilde{m})$. Al inicio del programa se definen una serie de módulos fijos, los cuales no deben ser recursivos, esto es que los módulos no pueden invocarse a sí mismos de manera directa o indirecta por medio de otro módulo. El lenguaje DSD se utiliza por medio de la herramienta de software Visual DSD (Lakin *et al.*, 2012a).

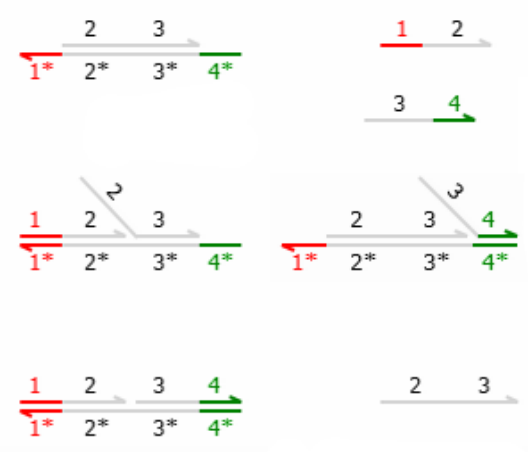


Figura 19: Representación gráfica de moléculas de ADN con el software Microsoft Visual DSD.

4.6.2. Visual DSD

Visual DSD es una implementación del lenguaje de programación DSD para diseñar, simular y analizar mecanismos a partir del desplazamiento de cadenas de ADN (Phillips y Cardelli, 2009). Como se mencionó en el apartado anterior, el lenguaje incluye elementos básicos de dominios de secuencias, *toeholds* y *branch migration* y se supone que las cadenas no poseen estructuras secundarias. La herramienta compila una colección de moléculas de ADN en un conjunto de reacciones químicas. También incluye un simulador estocástico que calcula las posibles trayectorias del sistema y muestra gráficos de las poblaciones de especies a través del tiempo; asimismo, cuenta con un simulador determinístico que construye y resuelve ecuaciones diferenciales ordinarias que representan la dinámica del sistema. Por otra parte, el espacio de estados alcanzables del sistema se puede construir mediante una cadena de Markov en tiempo continuo. Además, permite que los dominios se compilen como una secuencia de nucleótidos, con el fin de implementar los sistemas en laboratorio; también es posible desplegar las secuencias de los dominios en el modo de visualización gráfica de las moléculas para fines de demostración y análisis.

La herramienta de software Microsoft Visual DSD está disponible de forma gratuita en su versión web que puede encontrarse en *http://lepton.research.microsoft.com/webdna* o en su versión descargable que puede obtenerse de la misma liga. Existe la opción de utilizarse como una aplicación gráfica o mediante línea de comandos. En la Figura 20 se

muestra una captura de pantalla del entorno gráfico de la herramienta Visual DSD.

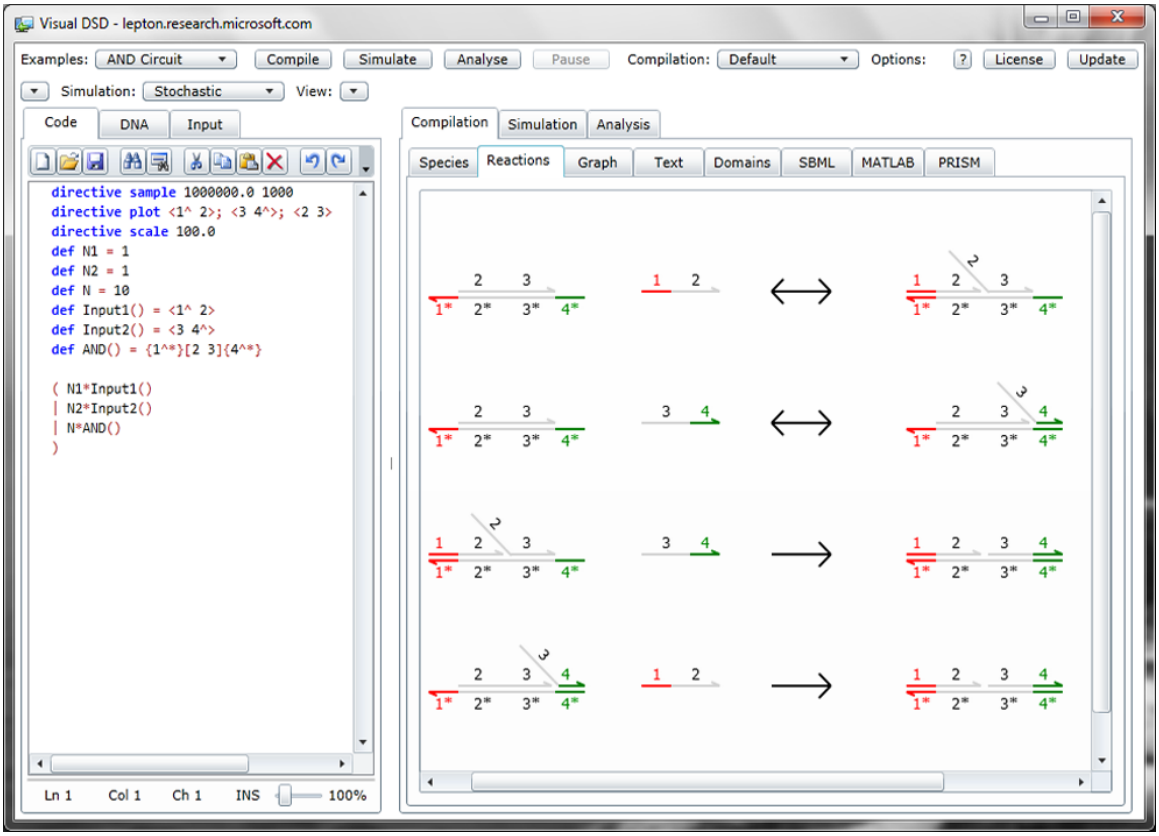


Figura 20: Captura de pantalla del software Microsoft Visual DSD. La sección izquierda muestra el código y la sección derecha la representación gráfica de las moléculas de ADN correspondientes.

Visual DSD ofrece la opción de escoger entre cuatro modelos semánticos (infinito, default, finito y detallado) para realizar la compilación; estos modelos especifican las posibles reacciones entre las moléculas de ADN. El número de reacciones en el sistema y el costo computacional de la compilación y simulación se incrementan conforme se avanza en los cuatro niveles de los modelos de compilación. El software cuenta con tres diferentes algoritmos de simulación a elegir. El simulador estocástico utiliza el algoritmo de Gillespie para generar una posible trayectoria del sistema a través del tiempo. El simulador determinístico funciona con un solucionador de ecuaciones diferenciales ordinarias con el método de Runge-Kutta-Felhberg (Fehlberg, 1969) para producir una gráfica determinista de las concentraciones y sus evoluciones a lo largo del tiempo. Para ambas simulaciones se supone que la propensidad de la reacción es proporcional al producto de los reactivos. El otro simulador es el JIT (*just-in-time*) (Lakin *et al.*, 2012b), esta opción afecta tanto la simulación como la compilación de las especies de ADN en reacciones

químicas. En particular, este algoritmo es útil para sistemas muy grandes donde existan miles de posibles reacciones y el proceso de compilación pueda tornarse muy largo. Funciona compilando dinámicamente nuevas reacciones durante el proceso de simulación; después de cada reacción, el algoritmo verifica que los productos de dicha reacción no sean repetidos, si no, lleva a cabo un solo paso de compilación donde se argumenta al sistema con las nuevas especies y nuevas reacciones. Ejecutar un mismo programa con el algoritmo JIT no siempre genera los mismos resultados ya que éstos dependen de las especies que se produzcan en cada ejecución y pueden no ser los mismos debido al procedimiento.

Los resultados de la simulación se pueden visualizar en forma de texto, mediante una representación tabular de los datos de las especies de ADN a través del tiempo o mediante una gráfica de las concentraciones de las especies a través del tiempo. Adicionalmente, es posible calcular el grafo de todos los posibles estados alcanzables del sistema a partir de las moléculas iniciales. Mediante esta opción, la herramienta calcula la cadena de Markov en tiempo continuo y muestra toda la información relacionada a los estados del sistema. Como se muestra en el grafo de estados de la Figura 21, cada estado alcanzable se representa por un nodo y cada posible reacción por una arista; el estado inicial se remarca con un borde negro mientras que los estados finales se indican con un borde de color rojo. Otra ventaja que presenta es que los datos de simulación se pueden exportar en diferentes formatos para ser analizados por otras herramientas, por ejemplo, los datos de las tablas o gráficas de las simulaciones pueden guardarse en formato CSV o TSV para después cargarlos en una hoja de cálculo; las redes de reacciones químicas pueden exportarse a SBML, siglas en inglés de *Systems Biology Markup Language*, el cual es un lenguaje de descripción basado en XML utilizado para representar modelos de procesos biológicos como redes metabólicas, rutas de señalización celular y redes de regulación génicas, entre otros. Las cadenas de Markov en tiempo continuo se pueden exportar como modelo para PRISM, lo que permite al usuario expresar las propiedades del sistema como fórmulas de lógica temporal y verificarlos mediante la comprobación de modelos estocásticos.

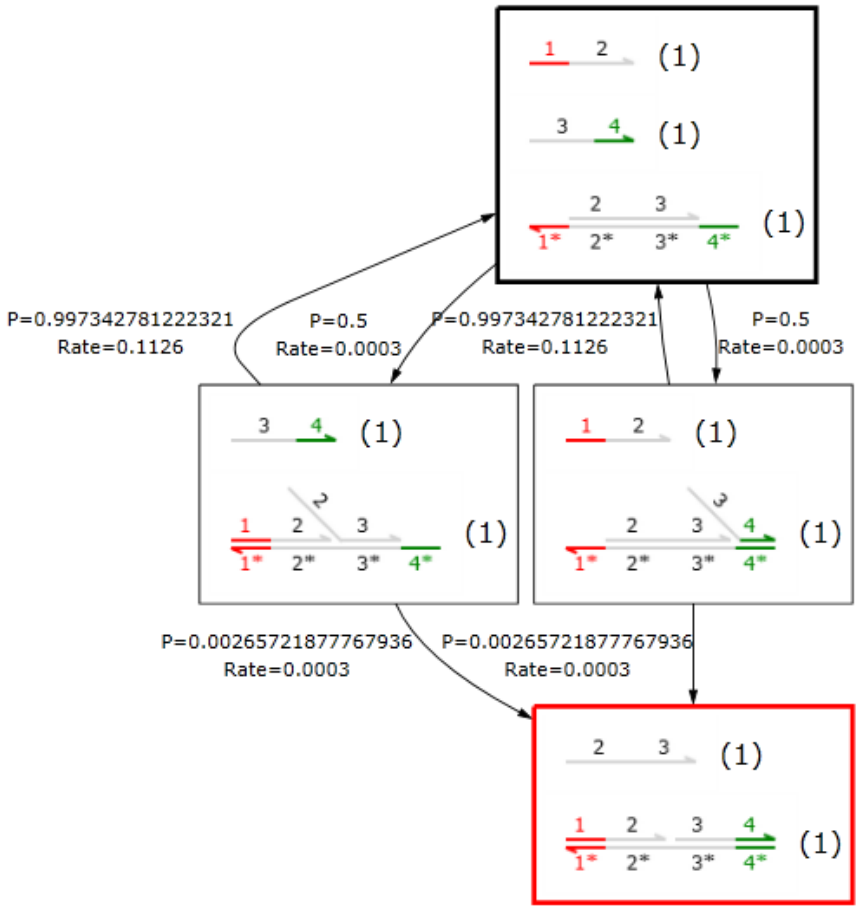


Figura 21: Ejemplo de grafo de estados de un sistema, captura de pantalla del software Microsoft Visual DSD.

4.6.3. PRISM (*Probabilistic Model Checker*)

El *model checking* probabilista es una técnica formal de verificación automática para analizar sistemas que presentan un comportamiento estocástico. Al igual que el *model checking* tradicional, esta técnica involucra la construcción de un modelo de estados finitos para un sistema real a partir de algún formalismo de alto nivel con la particularidad de incluir información acerca de la probabilidad y distribución en el tiempo de las transiciones entre los estados. Es importante porque a partir de este modelo es posible calcular automáticamente una gran cantidad de medidas cuantitativas del sistema original. PRISM es una herramienta de software que utiliza dicho enfoque y ofrece soporte para tres tipos de modelos probabilísticos: cadenas de Markov en tiempo discreto, procesos de decisión de Markov y cadenas de Markov en tiempo continuo. Es una herramienta gratuita y de

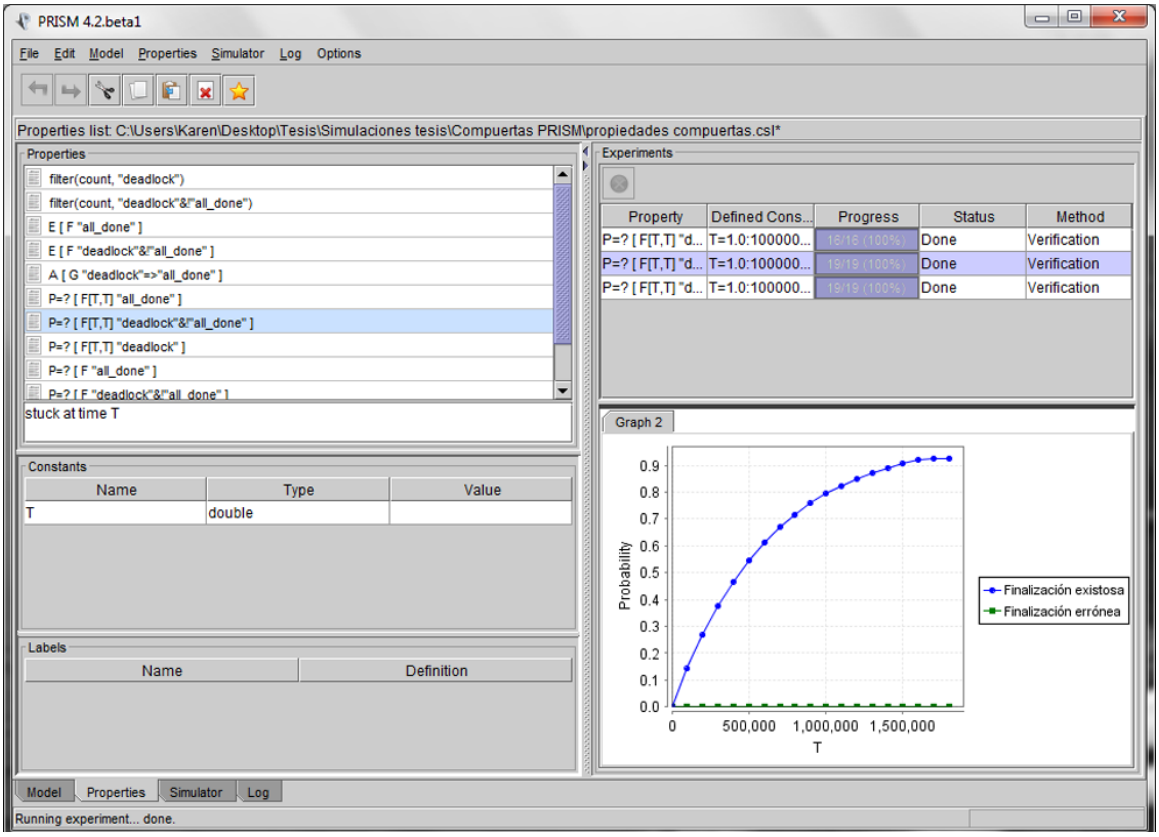


Figura 22: Captura de pantalla del software PRISM.

código abierto que se puede obtener de *www.cs.bham.ac.uk/∼dxp/prism*, funciona para la mayoría de los sistemas operativos y está disponible en su versión con interfaz gráfica o puede utilizarse mediante la línea de comandos. En la Figura 22 se muestra una captura de pantalla del software.

La herramienta también permite que los modelos se argumenten con *costos* y *recompensas*, asignando valores reales a los estados y transiciones del modelo. Esto permite el razonamiento de más medidas cuantitativas del sistema como los son el tiempo de finalización y el consumo de energía, entre otras. Los modelos se expresan mediante el lenguaje de modelado PRISM, el cual es un lenguaje basado en estados y en el formalismo de *Reactive Modules* (Alur y Henzinger, 1999). Los sistemas se describen como la composición paralela de un conjunto de módulos. Cada estado del módulo está dado por un conjunto de variables con valores finitos y su comportamiento está dado por un conjunto de comandos probabilísticos. El lenguaje admite variables globales, sincronización y varios procesos de operaciones algebraicas. Algunas de las funciones que ofrece

PRISM son la generación de series de resultados cuantitativos y su graficación para una mejor visualización.

PRISM incluye varias técnicas de análisis de modelos, entre ellas métodos cualitativos como algoritmos basados en grafos, cuantitativos para el cáculo numérico de probabilidades y valores esperados de costos y recompesas. Este software también ofrece soporte para cálculos numéricos aproximados usando métodos de Monte-Carlo y simulaciones de eventos discretos (Kwiatkowska *et al.*, 2011).

4.6.4. NUPACK (*Nucleic acid Package*)

La herramienta NUPACK (Zadeh *et al.*, 2011) es una aplicación web (*www.nupack.org*) que permite analizar y diseñar sistemas basados en ácidos nucleicos. Los algoritmos que utiliza se formulan en términos de estructuras secundarias de los ácidos nucléicos, en la mayoría de los casos los pseudo-lazos se excluyen del ensamble de las estructuras.

NUPACK se compone de tres módulos principales:

- Análisis: permite el análisis termodinámico de soluciones diluidas de cadenas de nucleótidos que interactúan entre sí.
- Diseño: posibilita el diseño de secuencias de multi-estado o de un solo estado para cadenas de nucleótidos que interactúan entre sí.
- Utilidades: admite el despliegue, evaluación y anotación de las propiedades de equilibrio de un complejo de cadenas de ácidos nucléicos.

Capítulo 5. Diseño conceptual de compuertas lógicas

Tal como se mencionó en capítulos anteriores, los circuitos a nivel molecular tienen una amplia gama de aplicaciones sobre todo en áreas como nanotecnología, biotecnología y medicina. Particularmente, el desplazamiento de cadenas de ADN es un enfoque ideal para el diseño de este tipo de estructuras ya que se basa únicamente en la hibridación de secuencias de nucleótidos complementarias para realizar los pasos de cálculo. Cabe mencionar que una de las principales dificultades para diseñar mecanismos correctos y robustos mediante este enfoque radica en la posible existencia de reacciones indeseadas entre las moléculas del sistema o interferencias. En este capítulo se presenta el diseño de las estructuras propuestas para cada una de las compuertas lógicas y su respectiva descripción.

5.1. Convenciones de diseño

Es necesario tomar en cuenta ciertas convenciones de diseño para que el modelo propuesto funcione de la manera esperada:

- Los *toeholds* o puntos de apoyo (dominios cortos) tienen una longitud de 4 a 10 nucleótidos y los dominios largos una longitud de 20 o más nucleótidos, esto asegura que los *toeholds* desencadenen las reacciones de desplazamiento y la dinámica de la reacción dependa de los dominios largos; también le confiere estabilidad a los complejos, evitando que las cadenas se separen o deshibriden.

- Las variables (tanto entradas como salidas) siempre se representan como cadenas sencillas de ADN con orientación $5' - 3'$ y deben cumplir con las características que se mencionan en el apartado correspondiente.

- Las hebras de protección son cadenas sencillas de ADN que permiten que las reacciones sean iniciadas únicamente por los *toeholds* siguiendo así la definición de reacción de desplazamiento de cadenas, algunas de estas hebras reaccionan con los complejos principales o con los complejos auxiliares para realizar operaciones; aquellas que no son útiles para este fin se les llama desperdicio.

- Se manejan dos tipos de *toeholds*: los universales y los específicos. Los primeros se utilizan en las variables y en los complejos de las compuertas, mientras que los específicos se requieren para realizar operaciones únicamente entre los complejos compuerta y los complejos auxiliares correspondientes; esto evita la ocurrencia de reacciones indeseadas.

- Las secuencias complementarias se representan con la misma letra y un asterisco, por ejemplo, la secuencia complementaria al dominio A es A^*.

- Cada compuerta lógica se compone de un complejo principal y de un complejo auxiliar.

- Los complejos principales de las compuertas son cadenas sencillas de ADN prehibridadas parcialmente complementarias que reaccionan con las variables de entrada para realizar las operaciones correspondientes de cada compuerta.

- Los complejos auxiliares son cadenas sencillas de ADN prehibridadas que reaccionan con las hebras de protección de sus respectivos complejos compuerta para liberar o atrapar la variable de salida.

- El complejo detector se utiliza para identificar la salida final del sistema emitiendo fluorescencia si la salida equivale a un '1' lógico.

5.2. Representación de variables

Como se explicó en el Capítulo 3, las compuertas lógicas responden de forma predecible a los valores de voltajes de entrada que se encuentran dentro de los intervalos definidos 0 y 1 (cuando no hay voltaje y cuando lo hay, respectivamente). En el diseño propuesto, las variables de entrada y salida se representan como cadenas sencillas de ADN y su valor booleano está dado por la ausencia ('0' lógico) o presencia de éstas ('1' lógico). Las entradas y salidas presentan el mismo formato de representación, de modo que una variable de salida puede servir como variable de entrada para otra compuerta; esto se requiere para llevar a cabo reacciones en cascada entre varias compuertas y de esta manera posibilitar la construcción de circuitos más robustos. En la Figura 23 se puede apreciar la representación esquemática de una variable (variable A), la cual consiste

en una hebra sencilla de ADN con orientación $5' - 3'$, compuesta de dos *toeholds* universales (uno en cada extremo), una secuencia específica (dominio largo) y un dominio representativo (dominio largo). Éstos dos últimos deben ser distintos para cada una de las variables en el sistema ya que esto permite su diferenciación.

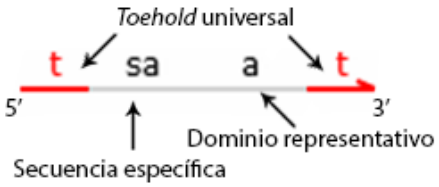


Figura 23: Representación de una variable (variable A).

5.3. Detección de resultados

El valor de una variable de salida en una compuerta lógica es booleano, es decir, un '1' ó '0' lógicos que en este diseño se representan por la presencia o ausencia, respectivamente, de la cadena correspondiente a la variable esperada. Para monitorear dicha variable se propone un complejo detector, el cual se puede apreciar en la Figura 24. Este complejo está formado a partir de dos moléculas ssADN parcialmente complementarias. La cadena inferior es completamente complementaria a la variable de salida esperada (en este caso Y) con un *quencher* o inhibidor de fluorescencia en el extremo $5'$ y expone un *toehold* universal para iniciar la reacción de hibridación con la variable de salida. La cadena superior corresponde a la hebra de protección con un fluoróforo en el extremo $3'$ de tal forma que, cuando estas cadenas se encuentran prehibridadas no existe fluorescencia y cuando la variable esperada (variable Y) hibrida con la cadena inferior complementaria desplazando la hebra de protección, ésta última emite fluorescencia.

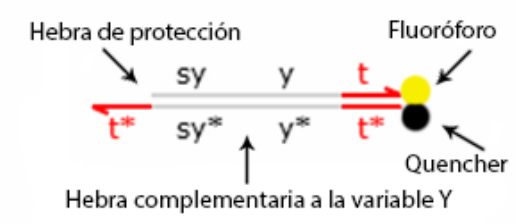


Figura 24: Complejo detector.

5.4. Modelos de compuertas lógicas

El diseño de una compuerta lógica está compuesto al menos por dos complejos de ADN: el complejo principal y un complejo auxiliar; éste último se requiere para realizar las operaciones necesarias para liberar la variable de salida. La variable de salida puede servir como entrada para otra compuerta, permitiendo la ocurrencia de reacciones en cascada entre dos o más compuertas para generar circuitos más complejos. Para mantener la relación con las tablas de verdad de las compuertas del Capítulo 3, las entradas corresponden a las variables A y B, la variable de salida corresponde a Y. A continuación se presentan los modelos de cada una de las compuertas propuestas y sus esquemas de reacción.

5.4.1. Compuerta AND

El diseño propuesto para esta compuerta se muestra en la Figura 25 y lo conforman dos complejos: el complejo principal de la compuerta AND y un complejo auxiliar. El complejo principal (Figura 25(a)) se compone de dos moléculas ssADN parcialmente complementarias prehibridadas. La cadena inferior es completamente complementaria a las variables de entrada, donde la mitad izquierda es complementaria a la primera variable de entrada (A) y la mitad derecha a la segunda (B); la cadena superior corresponde a la hebra de protección, la cual es completamente complementaria a la cadena inferior ($3' - 5'$) del complejo auxiliar y posee un *toehold* específico (dominio te) para iniciar la reacción de desplazamiento en el complejo auxiliar. Al complejo auxiliar (Figura 25(b)) lo conforman tres ssADN prehibridadas: la cadena superior que codifica la variable de salida (Y), una cadena superior (desperdicio) que funciona como hebra de protección para que la reacción de desplazamiento se inicie por medio del *toehold* específico de la cadena inferior (te^*) y la cadena inferior. Esta última es completamente complementaria a la hebra de protección de la compuerta AND. A continuación, se describen de manera esquemática las reacciones que ocurren en cada uno de los casos de la tabla de verdad:

- Caso 1: No hay entradas (A = 0, B = 0). Para el primer caso de la tabla de verdad de esta compuerta, cuando ambas variables tienen un valor de ‘0’ lógico, la salida también debe ser un ‘0’ lógico. En este diseño equivale a que ninguna de las dos

variables (A y B) estén presentes, por lo que no ocurrirá reacción alguna entre los complejos y por lo tanto no se liberará la variable de salida (Y), lo que indica un '0' lógico como salida, concordando con la tabla de verdad.

- Casos 2 y 3: Sólo una entrada (A = 0, B = 1) y (A = 1, B = 0). En caso de que sólo esté presente alguna de las dos entradas, la salida debe ser un '0' lógico. En la Figura 26 se pueden apreciar los pasos de la resolución esquemática de la compuerta si se tiene presente solamente la variable A ('1' lógico). El primer paso consiste en la hibridación de los *toeholds* universales complementarios (dominio t*) de la variable A con la hebra inferior del complejo compuerta. En el segundo paso, la variable A desplaza parcialmente la hebra de protección del complejo principal. Dado que la hebra de protección del complejo AND no se libera, ésta no puede reaccionar con el complejo auxiliar encargado de liberar la variable Y, por lo que la salida resulta en un '0' lógico ya que la variable Y no es liberada. De manera análoga, cuando solamente está presente la variable B, sucede algo similar, aunque en este caso la variable B hibrida con su sección complementaria en la hebra inferior del complejo principal sin desplazar completamente la hebra de protección, por lo que no reacciona con el complejo auxiliar y no libera la variable Y, lo que nos indica un '0' lógico como salida.

- Caso 4: Dos entradas (A = 1, B = 1). El único caso donde la compuerta AND regresa un '1' lógico es cuando ambas entradas tienen un valor de '1' lógico. En la Figura 27 se puede observar la resolución esquemática para este caso, donde como primer paso una de las dos variables (suponiendo que es la variable A) hibrida con su sección correspondiente en la hebra inferior del complejo principal de la compuerta por medio de los *toeholds* universales (dominio t). Como segundo paso, la hebra de protección se desplaza parcialmente y la variable B hibrida por medio de los *toeholds* universales con su otra sección correspondiente en la hebra inferior del complejo principal, lo que resulta en el desplazamiento total de la hebra de protección de este complejo como tercer paso. Esta última comienza el proceso de hibridación con el complejo auxiliar por medio de los *toeholds* específicos complementarios (dominios te*). El cuarto y último paso consiste en la hibridación total de la hebra de protección del complejo principal con la hebra inferior del complejo auxiliar (ya que son com-

pletamente complementarias) y el desplazamiento de la cadena correspondiente a la variable Y, dando como resultado un '1' lógico puesto que dicha variable ahora está presente.

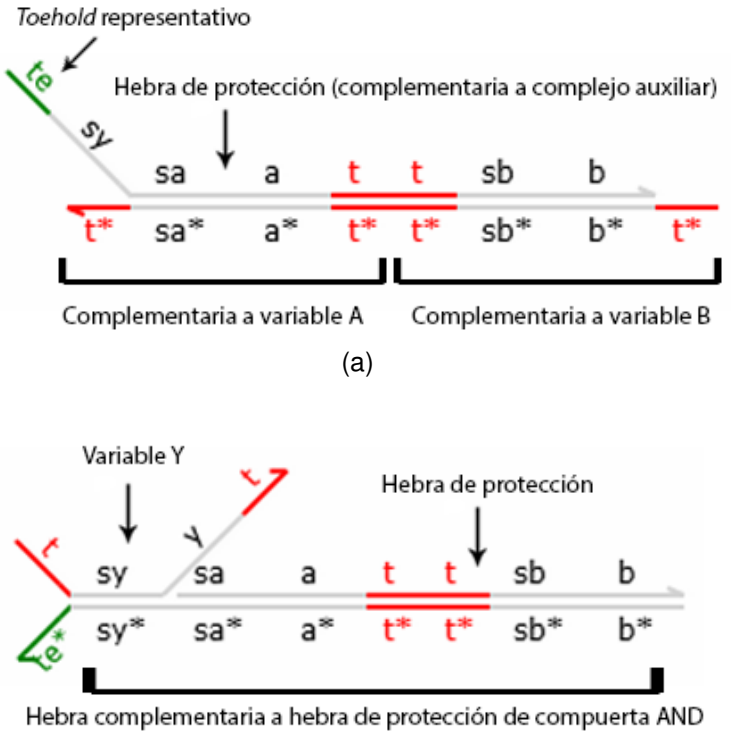


Figura 25: Diseño de la compuerta AND: (a) complejo principal y (b) complejo auxiliar.

5.4.2. Compuerta OR

El diseño que se propone para esta compuerta se muestra en la Figura 28 y lo conforman dos complejos: el complejo correspondiente a la compuerta OR (principal) y un complejo auxiliar. El complejo principal (Figura 28(a)) se compone de cuatro moléculas ssADN parcialmente complementarias prehibridadas. La cadena inferior es completamente complementaria a las variables de entrada, donde la mitad izquierda es complementaria a la primera variable de entrada (A) y la mitad derecha a la segunda (B); las cadenas superiores corresponden a las hebras de protección, siendo la primera y la última desperdicios, mientras que la de en medio es la hebra de protección destinada a reaccionar con el complejo auxiliar por medio del *toehold* específico. El complejo auxiliar (Figura 28(b)) está formado por tres moléculas ssADN prehibridadas: la cadena inferior es completamente complementaria a la hebra de protección de la compuerta OR y las cadenas

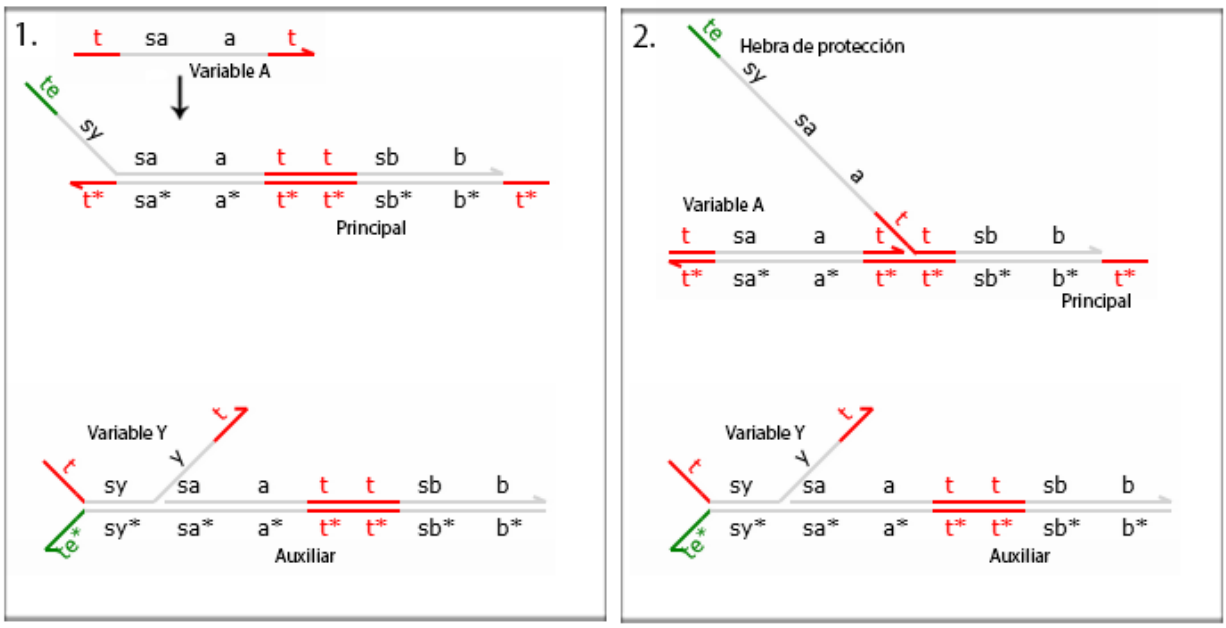


Figura 26: Resolución esquemática de la compuerta AND con una sola entrada (variable A).

superiores son la variable Y y una hebra de protección desperdicio. A continuación, se describen de manera esquemática las reacciones que ocurren en cada uno de los casos de la tabla de verdad:

- Caso 1: No hay entradas (A = 0, B = 0). El único caso donde la compuerta OR regresa una salida de '0' lógico es cuando no están presentes ninguna de las variables o tienen valor de '0'. Cuando no están presentes las variables de entrada, los complejos no reaccionan, por lo que no se libera la variable Y, resultando en un '0' lógico tal como lo indica la tabla de verdad.

- Casos 2 y 3: Sólo una entrada (A = 0, B = 1) y (A = 1, B = 0). En caso de que solamente esté presente alguna de las dos variables de entrada, ya sea A ó B, la compuerta da como resultado un '1' lógico, es decir, debe liberar la variable Y. La Figura 29 representa esquemáticamente las reacciones que ocurren cuando solamente está presente la variable A. Como primer paso, la variable A comienza el proceso de hibridación a partir del *toehold* universal (dominios t) con su sección correspondiente en la cadena inferior del complejo principal, desplazando completamente la hebra de protección desperdicio ($5' - 3'$) y parcialmente la hebra de protección

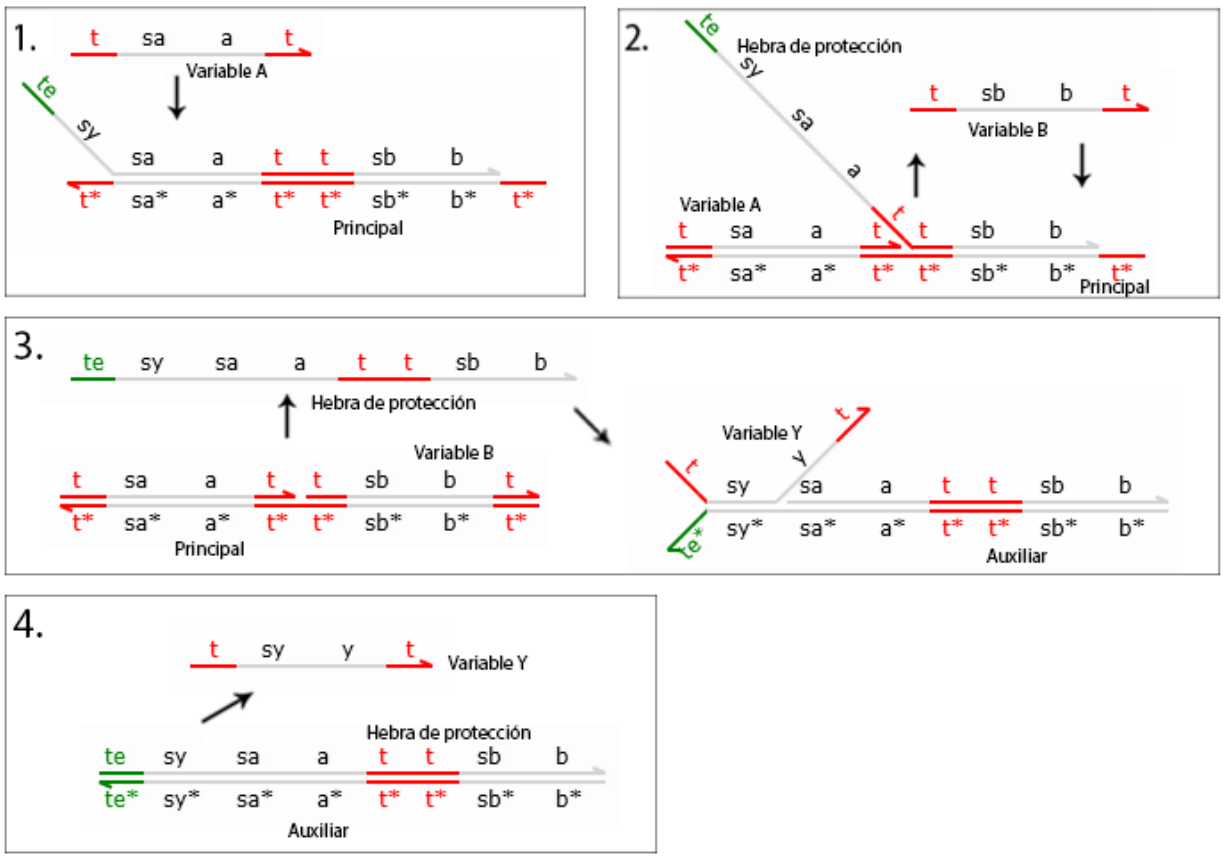
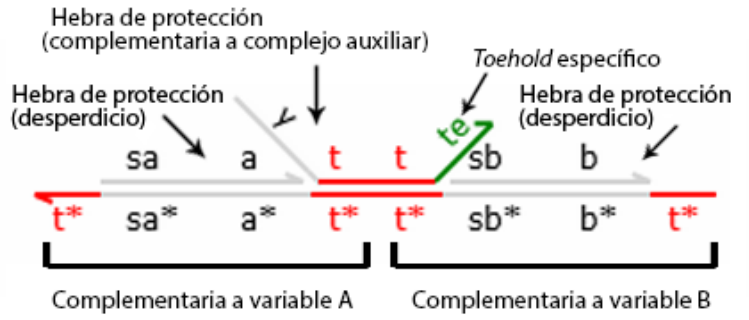


Figura 27: Resolución esquemática de la compuerta AND con dos entradas (variables A y B).

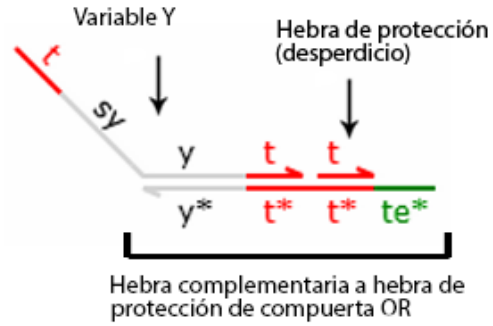
del complejo principal como segundo paso. En el tercer paso, la hebra de protección se separa completamente ya que los enlaces del *toehold* no son lo suficientemente fuertes para mantenerla hibridada y comienza el proceso de hibridación con la cadena inferior del complejo auxiliar (ya que son completamente complementarias) por acción de los *toeholds* específicos (dominios te*) como cuarto paso. Finalmente, la hebra de protección de la compuerta OR desplaza completamente tanto la hebra de protección desperdicio del complejo auxiliar como la cadena correspondiente a la variable Y, lo que produce como resultado un '1' lógico. De manera análoga, cuando sólo está presente la variable B sucede exactamente el mismo proceso.

- Caso 4: Dos entradas (A = 1, B = 1). Cuando ambas variables están presentes (A y B), la compuerta OR también regresa un '1' lógico por lo que debe liberar la variable Y. En la Figura 30 se puede observar la resolución esquemática de este caso, donde el seguimiento de los pasos es igual al caso donde sólo está presente una de las

variables, salvo que al estar presente también la segunda, ésta última hibrida con su parte complementaria en el complejo principal liberando la hebra de protección desperdicio (esto se puede observar en los pasos 2 y 3), lo que prácticamente no afecta al funcionamiento presentado anteriormente, ya que basta con que alguna variable esté presente en el sistema para que la compuerta OR regrese un valor lógico positivo (‘1’).



(a)



(b)

Figura 28: Diseño de la compuerta OR: (a) complejo principal y (b) complejo auxiliar.

5.4.3. Compuerta NOT

El diseño de la compuerta NOT se muestra en la Figura 31 y está compuesto por el complejo correspondiente a la compuerta (principal), un complejo auxiliar y la variable Y. La variable Y se encuentra presente desde un principio debido a que cuando la variable de entrada posee el valor de ‘0’ lógico (no está presente), la variable de salida debe ser un ‘1’ lógico. El complejo principal de la compuerta (Figura 31(a)) se compone de tres cadenas sencillas de ADN parcialmente complementarias prehibridadas: dos hebras de protección (una de desperdicio y una que reacciona con la compuerta auxiliar) y una cadena inferior

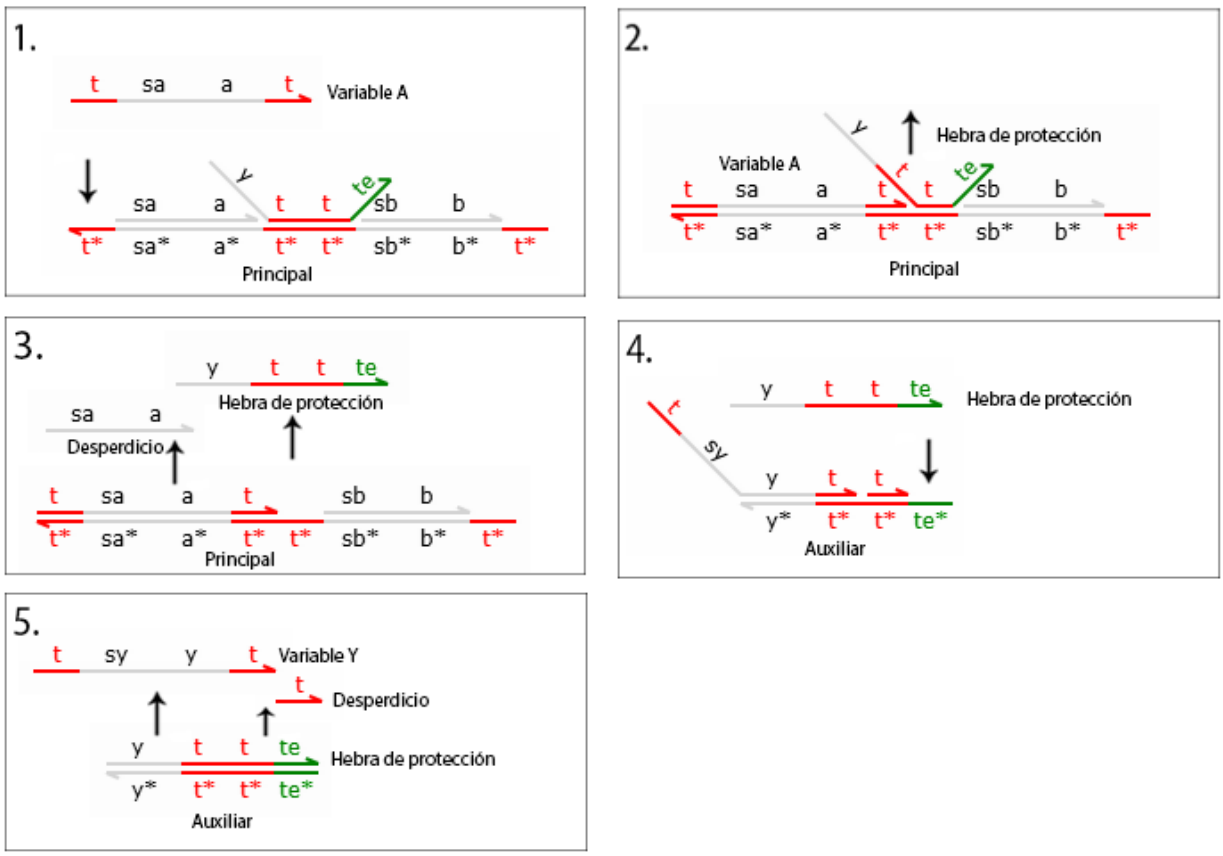


Figura 29: Resolución esquemática de la compuerta OR con una sola entrada (variable A).

(3′ − 5′) completamente complementaria a la variable de entrada A. Esta cadena inferior cuenta con un *toehold* universal complementario (dominio t*) para mantener prehibridada la hebra de protección que posee los *toeholds* específicos (dominios te) para reaccionar con el complejo auxiliar. El complejo auxiliar (Figura 31(b)) está compuesto por cuatro hebras sencillas de ADN. La cadena inferior es completamente complementaria a la variable Y en la parte izquierda y completamente complementaria a la hebra de protección de la compuerta principal NOT en la parte restante. Las tres cadenas superiores son hebras de protección desperdicio, de forma que si la hebra de protección reacciona con el complejo auxiliar se expone el *toehold* universal complementario (dominio t*) para que la variable de salida hibride, resultando en un '0' lógico ya que deja de estar presente. A continuación se describen de manera esquemática las reacciones que ocurren en los dos casos de la tabla de verdad para la compuerta en cuestión:

- Caso 1 (A = 0). En caso de que la variable de entrada no se encuentre presente

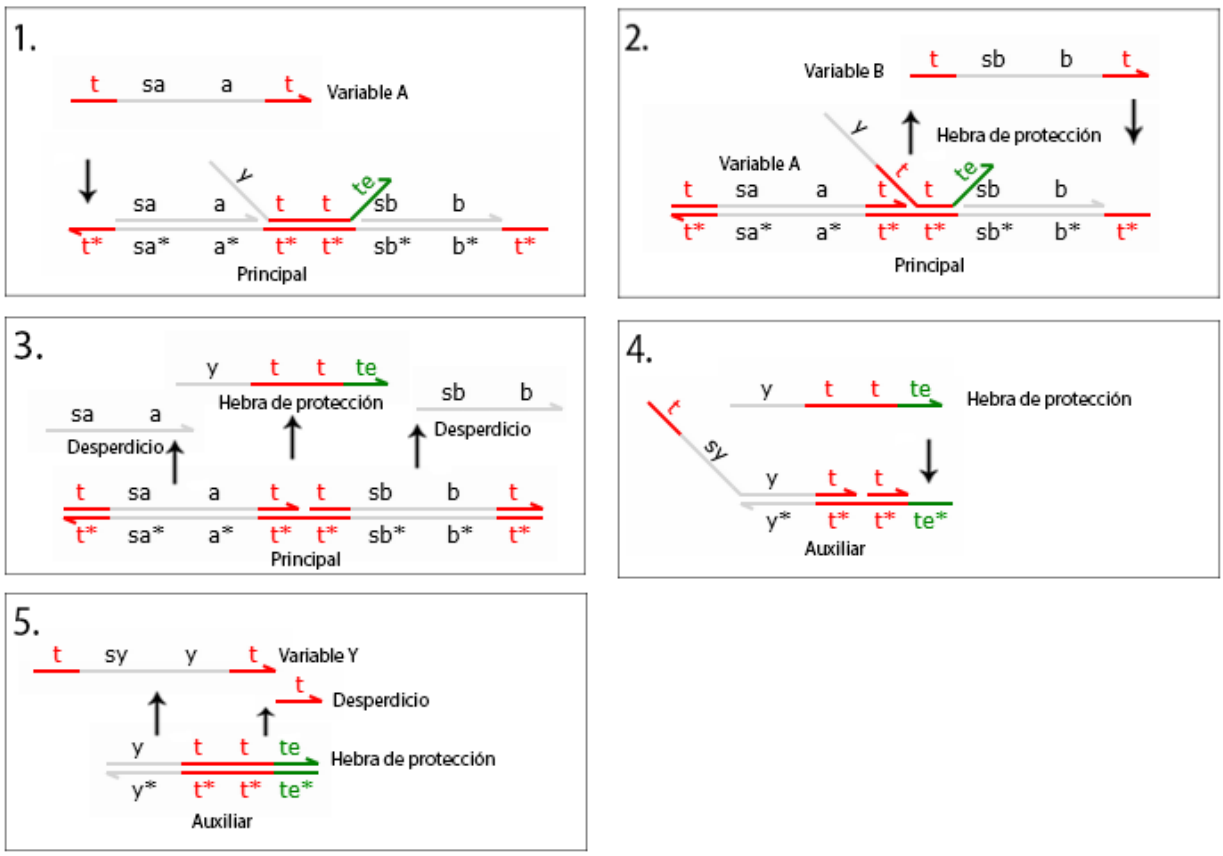


Figura 30: Resolución esquemática de la compuerta OR con dos entradas (variables A y B).

no ocurrirá reacción alguna, por lo que la variable de salida que se encuentra ya presente en el sistema representa un '1' lógico, concordando con la tabla de verdad.

- Caso 2 (A = 1). Cuando la variable de entrada está presente ('1' lógico), la variable de salida debe tener valor de '0' lógico (no estar presente). En la Figura 32 se puede apreciar la resolución esquemática de este caso, donde el primer paso consiste en la hibridación de la variable A con la hebra inferior del complejo principal por medio de los *toeholds* universales complementarios (dominios t*), desplazando completamente a la primera hebra de protección (desperdicio) y parcialmente la hebra de protección que reacciona con el complejo auxiliar como segundo paso. En el tercer paso, la hebra de protección de la compuerta ($5' - 3'$) se separa del complejo y comienza a hibridarse por medio de los *toeholds* universales con el complejo auxiliar, continuando el proceso de desplazamiento de las hebras de protección desperdicio mediante los *toeholds* específicos (dominios te) tal como se observa en el cuarto

paso. Como resultado de esto, queda expuesto un *toehold* universal complementario (dominio t*) que permite que se inicie la reacción de hibridación de la variable Y con su región complementaria en el complejo auxiliar, dando como resultado la ausencia de la variable Y y por ende un '0' lógico.

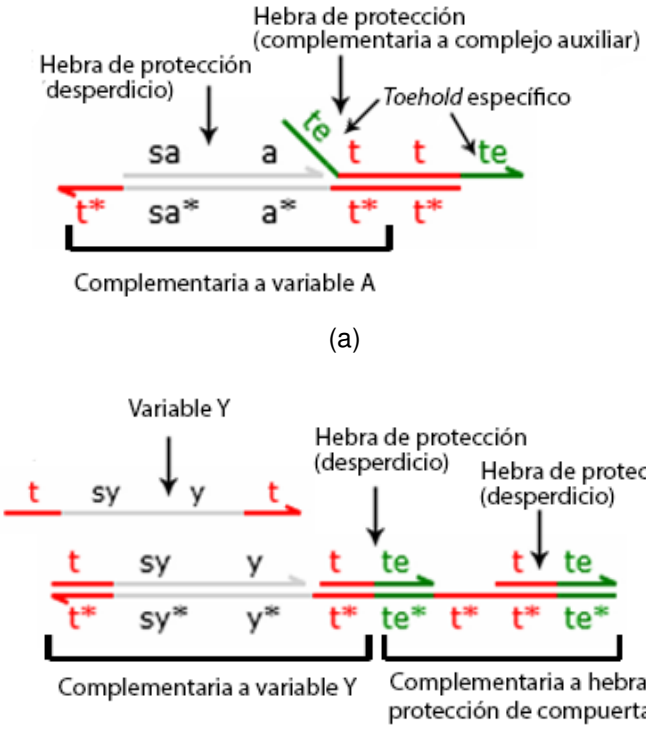


Figura 31: Diseño de la compuerta NOT: (a) complejo principal y (b) complejo auxiliar.

5.4.4. Compuerta NAND

El diseño propuesto para la compuerta NAND se muestra en la Figura 33 y está formado por el complejo correspondiente a la compuerta, un complejo auxiliar y la variable Y. El complejo principal de la compuerta (Figura 33(a)) se compone de cuatro moléculas ssADN parcialmente complementarias prehibridadas: una cadena inferior completamente complementaria a las variables de entrada, dos cadenas de protección de desperdicio y una hebra de protección que reacciona con el complejo auxiliar. La cadena inferior ($3' - 5'$) es completamente complementaria a la variable A por el lado izquierdo y por el lado derecho a la variable B; además, expone un *toehold* universal (dominio t*) en cada extremo de la cadena para iniciar las reacciones de desplazamiento. La hebra de protección con *toeholds* específicos (dominios te) reacciona con el complejo auxiliar para

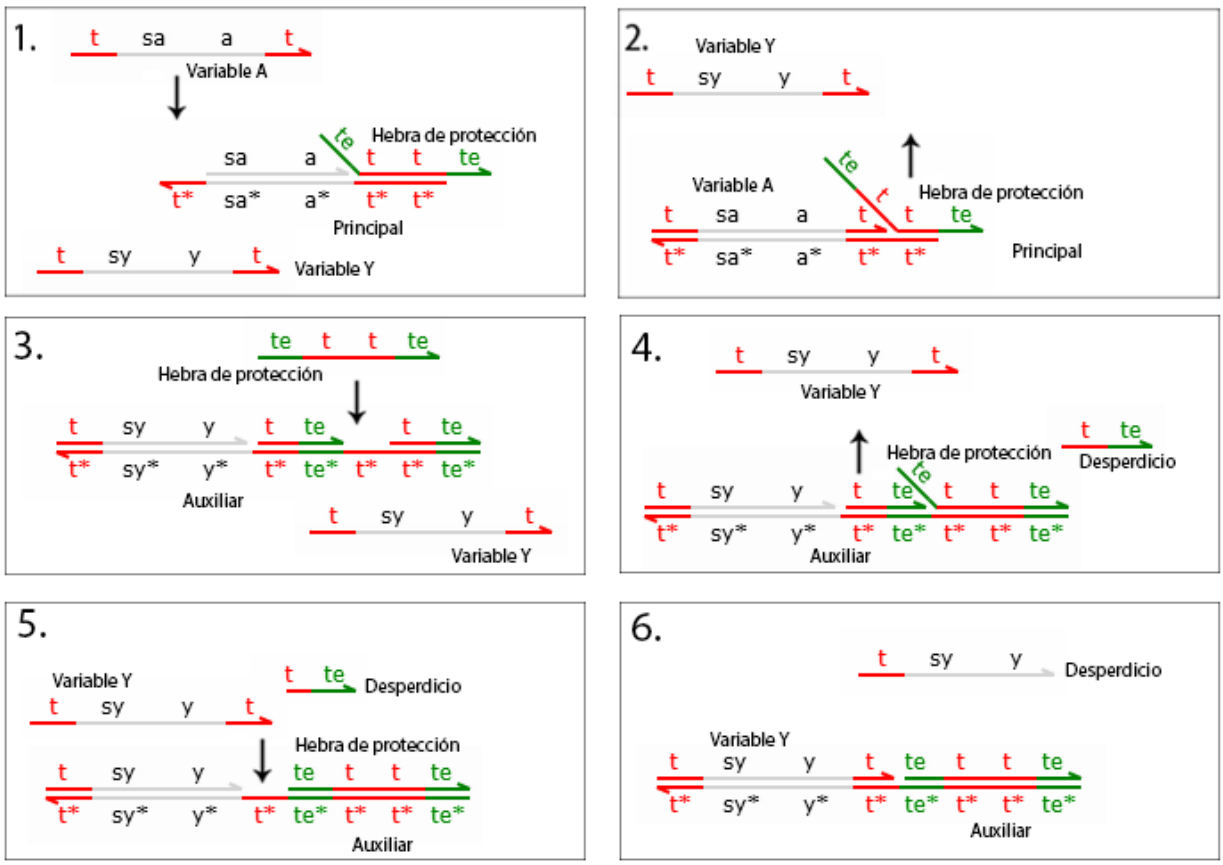


Figura 32: Resolución esquemática de la compuerta NOT con la entrada presente (variable A).

realizar las operaciones. El complejo auxiliar (Figura 33(b)) por su parte, está compuesto por tres hebras de protección de desperdicio y una cadena inferior. Esta cadena inferior es completamente complementaria a la variable Y por el lado izquierdo y completamente complementaria a la hebra de protección del complejo principal NAND por el lado derecho. Cabe mencionar que la única manera en que la compuerta NAND regresa un '0' lógico es cuando ambas variables tienen valor de '1', por lo que en el diseño de la compuerta la variable Y está presente desde un principio. A continuación, se describen de manera esquemática las reacciones que ocurren cuando no existen variables de entrada presentes, cuando sólo se encuentra una de ellas y cuando ambas están presentes:

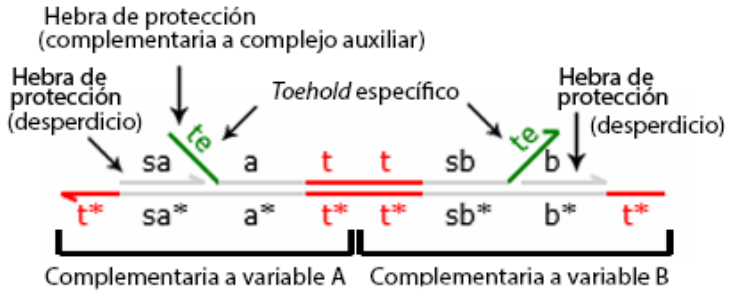
- Caso 1: No hay entradas (A = 0, B = 0). Al no estar presentes las variables de entrada no ocurren reacciones, por lo que la salida se mantiene en un '1' lógico.
- Casos 2 y 3: Sólo una entrada (A = 0, B = 1) y (A = 1, B = 0). En la Figura 34

se muestra la resolución esquemática del caso donde está presente una sola variable (variable A). El primer paso consiste en la hibridación de la variable A con el complejo principal mediante los *toeholds* universales complementarios (dominios t*). Como segundo paso se desplaza completamente la hebra de protección desperdicio y parcialmente la hebra de protección que reacciona con el complejo auxiliar. Observe que los enlaces entre los nucleótidos son suficientemente fuertes, por lo que esta última cadena continúa prehibridada. Consecuentemente, al estar presente la variable Y, la compuerta regresa un '1' lógico como salida.

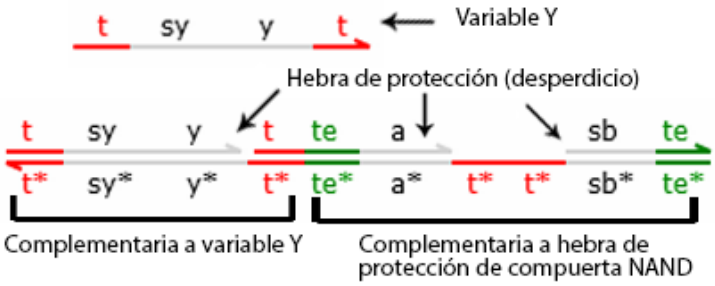
- Caso 4: Dos entradas (A = 1, B = 1). Cuando se encuentran presentes ambas variables de entrada (valor de '1'), la variable de salida debe tener valor de '0', por lo que ésta hibrida con el complejo auxiliar para que ya no se encuentre presente. Este caso se representa en la Figura 35, donde se puede observar que ambas variables de entrada hibridan con sus correspondientes partes complementarias por acción de los *toeholds* universales (dominios t) desplazando las hebras de protección desperdicio y la hebra de protección con los *toeholds* específicos (dominios te), tal como se ve reflejado en los pasos 1 - 3. Posteriormente, la hebra de protección ($5' - 3'$) del complejo principal comienza a hibridarse por medio de los *toeholds* universales y específicos complementarios con el complejo auxiliar (paso 4), desplazando completamente las dos hebras de protección desperdicio y dejando expuesto un *toehold* universal complementario (dominio t*) al que posee la variable Y (paso 5). Así comienza la reacción de hibridación entre ésta y el complejo auxiliar, desplazando la última hebra de protección desperdicio en el proceso (paso 6).

5.4.5. Compuerta XOR

El diseño propuesto para la compuerta XOR se muestra en la Figura 36 y lo conforman dos complejos, el complejo principal y un complejo auxiliar. El complejo principal (Figura 36(a)) se compone de siete ssADN parcialmente complementarias prehibridadas: una cadena inferior, cuatro hebras de protección que sirven para realizar las operaciones correpondientes y otras dos hebras de protección de desperdicio. La cadena inferior es completamente complementaria a las variables de entrada y a las hebras de protección A y B, en ese orden. Éstas últimas se utilizan para liberar la hebra de protección 2, la



(a)



(b)

Figura 33: Diseño de la compuerta NAND: (a) complejo principal y (b) complejo auxiliar.

cual reacciona con el complejo auxiliar para regresar un '1' lógico. La hebra de protección 1 se utiliza para monitorear si ambas entradas están presentes. De ser así, reacciona con el complejo auxiliar para producir un '0' lógico. El complejo auxiliar (Figura 36(b)) se compone de seis ssADN: una cadena inferior, cuatro hebras de protección desperdicio y la variable Y. La cadena inferior es completamente complementaria por el lado izquierdo a la variable Y, por en medio a la hebra de protección 1 y por el lado derecho a la hebra de protección 2. A continuación, se describen de manera esquemática las reacciones que ocurren cuando no existen variables de entrada presentes, cuando sólo se encuentra una de ellas y cuando ambas están presentes:

- Caso 1: No hay entradas (A = 0, B = 0). Al no estar presentes las variables de entrada no ocurren reacciones, por lo que la variable de salida no se encuentra presente indicando un '0' lógico.
- Casos 2 y 3: Sólo una entrada (A = 0, B = 1) y (A = 1, B = 0). Los únicos dos casos donde la compuerta regresa un '1' lógico es cuando está presente una sola varia-

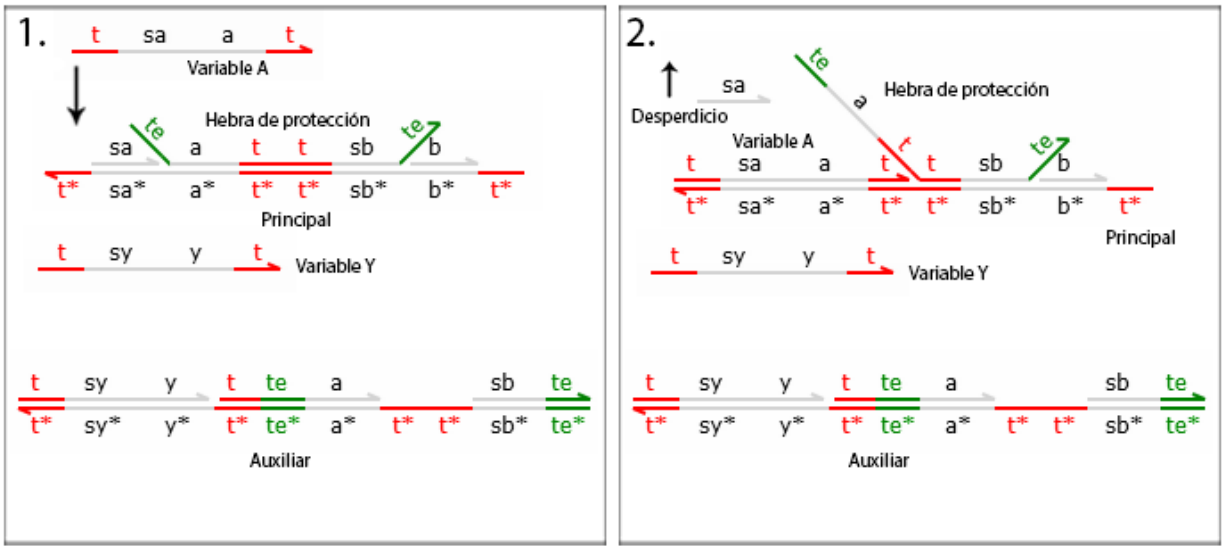


Figura 34: Resolución esquemática de la compuerta NAND con una sola entrada (variable A).

ble. La representación esquemática de uno de estos casos se puede apreciar en la Figura 37, donde solamente la variable A está presente. En el paso 1, la variable A comienza a hibridar con el complejo principal por medio de los *toeholds* universales complementarios (dominios t*), desplazando completamente la hebra de protección A (HPA) y parcialmente la hebra de protección 1, tal como se observa en el paso 2. En el paso 3, la HPA hibrida con el complejo principal mediante los *toeholds* adicionales (dominios ta) desplazando completamente la hebra de protección desperdicio y parcialmente la hebra de protección 2 (HP2). Dado que los enlaces entre las bases no son lo suficientemente fuertes, HP2 se separa por completo para hibridarse con el complejo auxiliar por medio de los *toeholds* específicos complementarios (paso 4), desplazando la hebra de protección desperdicio y la cadena correspondiente a la variable Y, lo que resulta en un '1' lógico. Cuando solamente está presente la variable B, sucede exactamente el mismo procedimiento sólo que la hebra de protección encargada de iniciar la reacción es la hebra de protección B, en lugar de la hebra de protección A.

- Caso 4: Dos entradas (A = 1, B = 1). En caso de que ambas variables de entrada estén presentes, la variable de salida debería estar ausente para que la compuerta XOR regrese un '0' lógico. En la Figura 38 se presenta la resolución esquemática de

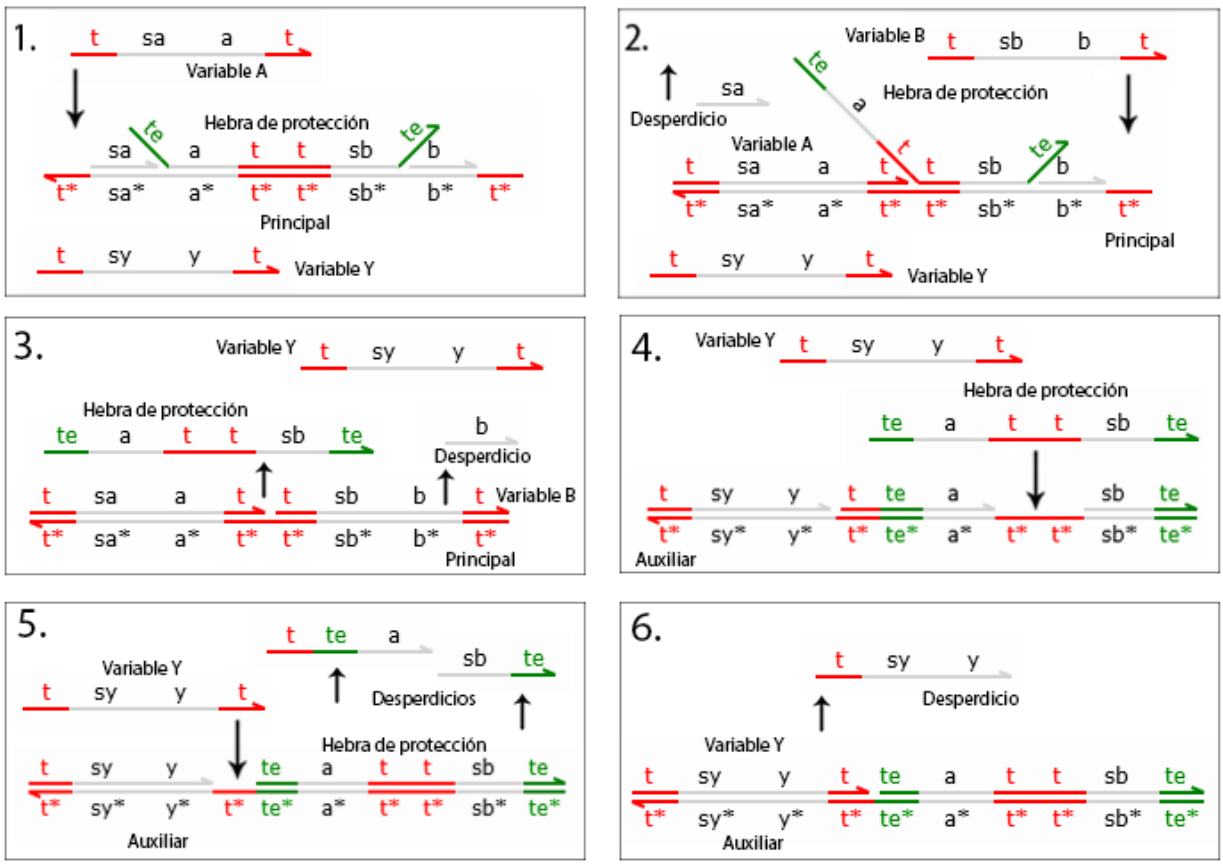
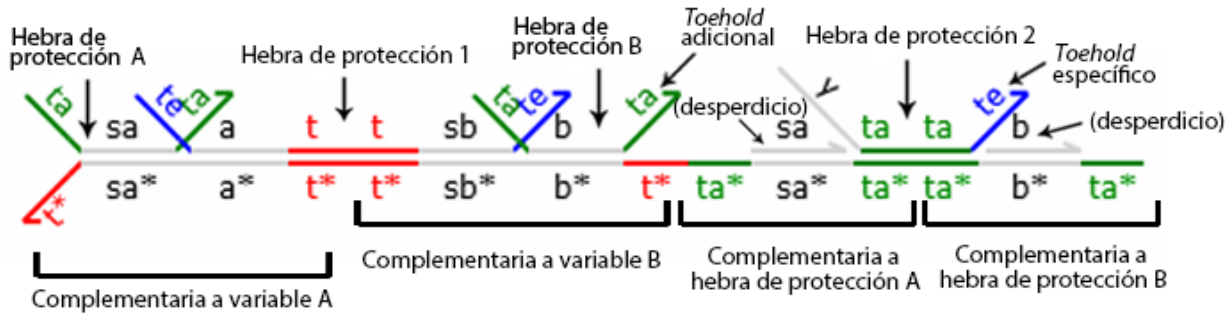
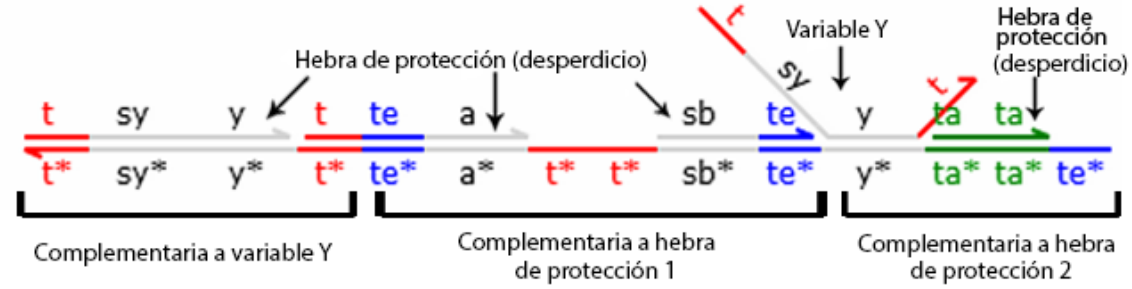


Figura 35: Resolución esquemática de la compuerta NAND con dos entradas (variables A y B).

este caso. Como primer paso hibridan las variables A y B con el complejo principal, desplazando las hebras de protección A y B (HPA y HPB), así como la hebra de protección 1 (paso 2). En el paso 3, HPA y HPB hibridan con sus secciones complementarias a los *toeholds* adicionales del complejo principal, desplazando completamente la hebra de protección 2 y las 2 hebras de protección de desperdicio (paso 4). En el paso 5, la hebra de protección 2 hibrida con el complejo auxiliar desplazando las dos hebras de protección de desperdicio y la variable Y (paso 6). La hebra de protección 1 comienza entonces a hibridarse con el complejo auxiliar mediante los *toeholds* universales desplazando las dos hebras de protección desperdicio que se encontraban prehibridadas, quedando como consecuencia expuesto un *toehold* universal complementario (paso 7). Posteriormente, la variable Y comienza el proceso de hibridación con el complejo auxiliar y desplaza la hebra de protección desperdicio correspondiente (paso 8), dando como resultado un '0' lógico.



(a)



(b)

Figura 36: Diseño de la compuerta XOR: (a) complejo principal y (b) complejo auxiliar.

5.4.6. Compuerta NOR

Para la compuerta NOR se propone el diseño que se muestra en la Figura 39, el cual se compone por un complejo principal, un complejo auxiliar y la variable Y. Esta variable se encuentra presente desde un principio debido a que el único caso donde la compuerta NOR retorna un valor de '1' lógico es cuando ninguna de las dos variables están presentes. El complejo principal (Figura 39(a)) se compone de cuatro ssADN parcialmente complementarias prehibridadas. La cadena inferior es completamente complementaria a las variables de entrada A y B. Las cadenas superiores incluyen dos hebras de protección desperdicio y la hebra de protección que contiene los *toeholds* específicos (dominios te) para reaccionar con el auxiliar. El complejo auxiliar (Figura 39(b)) también se conforma por cuatro ssADN: tres cadenas superiores que funcionan como hebras de protección (desperdicios) y una cadena inferior. La cadena inferior ($3' - 5'$) es completamente complementaria a la variable Y en el lado izquierdo y a la hebra de protección del complejo principal en el lado derecho. La cadena expone dos *toeholds* universales complementarios (dominios te*) para iniciar el proceso de hibridación con la hebra de protección

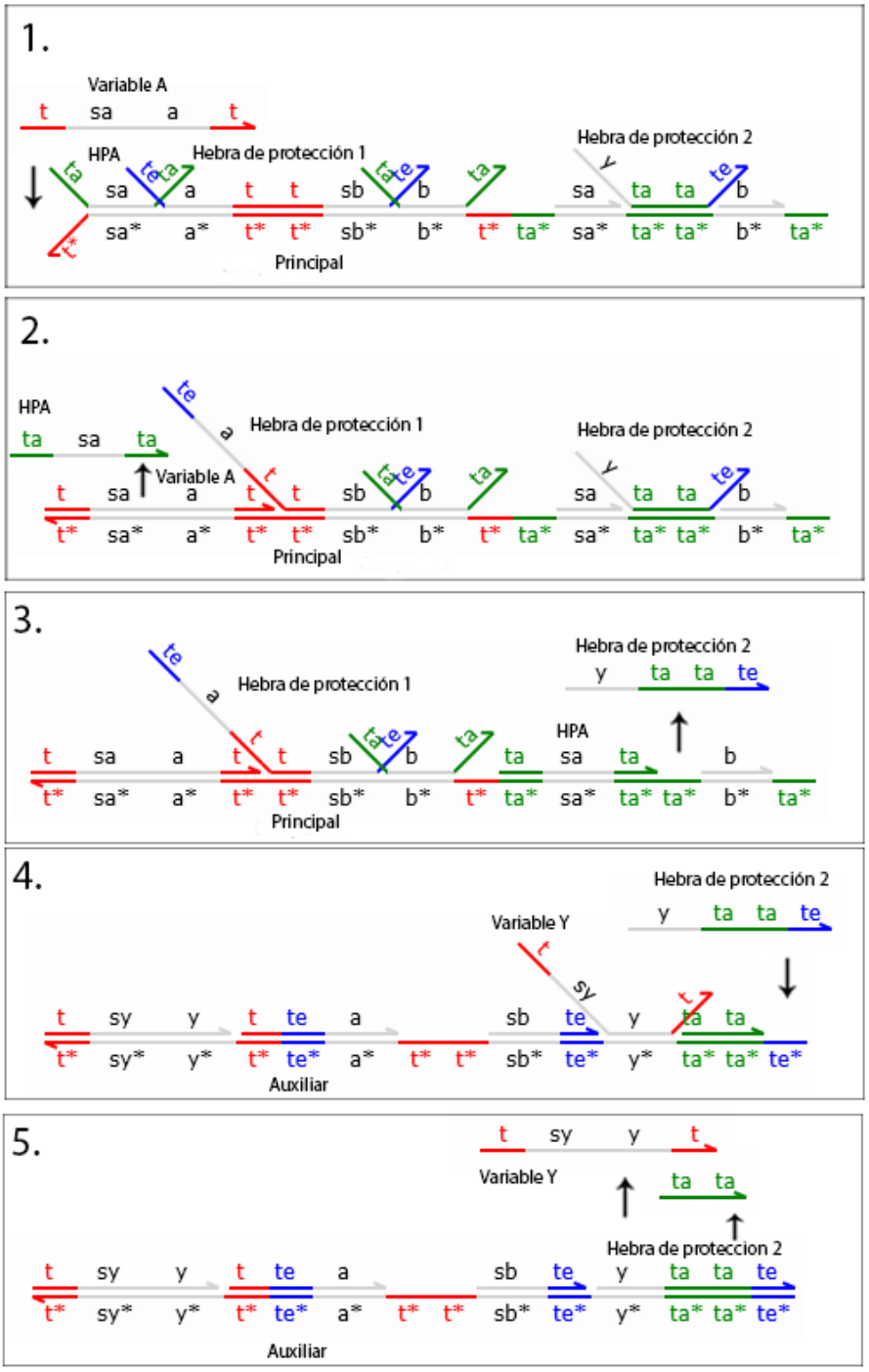


Figura 37: Resolución esquemática de la compuerta XOR con una sola entrada (variable A).

una vez que ésta se encuentre liberada. Esto permite que la variable de salida Y hibride con el complejo auxiliar, indicando un '0' lógico. A continuación, se describen de manera esquemática las reacciones que ocurren en cada uno de los casos de la tabla de verdad:

- Caso 1: No hay entradas (A = 0, B = 0). Al no estar presentes las variables de entrada no ocurren reacciones, por lo que la salida se mantiene en un ‘1’ lógico.

- Casos 2 y 3: Sólo una entrada (A = 0, B = 1) y (A = 1, B = 0). En la Figura 40 se esquematizan las reacciones que ocurren cuando solamente la variable A está presente. Primero, la variable de entrada comienza a hibridarse con su parte complementaria en el complejo principal por medio de los *toeholds* universales (paso 1) desplazando completamente la hebra de protección (desperdicio) y parcialmente la hebra de protección con los *toeholds* específicos (paso 2). Ésta última, al quedar hibridada solamente por los *toeholds*, termina por separarse ya que los enlaces no son lo suficientemente fuertes (paso 3). La hebra de protección del complejo principal hibrida con el complejo auxiliar por medio de los *toeholds* universales y específicos correspondientes (paso 4), desplazando las 3 hebras de protección desperdicio en el proceso y dejando expuesto el *toehold* universal complementario (paso 6). Esto permite que la variable Y se hibride, regresando un ‘0’ lógico como consecuencia (paso 7).

- Caso 4: Dos entradas (A = 1, B = 1). En la Figura 41 se tiene la resolución esquemática de este caso, donde primeramente hibridan las variables A y B con los *toeholds* universales del complejo principal (pasos 1 y 2), liberando las hebras de protección desperdicio correspondientes y la hebra de protección que reacciona con el complejo auxiliar (paso 3). Dicha hebra hibrida con el complejo auxiliar mediante los *toeholds* universales y específicos complementarios (paso 4), desplazando las hebras de protección desperdicio y dejando expuesto el *toehold* necesario para que la variable Y hibride con el complejo auxiliar (paso 6), dando como resultado un ‘0’ lógico al no estar presente.

5.4.7. Compuerta XNOR

El diseño propuesto para la compuerta XNOR se muestra en la Figura 42 y se compone de un complejo principal, un complejo auxiliar y la variable Y. La compuerta XNOR regresa una salida verdadera si ambas variables de entrada poseen el mismo valor. La variable Y se encuentra presente desde un inicio para cubrir el primer caso de la tabla de

verdad, donde ambas variables de entrada no están presentes y la salida corresponde a un ‘1’ lógico (variable Y presente). El complejo principal (Figura 42(a)) está formado por cinco cadenas sencillas de ADN prehibridadas: cuatro cadenas superiores que son hebras de protección y una cadena inferior. Las hebras de protección A y B poseen *toeholds* adicionales (dominios ta) que reaccionan con el mismo complejo para monitorear si ambas entradas están presentes. Las hebras de protección 1 y 2 tienen *toeholds* específicos (dominios te) que reaccionan con el complejo auxiliar para liberar o atrapar la variable de salida Y. La cadena inferior ($3' - 5'$) es completamente complementaria a la variable A, variable B, a la hebra de protección A y a la hebra de protección B (en ese orden) y expone los *toeholds* universales complementarios (dominios t*) para iniciar la reacción en cascada. El complejo auxiliar (Figura 42(b)) se construye a partir de seis ssADN: cuatro hebras de protección desperdicio, la variable Y y una cadena inferior. La cadena inferior posee sitios complementarios a la variable Y, a la hebra de protección 1 y a la hebra de protección 2. Observe que hay dos cadenas representando la variable Y, esto se debe a que cuando no se detectan entradas, la variable Y debe estar presente para cumplir con el primer caso de la tabla de verdad. Cuando se detecta una sola entrada, el complejo auxiliar atrapa la variable de salida para regresar un ‘0’ lógico y así cumplir con los siguientes dos casos de la tabla de verdad; sin embargo, si se detecta la presencia de la otra variable de entrada (esto se hace con las hebras de protección A y B), se libera la variable Y(2) para regresar un ‘1’ lógico y cubrir el último caso. A continuación, se describen de manera esquemática las reacciones que ocurren en cada uno de los casos de la tabla de verdad:

- Caso 1: No hay entradas (A = 0, B = 0). Al no estar presentes las variables de entrada no ocurren reacciones, por lo que la salida se mantiene en un ‘1’ lógico.
- Casos 2 y 3: Sólo una entrada (A = 0, B = 1) y (A = 1, B = 0). En la Figura 43 se representa esquemáticamente el caso donde sólo está presente la variable A. Como primer paso, se tiene la hibridación de la variable A con el complejo principal por medio de los *toeholds* universales complementarios (dominio t*), desplazando completamente la hebra de protección A (HPA) y parcialmente la hebra de protección 1 (paso 2). Ésta última, al quedar hibridada solamente por un *toehold*, termina

por desprenderse. A su vez, HPA hibrida con el complejo principal por medio de los *toeholds* adicionales complementarios (dominios ta*), desplazando parcialmente la hebra de protección 2 (paso 3). La hebra de protección 1 comienza a hibridarse con su sitio complementario en el complejo auxiliar por medio de los *toeholds* universales y específicos complementarios (paso 4) desplazando completamente una de las hebras de protección desperdicio y parcialmente a la otra, dejando expuesto el *toehold* universal complementario (paso 5). Esto permite que la variable Y pueda hibridarse con su sitio complementario en el complejo auxiliar, representando así un '0' lógico (paso 6).

- Caso 4: Dos entradas (A = 1, B = 1). En este caso, la variable Y debe estar presente para representar un '1' lógico como salida. El esquema de las reacciones que ocurren se puede observar en la Figura 44. El primer paso consiste en la hibridación de ambas variables de entrada (A y B) en el complejo principal. Esto sucede por medio de los *toeholds* universales, desplazando completamente las hebras de protección A y B, así como la hebra de protección 1 (paso 2). Como tercer paso, las hebras de protección A y B (HPA y HPB) hibridan con sus sitios complementarios en el complejo principal por medio de los *toeholds* adicionales, liberando la hebra de protección 2. Como cuarto paso, la hebra de protección 1 hibrida con el complejo auxiliar por medio de los *toeholds* universales y específicos complementarios, al igual que la hebra de protección 2, a través de los *toeholds* adicionales. Al hibridarse la hebra de protección 1, ésta desplaza completamente una de las hebras de protección desperdicio y parcialmente a la otra, dejando expuesto el *toehold* universal complementario (paso 5) para que la variable Y pueda hibridarse con el complejo auxiliar, representando así un '0' lógico (paso 6). Sin embargo, al hibridarse la hebra de protección 2, se desplaza la segunda cadena que codifica la variable de salida, es decir, la variable Y(2), dando como resultado un '1' lógico.

A manera de discusión, es importante mencionar que hasta ahora no existe una metodología específica para diseñar dispositivos lógicos con ADN. Uno de los puntos cruciales en el diseño es que los modelos deben apegarse a la realidad de modo que, al construir físicamente las estructuras diseñadas, éstas mantengan la funcionalidad esperada; sin

embargo, debido a la naturaleza aleatoria de las reacciones moleculares involucradas lo anterior no se puede garantizar completamente. Otro punto importante a considerar es la representación de las variables puesto que se busca que el sistema generado sea robusto, es por esto que, en el presente trabajo la representación propuesta permite que las variables de salida actúen como entradas para otras compuertas, lo que posibilita la construcción de circuitos de varias etapas. Adicionalmente, es necesario considerar que en caso de que alguna variable o hebra de protección equivocada interactúe con una compuerta, ésta debe encontrar dificultades para hibridarse, por ello se propuso utilizar *toeholds* específicos y adicionales en los diseños con objeto de evitar en la mayor medida posible este tipo de interferencias.

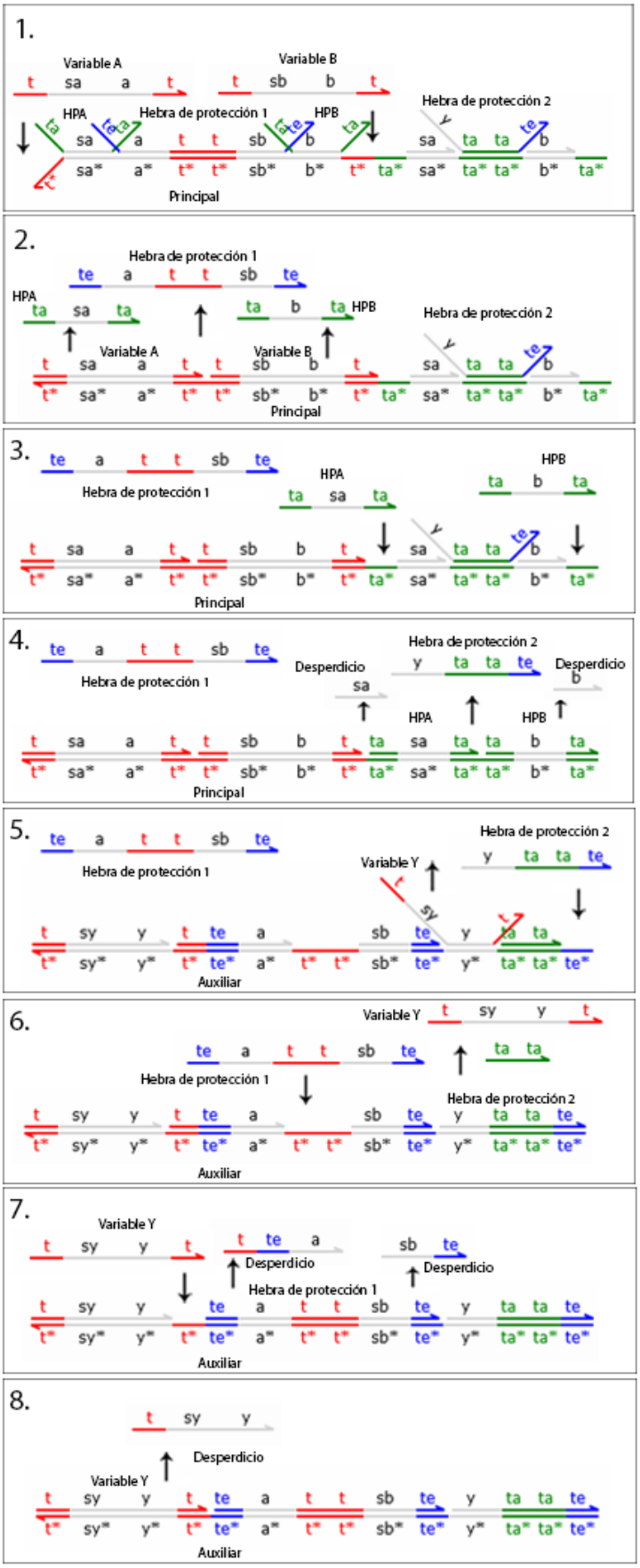
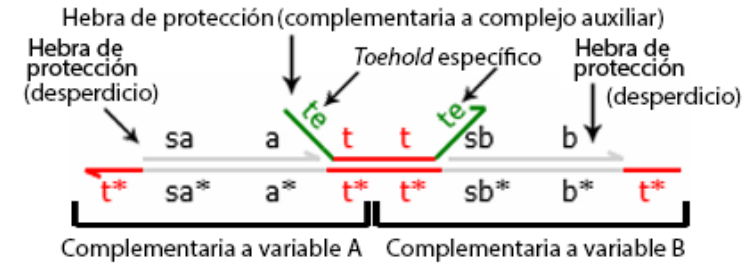
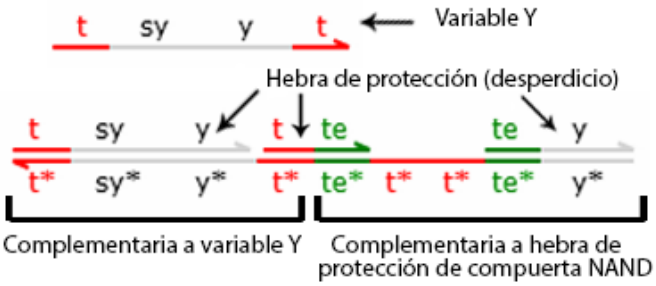


Figura 38: Resolución esquemática de la compuerta XOR con dos entradas (variables A y B).



(a)



(b)

Figura 39: Diseño de la compuerta NOR: (a) complejo principal y (b) complejo auxiliar.

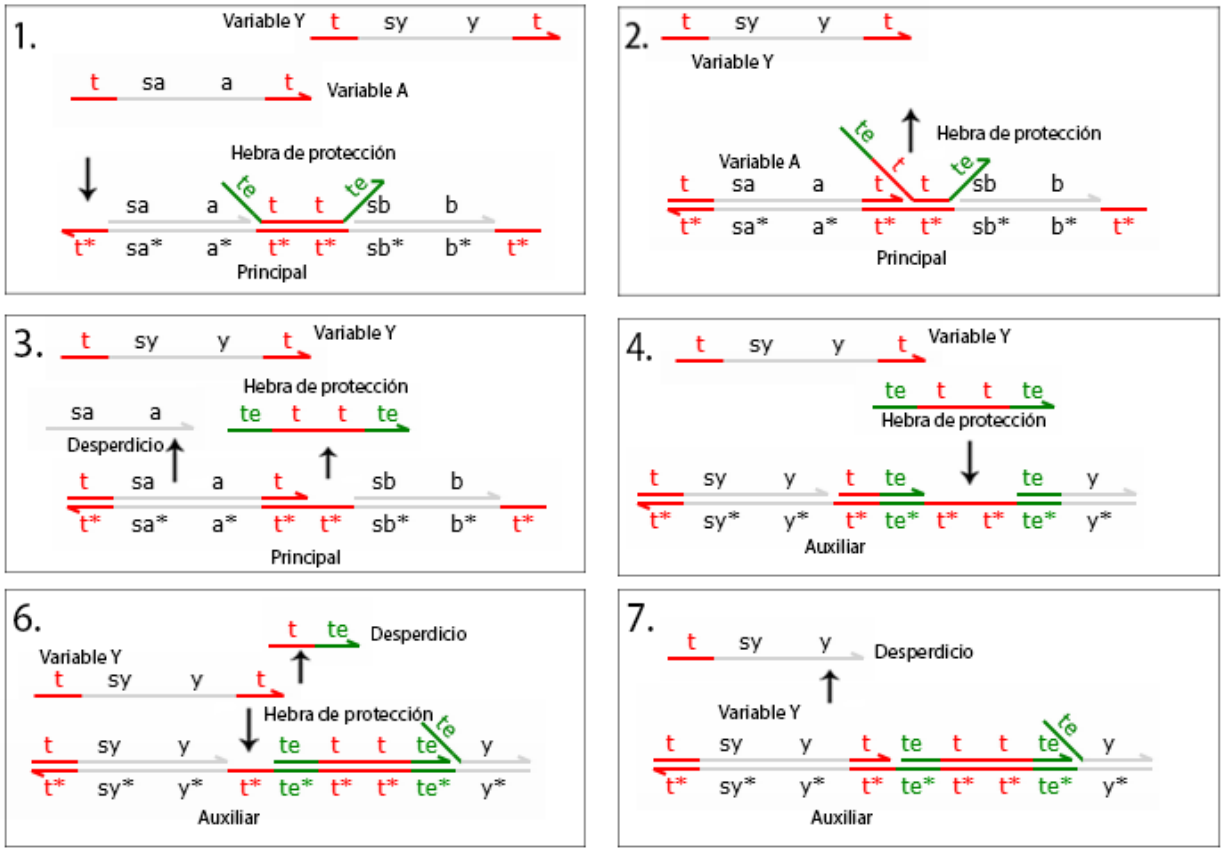


Figura 40: Resolución esquemática de la compuerta NOR con una sola entrada (variable A).

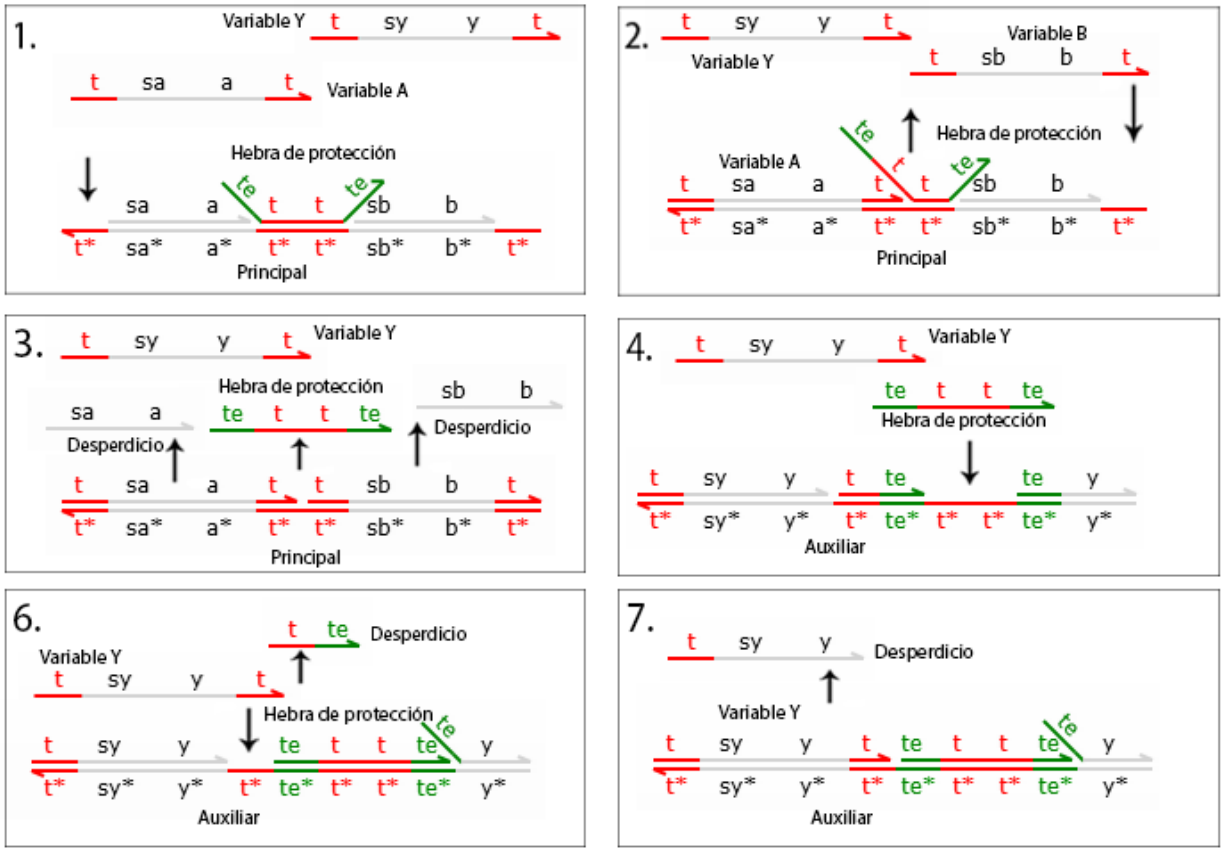
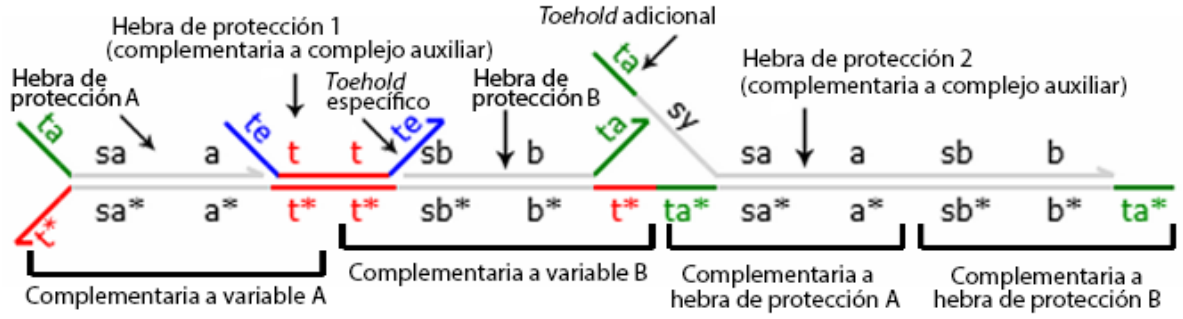
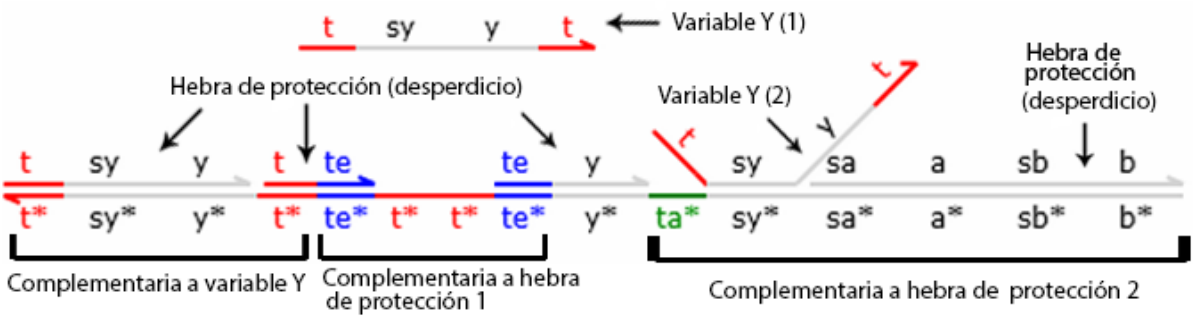


Figura 41: Resolución esquemática de la compuerta NOR con dos entradas (variables A y B).



(a)



(b)

Figura 42: Diseño de la compuerta XNOR: (a) complejo principal y (b) complejo auxiliar.

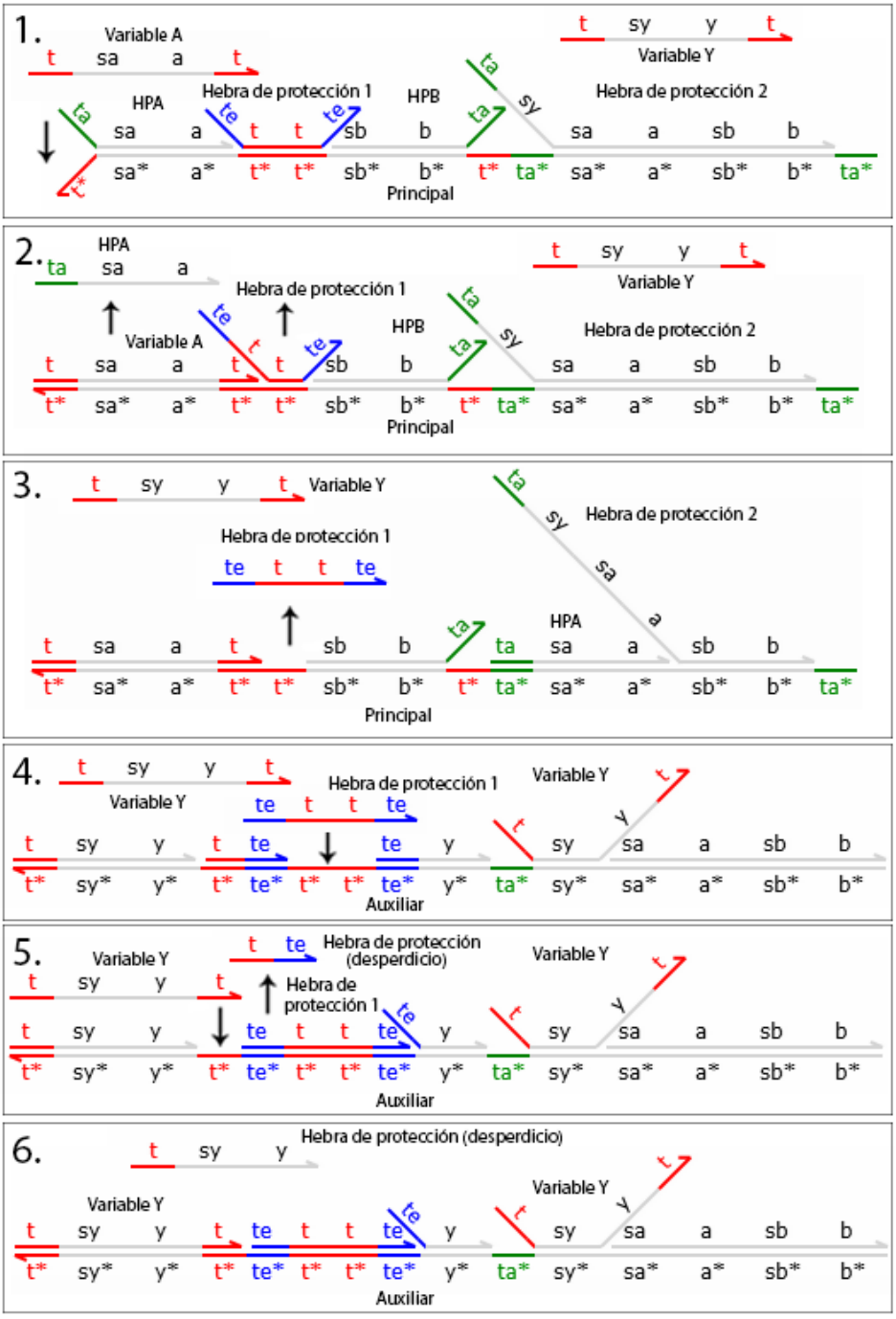


Figura 43: Resolución esquemática de la compuerta XNOR con una sola entrada (variable A).

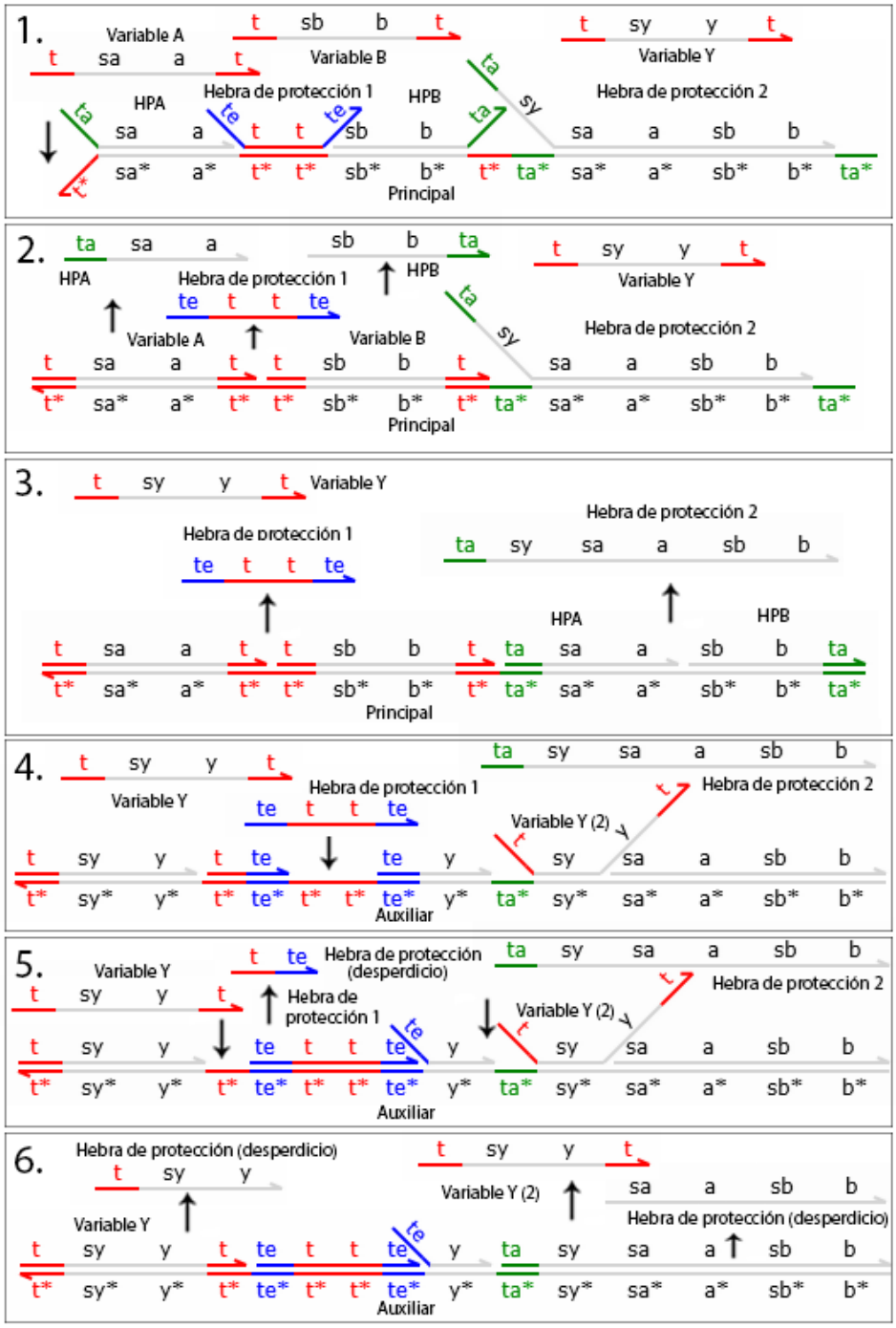


Figura 44: Resolución esquemática de la compuerta XNOR con dos entradas (variables A y B).

Capítulo 6. Simulación estocástica y desempeño de compuertas lógicas

En este capítulo se presentan los resultados de las simulaciones de los modelos propuestos mediante herramientas computacionales específicas. Para probar el funcionamiento de las estructuras diseñadas se realizaron experimentos *in silico*. Las simulaciones estocásticas se realizaron por medio de la herramienta Microsoft Visual DSD. Además, se realizó el análisis termodinámico de los complejos de ADN correspondientes a las compuertas lógicas con la herramienta NUPACK. Para el análisis de algunas propiedades cuantitavas de los diseños se utilizaron técnicas de verificación probabilística mediante la herramienta computacional PRISM.

6.1. Configuración de herramientas computacionales

6.1.1. NUPACK

Para que el modelo sea lo más realista posible, es necesario que los complejos de ADN exhiban el comportamiento deseado de manera física, particularmente en relación con la estabilidad de la estructuras, por lo que se realizó un análisis termodinámico. Este análisis consistió en el cálculo de la energía libre de cada una de las estructuras propuestas por medio de la herramienta computacional NUPACK utilizando los siguientes parámetros:

- Temperatura de 37°C.
- Modelo del vecino más cercano y parámetros energéticos de ADN de SantaLucia y Hicks (2004).
- Concentraciones de sales: Na^{+}= 1.0 M (SantaLucia y Hicks, 2004) y Mg^{++}= 0.0 M (Koehler y Peyret, 2005).

6.1.2. Visual DSD

Las simulaciones estocásticas de los modelos diseñados se realizaron mediante el algoritmo de Gillespie (Gillespie, 1977), el cual simula la evolución en el tiempo de un

sistema de reacciones químicas y por medio de cálculos basados en una constante de reacción estocástica, determina cuándo y de qué tipo será la siguiente reacción que ocurrirá (más detalles en el Capítulo 3). El algoritmo utilizado se encuentra implementado en el software Microsoft Visual DSD, que permite crear y analizar prototipos de dispositivos de cómputo diseñados mediante reacciones de desplazamiento de cadenas de ADN (ver Capítulo 4.6.2) por lo que las simulaciones se llevaron a cabo empleando esta herramienta.

Para cada uno de los siete modelos de compuertas propuestos, la simulación consistió en 10 ejecuciones para cada una de las posibles combinaciones de entradas según la tabla de verdad de cada compuerta. Éstas 10 ejecuciones se realizaron para:

- 1,000 moléculas de variables de entrada así como complejos principales y auxiliares.
- 10,000 moléculas de variables de entrada así como complejos principales y auxiliares.
- 100,000 moléculas de variables de entrada así como complejos principales y auxiliares.

Los parámetros utilizados en las simulaciones estocásticas corresponden a los siguientes:

- Velocidad de migración de ramificaciones de 8000 s^{-1} por nucleótido.
- Tasa de hibridación de $3 \times 10^{-4} nM^{-1} s^{-1}$.
- Tasa de desnaturalización de $0.1126\, s^{-1}$.
- 1000 puntos de muestra.
- El tiempo de simulación es variable para cada compuerta según el número de moléculas en el sistema.

En las siguientes secciones se presentan los resultados experimentales para cada uno de los modelos propuestos. Debido a que el comportamiento de las simulaciones fue similar para 1,000, 10,000 y 100,000 moléculas, sólo se muestran las gráficas de las simulaciones promedio para 100,000 ya que estás son las más representativas. Las gráficas muestran los primeros 200 instantes de tiempo de simulación puesto que de esta manera es más claro observar el comportamiento de las moléculas que intervienen en todos los diferentes casos de entrada al sistema. El único caso donde se muestra el comportamiento del sistema durante el tiempo completo de simulación es en el primero (A = 0, B = 0) debido a que se desea probar que el comportamiento es el mismo durante todo el tiempo de simulación.

6.2. PRISM

La herramienta Visual DSD presenta la opción de generar los modelos de cadenas de Markov en tiempo continuo de los dispositivos simulados en formato *sm* (utilizado por PRISM). Se aprovechó esta característica para realizar un análisis mediante la herramienta PRISM con el objetivo de confirmar que los modelos propuestos fueran correctos. El análisis consistió en la verificación de ciertas propiedades cuantitativas de cada uno de los modelos tales como: el tiempo estimado para que el sistema alcance un estado final exitoso, la probabilidad de que se alcance eventualmente un estado exitoso, la probabilidad de que eventualmente se alcance un estado final erróneo, la probabilidad de alcanzar un estado final exitoso en un tiempo T y la probabilidad de alcanzar un estado final erróneo en un tiempo T. Dichas propiedades se expresan formalmente en lógica temporal para ser analizadas automáticamente por el software en cuestión. El primer paso para la verificación consiste en identificar los estados finales o exitosos del modelo, esto se consigue de manera genérica, tal como se sugiere en (Lakin *et al.*, 2012a), con el siguiente código en lenguaje PRISM:

```
label "all_done" = strands_reactive = output & output = N & gates_reactive = 0;
```

donde *"all_done"* es una etiqueta boolena creada a partir de la palabra reservada `label`, `strands_reactive` y `gates_reactive` son variables (generadas automáticamente por Visual DSD) que contabilizan el número de cadenas y compuertas capaces de generar

reacciones, es decir, que contengan *toeholds* expuestos capaces de iniciar reacciones de desplazamiento. Las variables `output` y `N` contabilizan el número de resultados generados y la cantidad de resultados esperados, respectivamente. Cabe mencionar que fue necesario modificar las variables `strands_reactive` y `gates_reactive` para redefinir el estado exitoso de cada uno de los modelos de las compuertas ya que, contrario al ejemplo genérico, los modelos propuestos están diseñados para admitir dos entradas, por lo que de sólo estar presente alguna de ellas quedarían *toeholds* activos en los complejos compuerta. Una vez que el estado final exitoso se define, se procede a verificar la estabilidad del modelo, utilizando las expresiones en lógica temporal sugeridas en (Lakin *et al.*, 2012a), las cuales indican que el estado final deseado debe especificarse por la etiqueta *"all_done"* y debe existir al menos alguna ruta que conduzca a dicho estado:

- A [G *"deadlock"* => *"all_done"*]
- E [F *"all_done"*]

donde *"deadlock"* es una etiqueta booleana reservada para definir los estados finales del modelo (no necesariamente exitosos), es decir, los estados donde ya no se pueden generar más reacciones. Las literales A y E representan el operador universal y existencial, respectivamente. Las literales G y F se refieren a que la evaluación se realiza de forma global y que la condición a evaluar se espera que se cumpla de manera futura o eventual. Al verificar estas expresiones se obtiene un valor booleano; para afirmar que el modelo es correcto o estable, ambas propiedades deben regresar un valor positivo, de lo contrario la herramienta proporciona un contraejemplo a manera de demostración. Cabe mencionar que la evaluación de las propiedades anteriores fue positiva para todos los modelos propuestos. Las propiedades cuantitativas evaluadas probabilísticamente se expresan en lógica temporal *CLS* de la siguiente manera:

- R {"time"} = ? [F *"all_done"*]: permite calcular el tiempo estimado para que el sistema alcance un estado final exitoso.
- P = ? [F *"all_done"*]: indica la probabilidad de que el sistema alcance eventualmente un estado exitoso.

- P = ? [F *“deadlock”* &! *“all_done”*]: indica la probabilidad de que el sistema alcance eventualmente un estado final erróneo.
- P = ? [F [T, T] *“all_done”*]: indica la probabilidad de que el sistema alcance un estado exitoso en un tiempo T.
- P = ? [F [T, T] *“deadlock”* &! *“all_done”*]: indica la probabilidad de que el sistema alcance un estado final erróneo en un tiempo T.

6.3. Resultados experimentales

6.3.1. Compuerta AND

En la Tabla 2 se muestran las secuencias pertenecientes a cada uno de los dominios de las cadenas de ADN que se utilizaron para realizar las simulaciones estocásticas del modelo de la compuerta AND. Cabe mencionar que, tanto las secuencias utilizadas para esta compuerta como para todas las demás fueron obtenidas por medio de la herramienta Visual DSD y cumplen con las necesidades de los diseños, por lo que el utilizar otras secuencias alteraría los resultados de las simulaciones. Este complejo (Figura 45(a)) está formado por un total de 220 nucleótidos y presenta una energía libre de $-108.26\,kcal/mol$ mientras que el complejo auxiliar (Figura 45(b)) está formado por 260 nucleótidos en total y presenta una energía libre de $-131.21\,kcal/mol$, por lo que ambos son termodinámicamente estables. La Figura 46 muestra la gráfica del promedio de las 10 simulaciones para los cuatro casos de la tabla de verdad de la compuerta AND para 100,000 moléculas de entrada. En total, cada simulación requirió 17,000 unidades de tiempo:

- Caso 1 (A = 0, B = 0). En la Figura 46(a) se puede observar que las concentraciones del complejo principal y el complejo auxiliar no se alteran, ya que no se encuentran presentes ninguna de las dos variables de entrada (A y B). Este comportamiento no se puede apreciar directamente puesto que la línea correspondiente a la concentración de la variable Y (que también es de 0 moléculas) está superpuesta. La fiabilidad para este caso es del 100 % dado que de las 0 moléculas de la variable de salida Y que se esperaban, se obtuvieron 0.

- Caso 2 (A = 0, B = 1). La simulación promedio para esta combinación de entradas se muestra en la Figura 46(b), donde se puede apreciar cómo la variable B se va consumiendo a lo largo del tiempo de simulación al hibridarse con el complejo principal (por lo que disminuye la concentración de la compuerta), aumentando la concentración del complejo AND/B y en el caso de la variable Y, la concentración siempre es de 0 moléculas. La fiabilidad para este caso es del 100 % dado que de las 0 moléculas de la variable de salida Y que se esperaban, se obtuvieron 0.

- Caso 3 (A = 1, B = 0). En la Figura 46(c) se tiene la simulación promedio del caso donde sólo está presente la variable A, cuyo comportamiento es exactamente el mismo que el caso descrito anteriormente (cuando sólo está B), salvo que ahora las reacciones se desencadenan a través de la variable A. La fiabilidad para este caso es del 100 % dado que de las 0 moléculas de la variable de salida Y que se esperaban, se obtuvieron 0.

- Caso 4 (A = 1, B = 1). La simulación promedio del caso donde están ambas variables presentes se muestra en la Figura 46(d), donde se observa cómo las variables de entrada se van consumiendo al hibridarse con el complejo principal, disminuyendo como consecuencia la concentración del complejo auxiliar al liberar la variable de salida Y, cuya concentración aumenta a lo largo del tiempo indicando un '1' lógico. La fiabilidad para este caso es del 99.9 % dado que de las 100,000 moléculas de la variable de salida Y que se esperaban, se obtuvieron 99,996.

Con respecto a las propiedades cuantitativas evaluadas, la probabilidad de alcanzar un estado final existoso eventualmente es de 1 mientras que la probabilidad de alcanzar un estado final erróneo es igual a 0, lo cual se puede observar en la Figura 47. De aquí se puede inferir que el modelo es estable y funciona de la manera esperada. El tiempo estimado para alcanzar un estado final es de aproximadamente 633,650 unidades de tiempo. En cuanto al análisis del espacio de estados, los resultados obtenidos con Visual DSD y PRISM coincidieron y se pueden consultar en el Apéndice A.1. Al analizar las simulaciones se puede observar que las reacciones se llevaron a cabo de la manera esperada y por lo tanto se consiguió emular el comportamiento propio de la compuerta AND.

Tabla 2: Secuencias correspondientes a cada dominio para las simulaciones estocásticas de la compuerta AND.

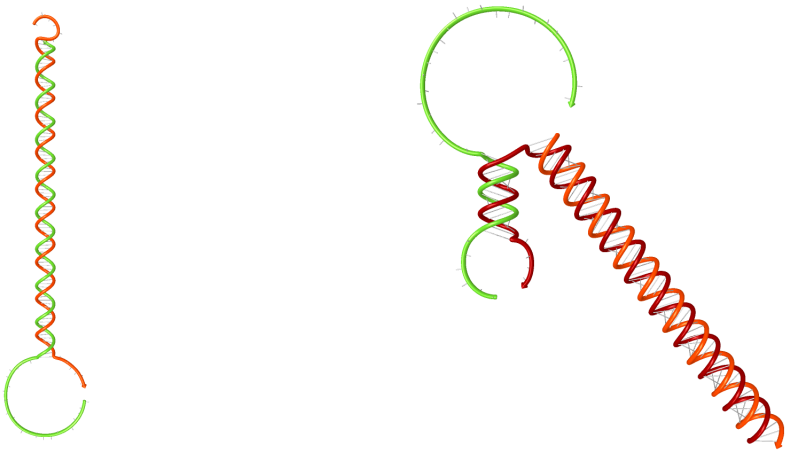
Dominio	Secuencia
t	$5'-$ TATTCC $-3'$
te	$5'-$ GCTA $-3'$
a	$5'-$ CCCTTTACATTACATAACAA $-3'$
b	$5'-$ CCCTTAACTTAACAAATCTA $-3'$
sa	$5'-$ CCCTTTTCTAAACTAAACAA $-3'$
sb	$5'-$ CCCTTATCATATCAATACAA $-3'$
sy	$5'-$ CCCAAAACAAAACAAAACAA $-3'$
y	$5'-$ CCCTATTCAATTCAAATCAA $-3'$

6.3.2. Compuerta OR

En la Tabla 3 se muestran las secuencias pertenecientes a cada uno de los dominios de las cadenas de ADN que se utilizaron durante las simulaciones estocásticas del modelo de la compuerta OR. El complejo principal (Figura 48(a)) está formado por un total de 220 nucleótidos y presenta una energía libre de $-107.68\,kcal/mol$ mientras que el complejo auxiliar (Figura 48(b)) está formado por 94 nucleótidos en total y presenta una energía libre de $-36.48\,kcal/mol$, por lo que ambos son termodinámicamente estables. La Figura 49 muestra la gráfica del promedio de las 10 simulaciones para los cuatro casos de la tabla de verdad de la compuerta OR para 100,000 moléculas de entrada (variables y complejos), dichas simulaciones requirieron 19,500 unidades de tiempo:

- Caso 1 (A = 0, B = 0). En la Figura 49(a) se muestra la simulación del primer caso de la tabla de verdad donde no se encuentran presentes las entradas, por lo que ninguna reacción ocurre y las concentraciones iniciales de cada especie de molécula se mantienen hasta el final de la simulación; tal como se esperaba, la concentración de la variable Y es de 0 moléculas, lo que indica un '0' lógico y un 100 % de fiabilidad para este caso.

- Caso 2 (A = 0, B = 1). El promedio de las simulaciones del caso donde sólo se encuentra presente la variable B se ve representado en la Figura 49(b), donde se observa que dicha variable se va consumiendo al hibridarse con el complejo principal al igual que el complejo auxiliar al liberar la variable Y, aumentando la concentración



(a) (b)

Figura 45: Estructura secundaria de la compuerta AND generada por NUPACK: (a) complejo principal; (b) complejo auxiliar.

de ésta última y la del complejo OR/B. La fiabilidad para este caso es del 99.9 % dado que de las 100,000 moléculas de la variable de salida Y que se esperaban, se obtuvieron 99,978 en promedio.

- Caso 3 (A = 1, B = 0). El comportamiento de la simulación promedio del caso donde sólo está presente la variable A es idéntico al caso anterior y se ve reflejado en la gráfica de la Figura 49(c). La fiabilidad para este caso es del 99.9 % dado que de las 100,000 moléculas de la variable de salida Y que se esperaban, en promedio se obtuvieron 99,979.

- Caso 4 (A = 1, B = 1). La simulación promedio para este caso (Figura 49(d)) llegó a su término antes del tiempo de simulación predeterminado; en ella se puede observar cómo la concentración de la variable Y aumenta a lo largo de la simulación (alcanzando las 100,000 moléculas) y las entradas se consumen junto con los complejos principal y auxiliar. Además, el complejo principal con las dos variables hibridadas aumenta, indicando que las reacciones ocurrieron de la manera esperada. La fiabilidad para este caso es del 100 % dado que de las 100,000 moléculas de la variable de salida Y que se esperaban, se obtuvieron 100,000.

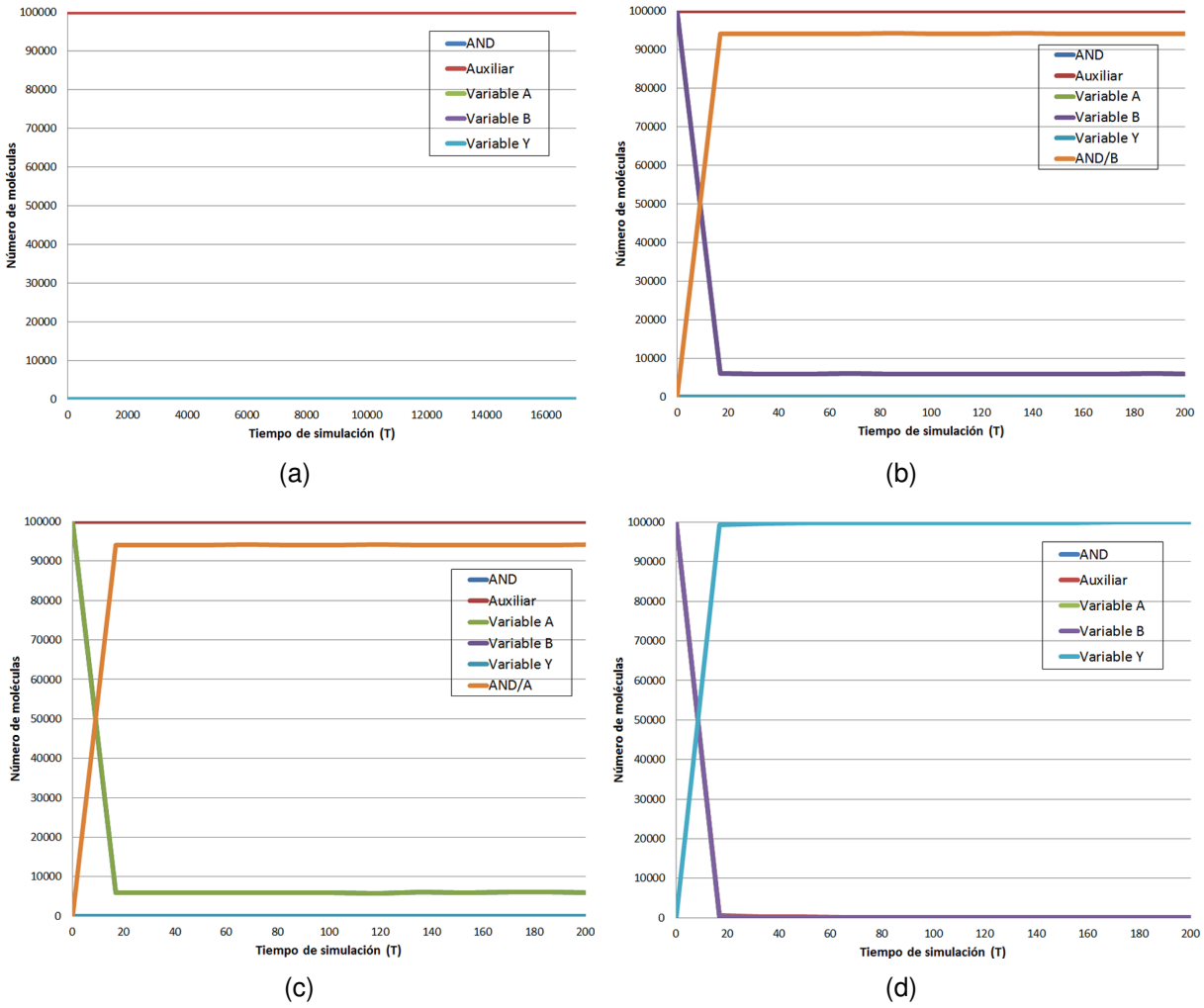


Figura 46: Simulación promedio de la compuerta AND con 100,000 moléculas de entrada: (a) caso 1 (A = 0, B = 0); (b) caso 2 (A = 0, B = 1); (c) caso 3 (A = 1, B = 0); (d) caso 4 (A = 1, B = 1).

Con respecto a las propiedades cuantitativas evaluadas, la probabilidad de alcanzar un estado final existoso eventualmente es de 1, mientras que la probabilidad de alcanzar un estado final erróneo es igual a 0 (Figura 50). De lo anterior se puede inferir que el modelo es estable y funciona de la manera esperada. El tiempo estimado para alcanzar un estado final es de aproximadamente 4,175 unidades de tiempo. En cuanto al análisis del espacio de estados, los resultados obtenidos con Visual DSD y PRISM coincidieron y se pueden consultar en el Apéndice A.2. Así, al analizar los resultados experimentales, se puede concluir que el diseño de la compuerta OR es correcto puesto que el comportamiento emula la operación de la disyunción lógica.

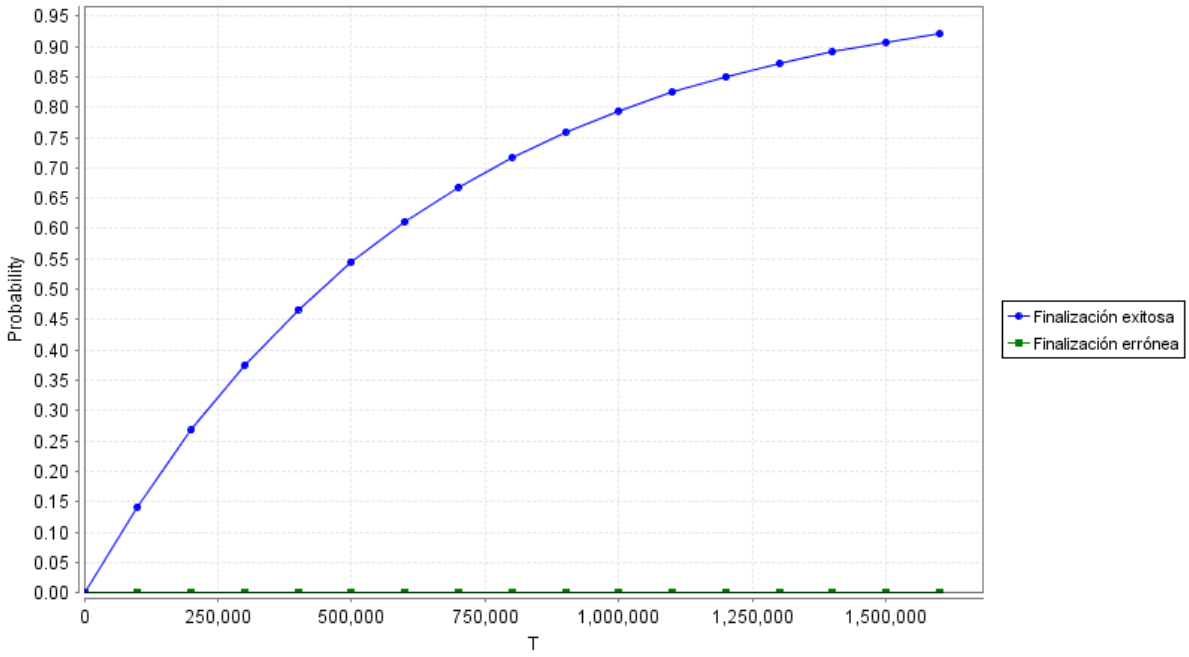


Figura 47: Probabilidad de finalización exitosa vs. errónea del modelo de compuerta AND.

6.3.3. Compuerta NOT

En la Tabla 4 se muestran las secuencias pertenecientes a cada uno de los dominios de las cadenas de ADN que se utilizaron para realizar las simulaciones estocásticas del modelo de la compuerta NOT. El complejo principal (Figura 51(a)) está formado por un total de 118 nucleótidos y presenta una energía libre de $-61.90\,kcal/mol$ mientras que el complejo auxiliar (Figura 51(b)) está formado por 138 nucleótidos en total y presenta una energía libre de $-78.10\,kcal/mol$, por lo que ambos son termodinámicamente estables. La Figura 52 muestra las gráficas del promedio de 10 simulaciones para los dos casos de la tabla de verdad de la compuerta NOT para 100,000 moléculas de entrada. Cada simulación requirió 60,500 unidades de tiempo.

- Caso 1 (A=0). En la Figura 52(a) se muestra la gráfica de la simulación promedio para el caso donde la entrada no está presente (representando un valor de 0). Al no existir dicha variable, no se desencadenan las reacciones necesarias para que se consuma la variable de salida Y, por lo que la concentración de ésta no se modifica, indicando un '1' lógico como salida. Observe que las concentraciones del complejo principal y el complejo auxiliar se mantienen en 100,000 moléculas a lo largo del

Tabla 3: Secuencias correspondientes a cada dominio para las simulaciones estocásticas de la compuerta OR.

Dominio	Secuencia
t	$5'-$ TATTCC $-3'$
te	$5'-$ GCTA $-3'$
a	$5'-$ CCCTTTTCTAAACTAAACAA $-3'$
b	$5'-$ CCCTTAACTTAACAAATCTA $-3'$
sa	$5'-$ CCCAAAACAAAACAAAACAA $-3'$
sb	$5'-$ CCCTTATCATATCAATACAA $-3'$
sy	$5'-$ CCCTATTCAATTCAAATCAA $-3'$
y	$5'-$ CCCTTTACATTACATAACAA $-3'$

tiempo de simulación mientras que la concentración de la variable A (y la del complejo auxiliar/Y) es siempre 0. Con base en lo anterior, la fiabilidad para este caso es del 100 % dado que de las 100,000 moléculas de la variable de salida Y que se esperaban, se obtuvieron 100,000.

- Caso 2 (A=1). La simulación promedio para el caso donde la entrada A está presente (indicando un valor de '1') se muestra en la Figura 52(b). Observe cómo las concentraciones de la variable A, del complejo principal, del complejo auxiliar y de la variable Y (variable de salida) van disminuyendo a lo largo del tiempo de simulación; esto se debe a que la entrada se está consumiendo al reaccionar con la compuerta (complejo NOT/A) y se están llevando a cabo las reacciones correspondientes entre el complejo principal y el auxiliar para atrapar la variable Y, por lo que se puede observar que la concentración de complejo auxiliar/Y aumenta y la concentración de la variable de salida Y dismunuye para indicar un '0' lógico. La fiabilidad para este caso es del 99.3 % dado que de las 100,000 moléculas del complejo auxiliar/Y que se esperaban, se obtuvieron en promedio 99,324 y de las 0 moléculas de la variable de salida Y que se esperaban, se obtuvo un promedio de 676.

Con respecto a las propiedades cuantitativas evaluadas, la probabilidad de alcanzar un estado final existoso eventualmente es de 1 mientras que la probabilidad de alcanzar un estado final erróneo es igual a 0, por lo que se puede deducir que el modelo es estable y funciona de la manera esperada tal y como se puede observar en la Figura 53. El

(a) (b)

Figura 48: Estructura secundaria de la compuerta OR generada por NUPACK: (a) complejo principal; (b) complejo auxiliar.

tiempo estimado para alcanzar un estado final es de aproximadamente 10,044 unidades de tiempo. En cuanto al análisis del espacio de estados, los resultados obtenidos con Visual DSD y PRISM coincidieron y se pueden consultar en el Apéndice A.3. Así, al analizar los resultados experimentales, se puede inferir que el diseño de la compuerta NOT es correcto.

Tabla 4: Secuencias correspondientes a cada dominio para las simulaciones estocásticas de la compuerta NOT.

Dominio	Secuencia
t	$5'-$ TATTCC $-3'$
te	$5'-$ GCTA $-3'$
a	$5'-$ CCCTTTTCTAAACTAAACAA $-3'$
sa	$5'-$ CCCAAAACAAAACAAAACAA $-3'$
sy	$5'-$ CCCTTTACATTACATAACAA $-3'$
y	$5'-$ CCCTTATCATATCAATACAA $-3'$

6.3.4. Compuerta NAND

En la Tabla 5 se muestran las secuencias pertenecientes a cada uno de los dominios de las cadenas de ADN que se utilizaron para realizar las simulaciones estocásticas del modelo de la compuerta NAND. El complejo principal (Figura 54(a)) está formado por un

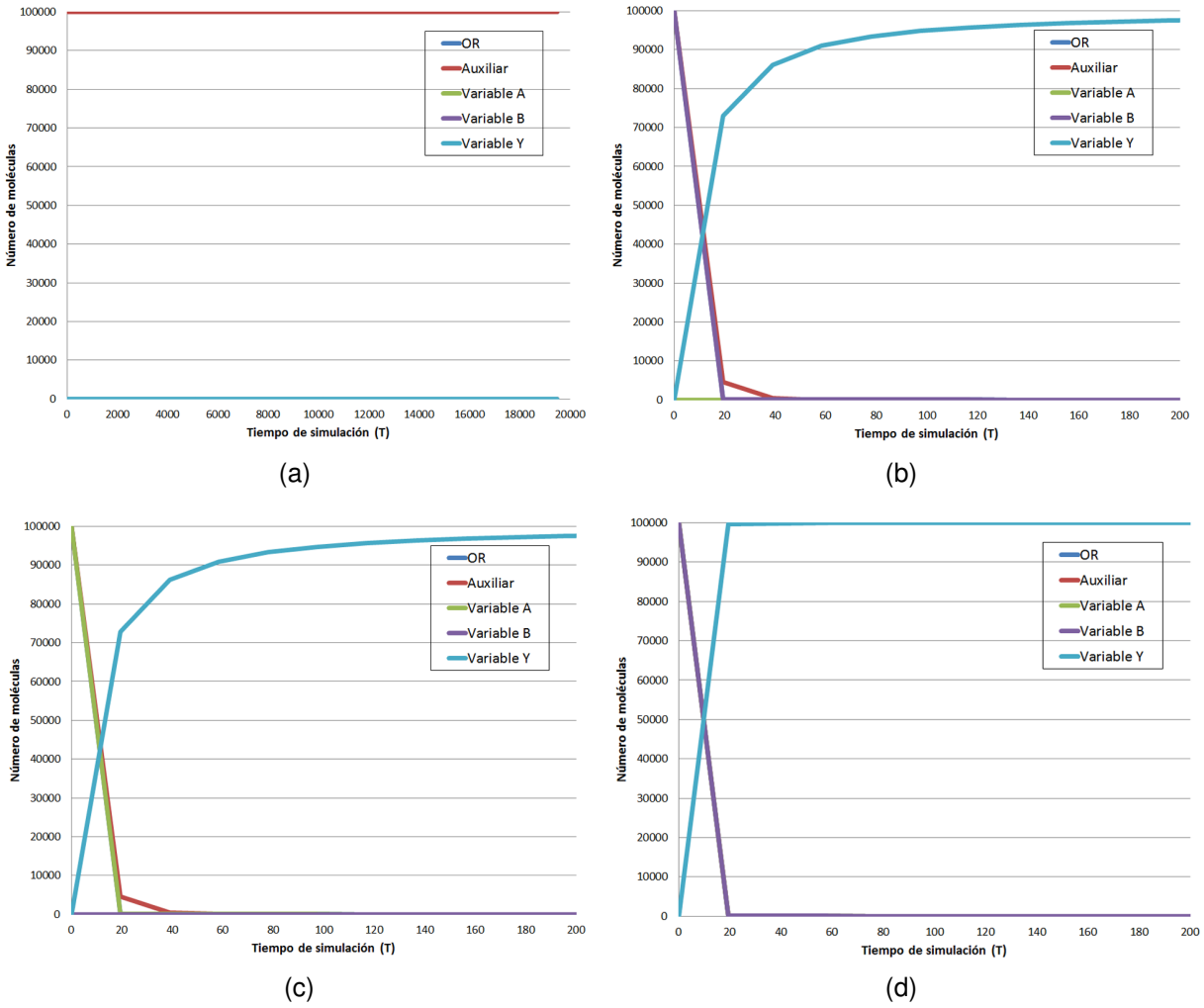


Figura 49: Simulación promedio de la compuerta OR con 100,000 moléculas de entrada: (a) caso 1 (A = 0, B = 0); (b) caso 2 (A = 0, B = 1); (c) caso 3 (A = 1, B = 0); (d) caso 4 (A = 1, B = 1).

total de 204 nucleótidos y presenta una energía libre de $-108.34\,kcal/mol$ mientras que el complejo auxiliar (Figura 54(b)) está formado por 212 nucleótidos en total y presenta una energía libre de $-118.43\,kcal/mol$, por lo que ambos son termodinámicamente estables. La Figura 55 muestra la gráfica del promedio de las 10 simulaciones para los cuatro casos de la tabla de verdad de dicha compuerta para 100,000 moléculas de entrada. En total, las simulaciones requirieron 17,500 unidades de tiempo.

- Caso 1 (A = 0, B = 0). En la Figura 55(a) se puede observar que las concentraciones del complejo principal, del complejo auxiliar y de la variable de salida Y no se alteran, esto debido a que no se encuentran ninguna de las dos variables de entrada (A y B). La fiabilidad para este caso es del 100 % dado que de las 100,000 moléculas de

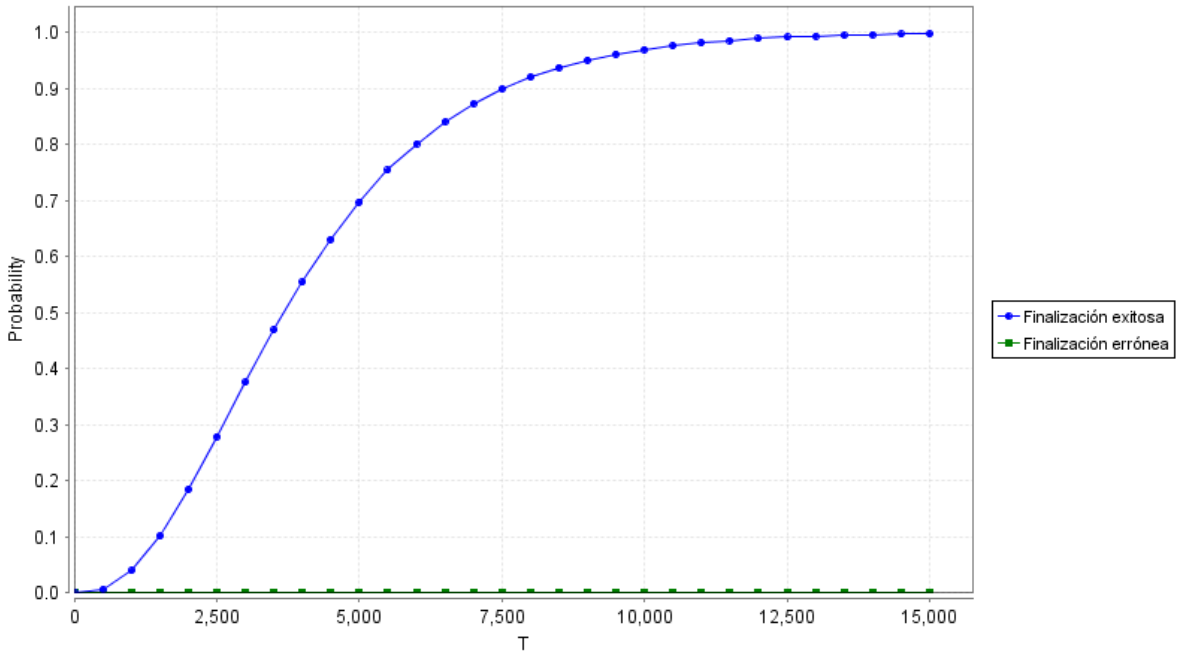


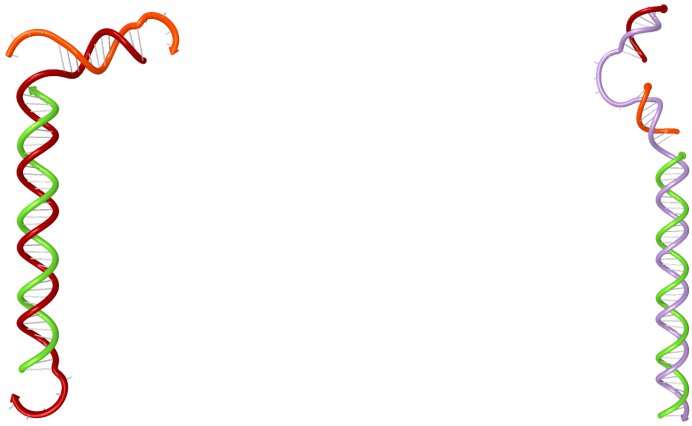
Figura 50: Probabilidad de finalización exitosa vs. errónea del modelo de compuerta OR.

la variable de salida Y que se esperaban, se obtuvieron 100,000.

- Caso 2 (A = 0, B = 1). La simulación promedio de cuando sólo se tiene la entrada B se muestra en la Figura 55(b), donde se puede apreciar cómo esta variable se va consumiendo a lo largo del tiempo de simulación al hibridarse con el complejo principal (por lo que disminuye la concentración de la compuerta), aumentando la concentración del complejo NAND/B. La concentración de la variable Y no se altera ya que siempre se mantiene en 100,000 moléculas, por lo que la fiabilidad para este caso es del 100 %.

- Caso 3 (A = 1, B = 0). En la Figura 55(c) se tiene la simulación promedio del caso donde sólo está presente la variable A, cuyo comportamiento es exactamente el mismo que el caso anteriormente descrito (cuando sólo está B), salvo que ahora la variable A es la que desencadena las reacciones, por lo que la fiabilidad es también del 100 %.

- Caso 4 (A = 1, B = 1). La simulación promedio de cuando ambas variables están presentes se muestra en la Figura 55(d). Aquí se puede observar cómo las variables de entrada se van consumiendo al hibridarse con el complejo principal mientras que



(a) (b)

Figura 51: Estructura secundaria de la compuerta NOT generada por NUPACK: (a) compuerta principal; (b) complejo auxiliar.

la concentración del complejo auxiliar disminuye al hibridar la variable Y con éste. La variable Y disminuye así a lo largo de la simulación, indicando un '0' lógico. La fiabilidad para este caso es del 99.9 % dado que de las 0 moléculas de la variable de salida Y que se esperaban, se obtuvieron 48 y de las 100,000 moléculas de complejo auxiliar/Y se obtuvieron 99,952.

Con respecto a las propiedades cuantitativas evaluadas, tal como se puede observar en la Figura 56, la probabilidad de alcanzar un estado final existoso eventualmente es de 1 mientras que la probabilidad de alcanzar un estado final erróneo es igual a 0, por lo que se puede inferir que el modelo es estable y funciona de la manera esperada. El tiempo estimado para alcanzar un estado final es de aproximadamente 13,1267 unidades de tiempo. En cuanto al análisis del espacio de estados, los resultados obtenidos con Visual DSD y PRISM coincidieron y se pueden consultar en el Apéndice A.4. Al analizar las gráficas de los cuatro casos de la tabla de verdad y de las probabilidades de finalización del modelo, se verificó que el modelo propuesto para la compuerta NAND se comporta de la manera esperada.

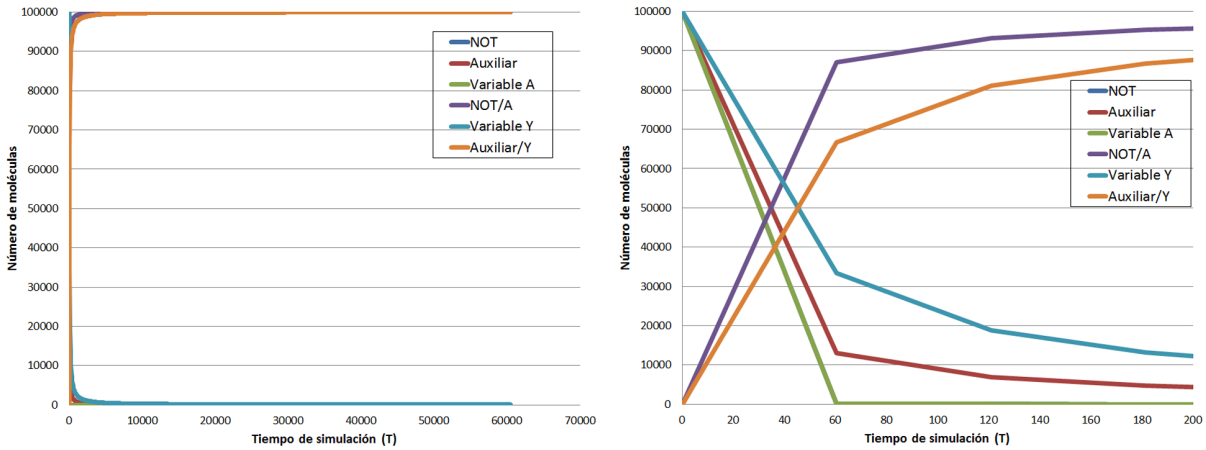


Figura 52: Simulación promedio de la compuerta NOT con 100,000 moléculas de entrada: (a) caso 1 (A = 0) y (b) caso 2 (A = 1).

Tabla 5: Secuencias correspondientes a cada dominio para las simulaciones estocásticas de la compuerta NAND.

Dominio	Secuencia
t	$5'-$ TATTCC $-3'$
te	$5'-$ GCTA $-3'$
a	$5'-$ CCCTTTTCTAAACTAAACAA $-3'$
b	$5'-$ CCCTTATCATATCAATACAA $-3'$
sa	$5'-$ CCCAAAACAAAACAAAACAA $-3'$
sb	$5'-$ CCCTTTACATTACATAACAA $-3'$
sy	$5'-$ CCCTTAACTTAACAAATCTA $-3'$
y	$5'-$ CCCTATTCAATTCAAATCAA $-3'$

6.3.5. Compuerta NOR

En la Tabla 6 se muestran las secuencias pertenecientes a cada uno de los dominios de las cadenas de ADN que se utilizaron para realizar las simulaciones estocásticas del modelo de la compuerta NOR. El complejo principal (Figura 57(a)) está formado por un total de 204 nucleótidos y presenta una energía libre de $-108.04\,kcal/mol$ mientras el complejo auxiliar (Figura 57(b)) está formado por 172 nucleótidos en total y presenta una energía libre de $-94.71\,kcal/mol$, por lo que ambos son termodinámicamente estables. La Figura 58 muestra la gráfica del promedio de las 10 simulaciones para los cuatro casos de la tabla de verdad de dicha compuerta para 100,000 moléculas de entrada. Cada una de las simulaciones requirió 70,000 unidades de tiempo.

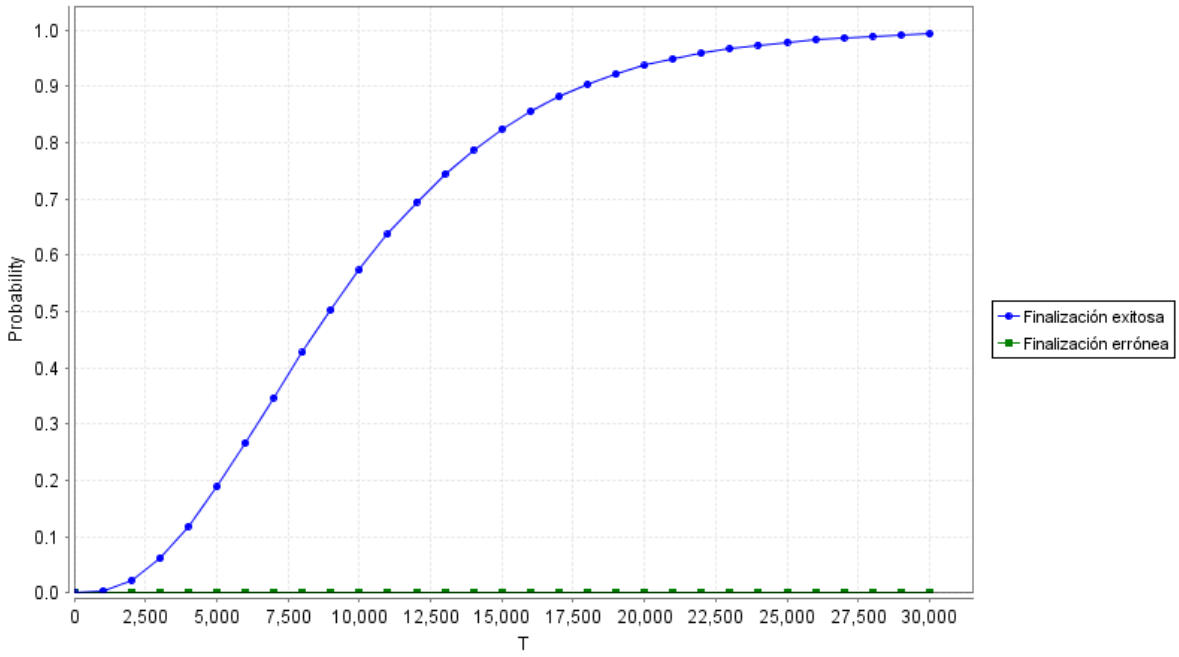
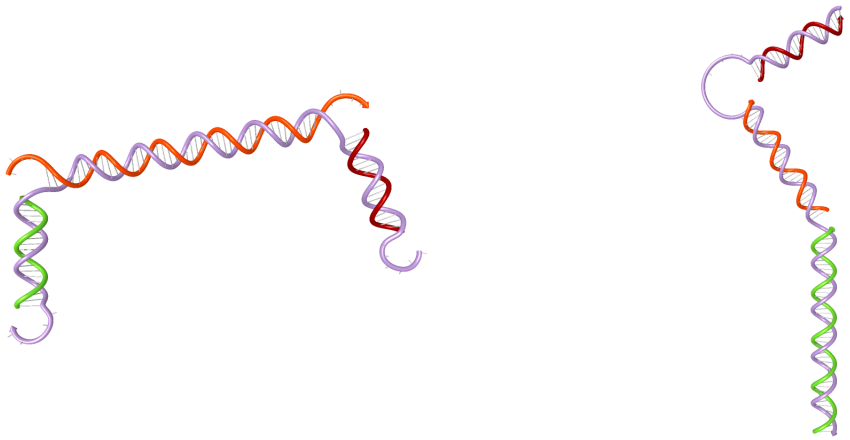


Figura 53: Probabilidad de finalización exitosa vs. errónea del modelo de compuerta NOT.

- Caso 1 (A = 0, B = 0). En la Figura 58(a) se muestra la simulación promedio del primer caso de la tabla de verdad donde se puede apreciar que las reacciones no ocurren y las concentraciones iniciales de cada especie de molécula se mantienen hasta el final de la simulación. Al final de la simulación, la concentración de la variable Y es de 100,000 moléculas, lo que indica un '1' lógico y una fiabilidad del 100 %.

- Caso 2 (A = 0, B = 1). El promedio de las simulaciones del caso donde sólo se encuentra la variable B se ve representado en la Figura 58(b). Observe que la variable B se consume al hibridarse con el complejo principal, al igual que el complejo auxiliar al hibridarse con la variable Y, disminuyendo la concentración de ésta última y aumentando la del complejo NOR/B. La fiabilidad para este caso es del 99.9 % dado que de las 0 moléculas de la variable de salida Y que se esperaban, se obtuvieron 25 y de las 100,000 moléculas del complejo auxiliar/Y se obtuvieron 99,975.

- Caso 3 (A = 1, B = 0). El comportamiento de la simulación del caso donde sólo está presente la variable A es idéntico al caso anterior y se ve reflejado en la gráfica de la Figura 58(c). La fiabilidad para este caso también es del 99.9 % dado que de



(a) (b)

Figura 54: Estructura secundaria de la compuerta NAND generada por NUPACK: (a) complejo principal; (b) complejo auxiliar.

las 0 moléculas de la variable de salida Y que se esperaban, se obtuvieron 25 y de las 100,000 moléculas del complejo auxiliar/Y se obtuvieron 99,975.

- Caso 4 (A = 1, B = 1). La simulación promedio con las dos entradas presentes (Figura 58(d)) muestra que la concentración de la variable Y disminuyó a lo largo de la simulación (indicando un '0' lógico) y las entradas se consumieron junto con los complejos compuerta y auxiliar. También se puede observar que la concentración del complejo principal con las dos variables hibridadas aumentó, indicando que las reacciones ocurrieron de la forma esperada. La fiabilidad para este caso es del 99.9 % dado que de las 0 moléculas de la variable de salida Y que se esperaban, se obtuvieron 13 y de las 100,000 moléculas del complejo auxiliar/Y se obtuvieron 99,987.

En lo que respecta al análisis de propiedades, la probabilidad de alcanzar un estado final existoso eventualmente es de 1 mientras que la probabilidad de alcanzar un estado final erróneo es igual a 0 (tal como se puede ver en la Figura 59), por lo que se puede inferir que el modelo es estable y funciona de la manera esperada. El tiempo esperado para alcanzar un estado final es de aproximadamente 6,970 unidades de tiempo. En cuanto al análisis del espacio de estados, los resultados obtenidos con Visual DSD y PRISM

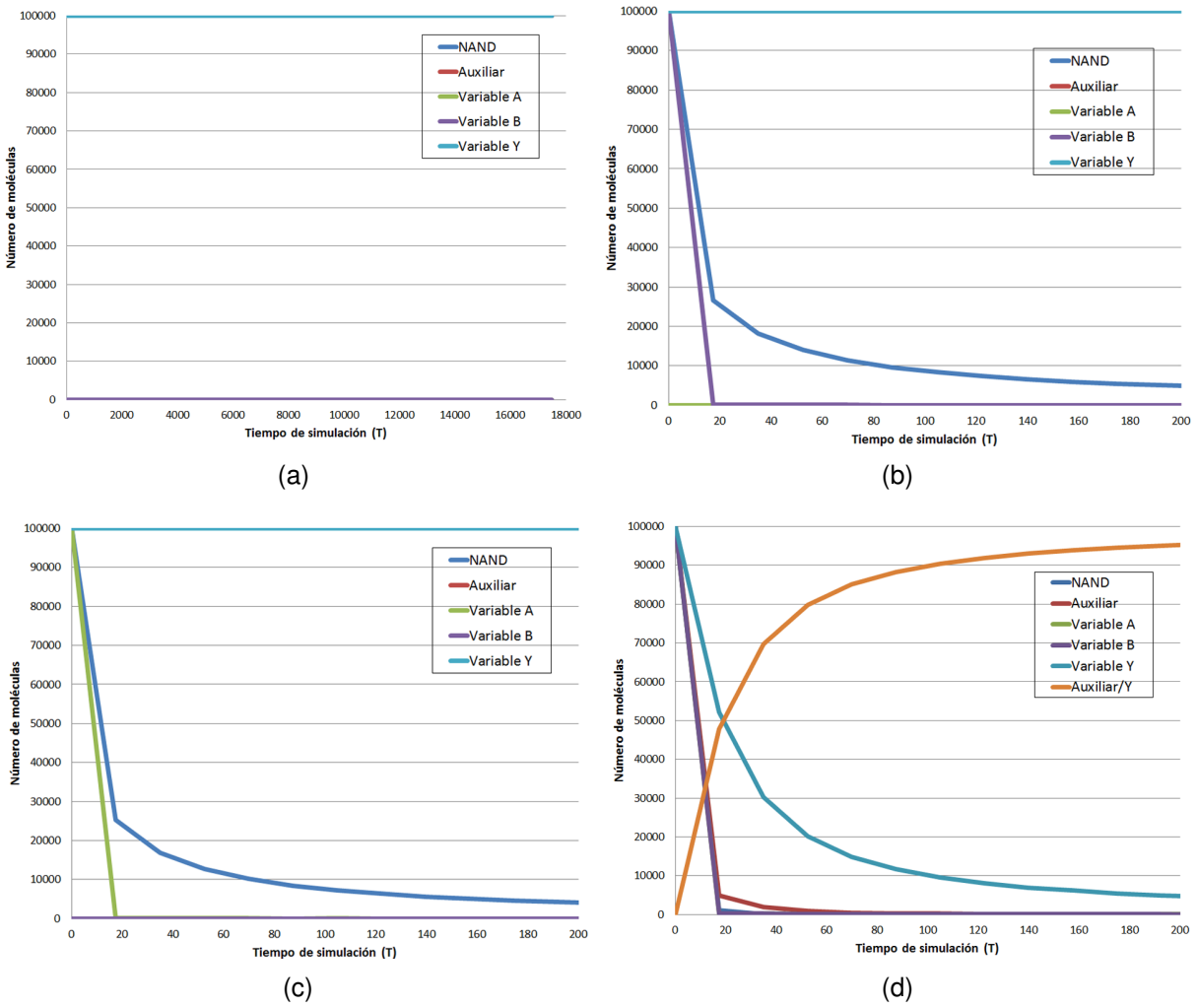


Figura 55: Simulación promedio de la compuerta NAND con 100,000 moléculas de entrada: (a) caso 1 (A = 0, B = 0); (b) caso 2 (A = 0, B = 1); (c) caso 3 (A = 1, B = 0); (d) caso 4 (A = 1, B = 1).

coincidieron y se pueden consultar en el Apéndice A.5. Así, al analizar los resultados experimentales, se concluyó que el diseño de la compuerta NOR es correcto.

6.3.6. Compuerta XOR

En la Tabla 7 se muestran las secuencias pertenecientes a cada uno de los dominios de las cadenas de ADN que se utilizaron para realizar las simulaciones estocásticas del modelo de la compuerta XOR. El complejo principal (Figura 60(a)) está formado por un total de 348 nucleótidos y presenta una energía libre de $-167.52\,kcal/mol$ mientras que el complejo auxiliar (Figura 60(b)) está formado por 304 nucleótidos en total y presenta una energía libre de $-151.64\,kcal/mol$, por lo que ambos son termodinámicamente estables.

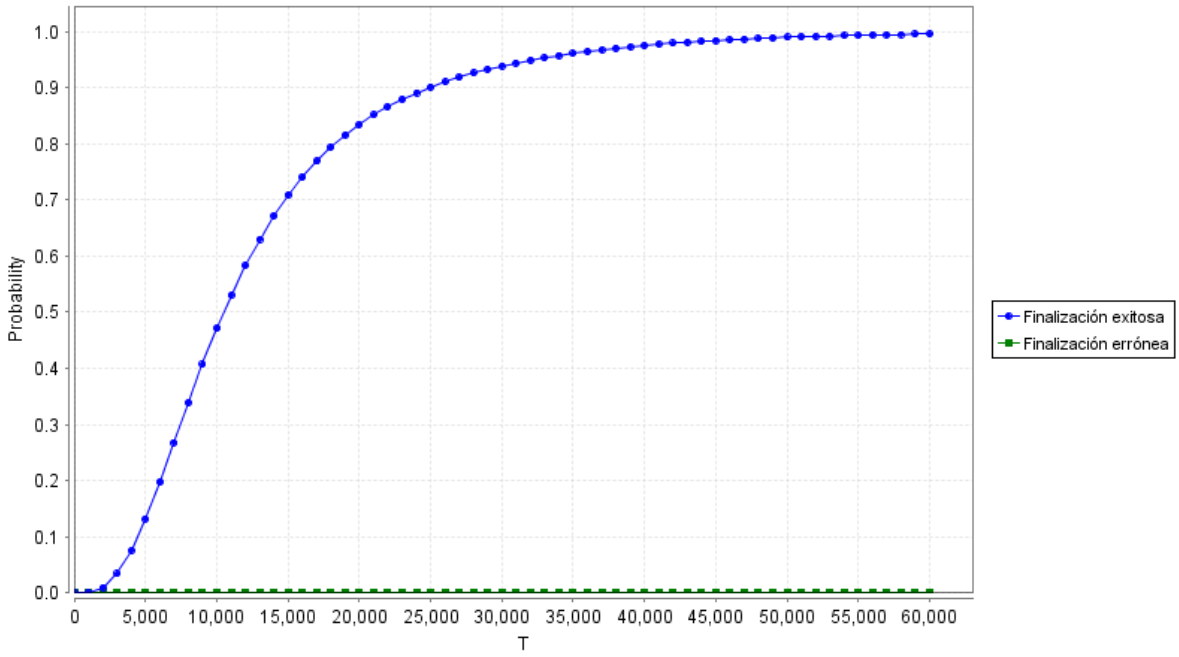


Figura 56: Probabilidad de finalización exitosa vs. errónea del modelo de compuerta NAND.

La Figura 61 muestra la gráfica del promedio de 10 simulaciones para los cuatro casos de la tabla de verdad de esta compuerta para 100,000 moléculas de entrada, dichas simulaciones requirieron 8,500 unidades de tiempo:

- Caso 1 (A = 0, B = 0). En la Figura 61(a) se muestra la simulación promedio del primer caso de la tabla de verdad donde no se encuentran las entradas presentes. Consecuentemente, ninguna reacción ocurre y las concentraciones iniciales de cada especie de molécula se mantienen hasta el final de la simulación; la concentración de la variable Y es de 0 moléculas (indicando un '0' lógico) por lo que la fiabilidad es del 100 % para este caso.

- Caso 2 (A = 0, B = 1). El promedio de las simulaciones del caso donde sólo se encuentra la variable B se muestra en la Figura 61(b). Note que la variable B se va consumiendo al hibridarse con el complejo principal de la compuerta XOR, al igual que la del complejo auxiliar al liberar la variable Y, aumentando la concentración de ésta última al igual que la del complejo XOR/B. La fiabilidad para este caso es del 99.7 % dado que de las 100,000 moléculas de la variable de salida Y que se esperaban, se obtuvieron 99,748.

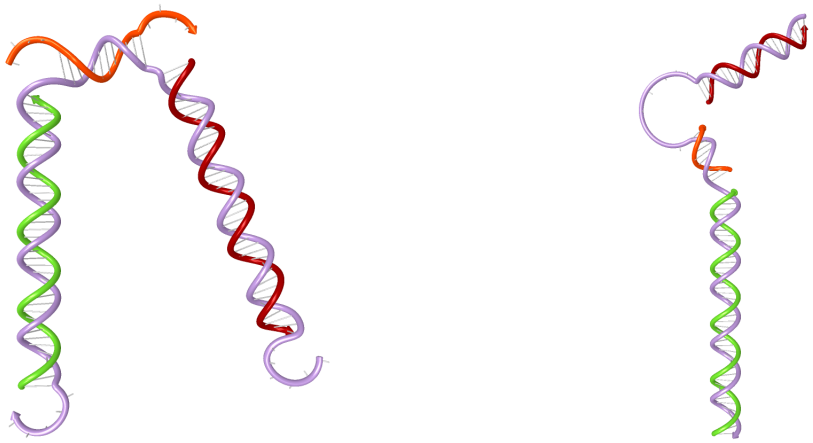
Tabla 6: Secuencias correspondientes a cada dominio para las simulaciones estocásticas de la compuerta NOR.

Dominio	Secuencia
t	$5'-$ TATTCC $-3'$
te	$5'-$ GCTA $-3'$
a	$5'-$ CCCTTTTCTAAACTAAACAA $-3'$
b	$5'-$ CCCTTATCATATCAATACAA $-3'$
sa	$5'-$ CCCAAAACAAAACAAAACAA $-3'$
sb	$5'-$ CCCTTTACATTACATAACAA $-3'$
sy	$5'-$ CCCTTAACTTAACAAATCTA $-3'$
y	$5'-$ CCCTATTCAATTCAAATCAA $-3'$

- Caso 3 (A = 1, B = 0). El comportamiento de la simulación promedio de este caso se presenta en la gráfica de la Figura 61(c). La fiabilidad obtenida es del 99.8 % dado que de las 100,000 moléculas de la variable de salida Y que se esperaban, se obtuvieron 99,887.

- Caso 4 (A = 1, B = 1). La simulación promedio con las dos entradas presentes (Figura 61(d)) muestra que la concentración de la variable Y disminuyó a lo largo de la simulación para indicar un '0' lógico y las entradas se consumieron junto con los complejos principal y auxiliar. También se puede observar que la concentración del complejo XOR/A y B así como la del complejo auxiliar/Y aumentaron, indicando que las reacciones ocurrieron tal como se esperaba. La fiabilidad para este caso es del 99.8 % dado que de las 0 moléculas de la variable de salida Y que se esperaban, se obtuvieron 104 y de las 100,000 moléculas del complejo auxiliar/Y se obtuvieron 99,897 en promedio.

Con respecto al análisis de propiedades, la probabilidad de alcanzar un estado final existoso eventualmente es de 1 mientras que la probabilidad de alcanzar un estado final erróneo es igual a 0, tal como se puede observar en la Figura 62, por lo que se puede inferir que el modelo es estable y funciona de la manera esperada. El tiempo estimado para alcanzar un estado final es de aproximadamente 15,317 unidades de tiempo. En cuanto al análisis del espacio de estados, los resultados obtenidos con Visual DSD y PRISM coincidieron y se pueden consultar en el Apéndice A.6. Al analizar las gráficas de



(a) (b)

Figura 57: Representación de la estructura secundaria del modelo de compuerta NOR: (a) complejo principal; (b) complejo auxiliar.

los promedios de las simulaciones para los cuatro casos de la tabla de verdad y de probabilidades de finalización del modelo, se puede concluir que el diseño de la compuerta XOR es correcto puesto que el comportamiento emula la operación del OR exclusivo.

Tabla 7: Secuencias correspondientes a cada dominio para las simulaciones estocásticas de la compuerta XOR.

Dominio	Secuencia
t	$5'-$ TATTCC $-3'$
ta	$5'-$ GCTA $-3'$
te	$5'-$ GTCA $-3'$
a	$5'-$ CCCTTTTCTAAACTAAACAA $-3'$
b	$5'-$ CCCTTATCATATCAATACAA $-3'$
sa	$5'-$ CCCAAAACAAAACAAAACAA $-3'$
sb	$5'-$ CCCTTTACATTACATAACAA $-3'$
sy	$5'-$ CCCTATTCAATTCAAATCAA $-3'$
y	$5'-$ CCCTTAACTTAACAAATCTA $-3'$

6.3.7. Compuerta XNOR

En la Tabla 8 se muestran las secuencias pertenecientes a cada uno de los dominios de las cadenas de ADN que se utilizaron en las simulaciones estocásticas del modelo de la compuerta XNOR. El complejo principal (Figura 63(a)) está formado por un total

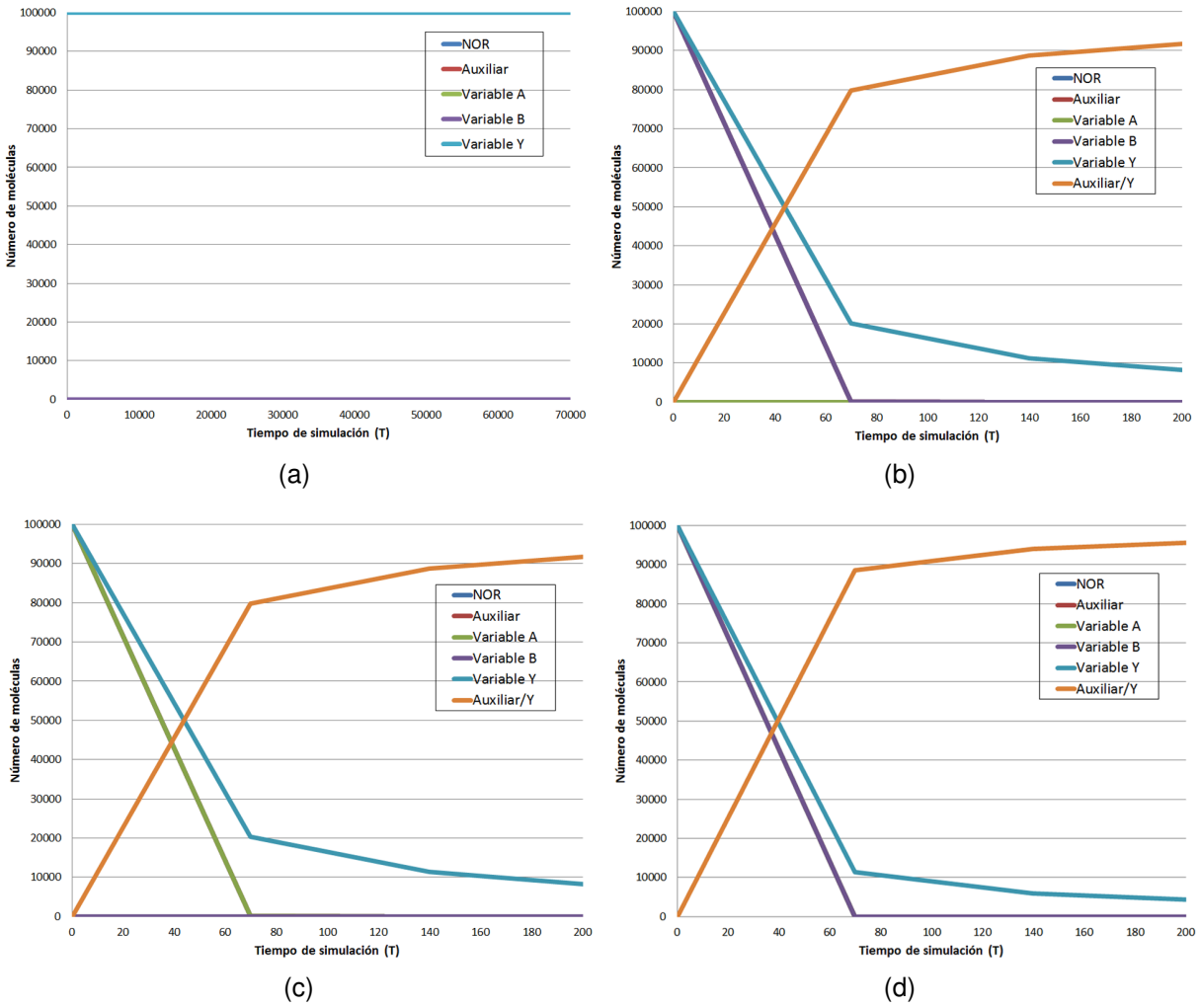


Figura 58: Simulación promedio de la compuerta NOR con 100,000 moléculas de entrada: (a) caso 1 (A = 0, B = 0); (b) caso 2 (A = 0, B = 1); (c) caso 3 (A = 1, B = 0); (d) caso 4 (A = 1, B = 1).

de 404 nucleótidos y presenta una energía libre de $-204.98\,kcal/mol$ mientras que el complejo auxiliar (Figura 63(b)) está formado por 408 nucleótidos en total y presenta una energía libre de $-212.37\,kcal/mol$, por lo que ambos son termodinámicamente estables. La Figura 64 muestra la gráfica del promedio de 10 simulaciones para los cuatro casos de la tabla de verdad de la compuerta XNOR para 100,000 moléculas de entrada, cada simulación requirió 9,500 unidades de tiempo:

- Caso 1 (A = 0, B = 0). En la Figura 64(a) se muestra la simulación promedio del primer caso de la tabla de verdad donde no se encuentran las entradas presentes por lo que no ocurren reacciones y las concentraciones iniciales de cada especie de molécula se mantienen hasta el final de la simulación. La concentración de la varia-

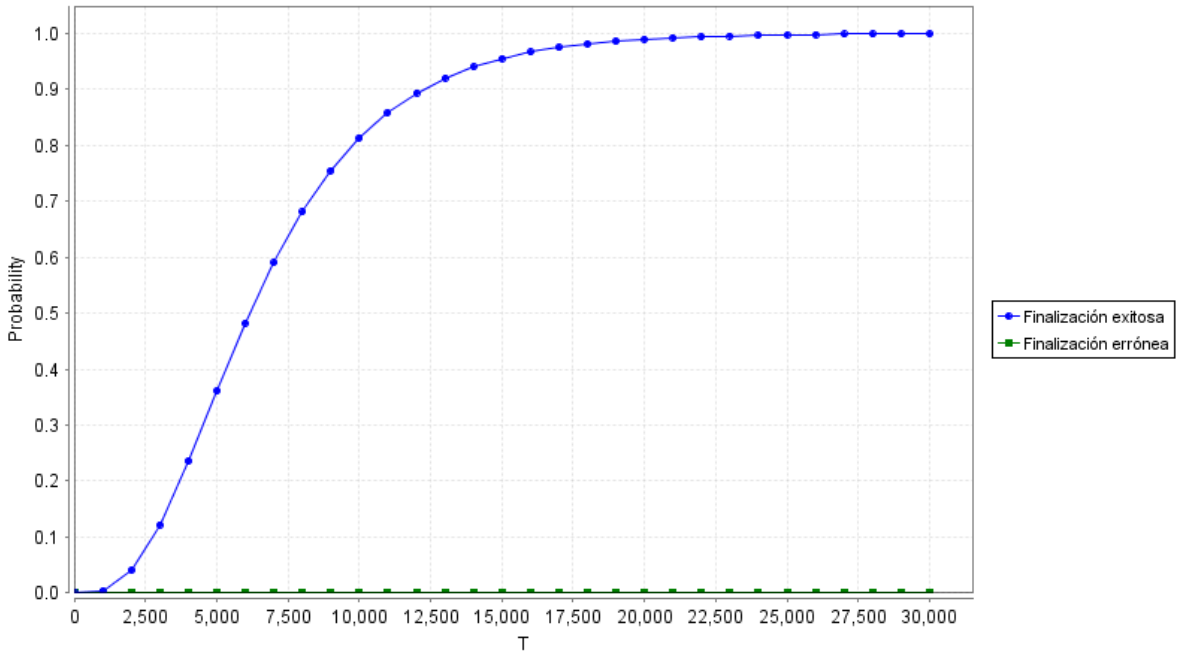
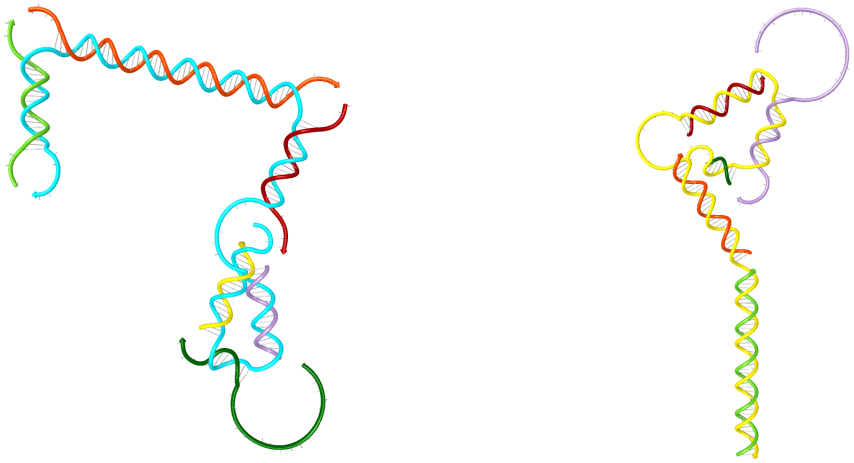


Figura 59: Probabilidad de finalización exitosa vs. errónea del modelo de compuerta NOR.



(a) (b)

Figura 60: Estructura secundaria de la compuerta XOR generada por NUPACK: (a) complejo principal; (b) complejo auxiliar.

ble Y es de 100,000 moléculas, lo que indica un '1' lógico y un 100 % de fiabilidad.

- Caso 2 (A = 0, B = 1). El promedio de las simulaciones del caso donde sólo se encuentra la variable B se ve representado en la Figura 64(b). La variable B se va consumiendo al hibridarse con el complejo principal, al igual que el complejo auxiliar

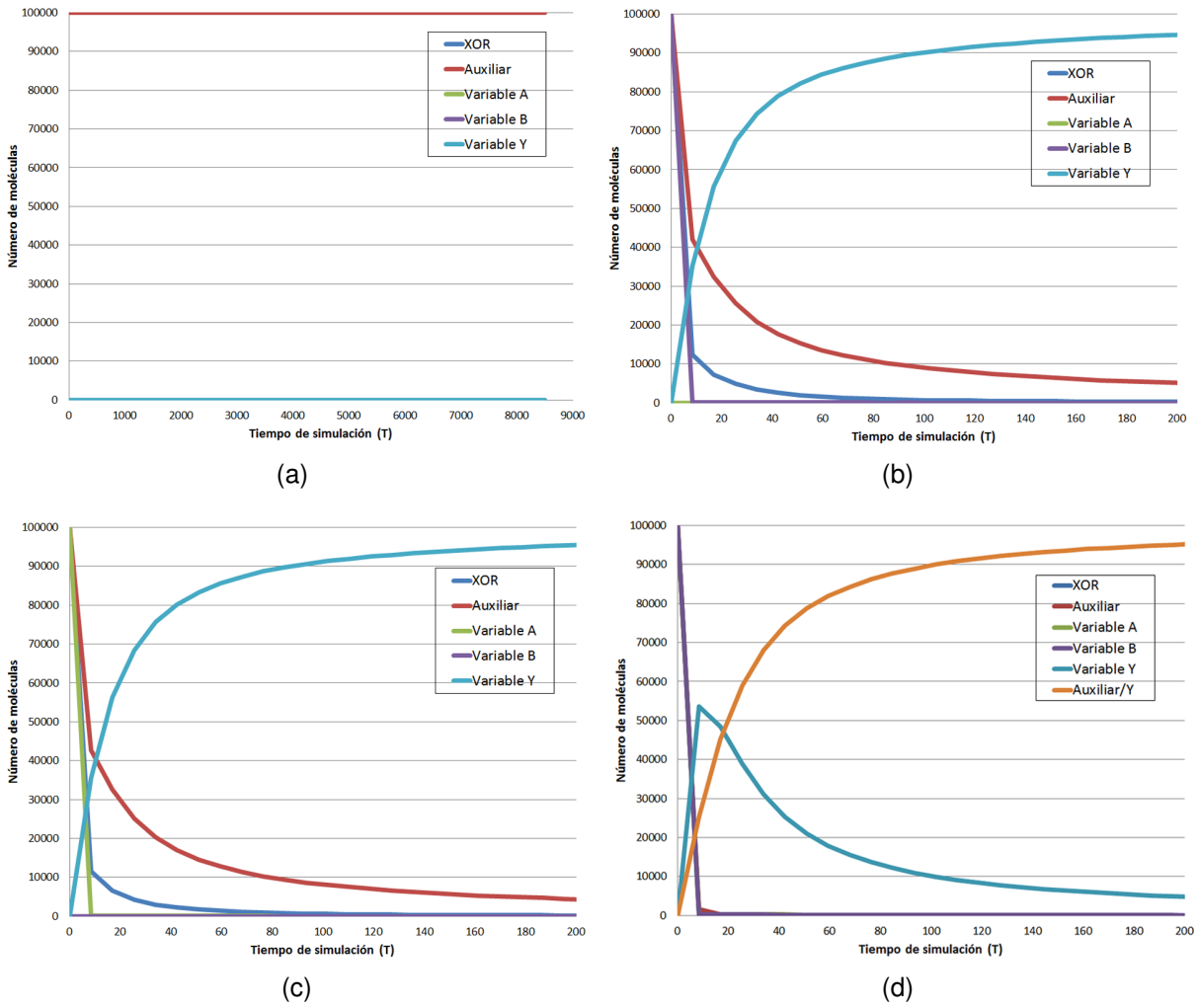


Figura 61: Simulación promedio de la compuerta XOR con 100,000 moléculas de entrada: (a) caso 1 (A = 0, B = 0); (b) caso 2 (A = 0, B = 1); (c) caso 3 (A = 1, B = 0); (d) caso 4 (A = 1, B = 1).

al hibridarse con la variable Y, disminuyendo la concentración de ésta última y aumentando la del complejo XNOR/B. La fiabilidad para este caso es del 99.8 % dado que de las 0 moléculas de la variable de salida Y que se esperaban, se obtuvieron 190 y de las 100,000 moléculas del complejo auxiliar/Y se obtuvieron 99,810.

- Caso 3 (A = 1, B = 0). El comportamiento de la simulación promedio de este caso es idéntico al caso anterior y se ve reflejado en la gráfica de la Figura 64(c). La fiabilidad para este caso es del 99.8 % dado que de las 0 moléculas de la variable de salida Y que se esperaban, se obtuvieron 190 y de las 100,000 moléculas del complejo auxiliar/Y se obtuvieron 99,910.

- Caso 4 (A = 1, B = 1). La simulación promedio con las dos entradas presentes

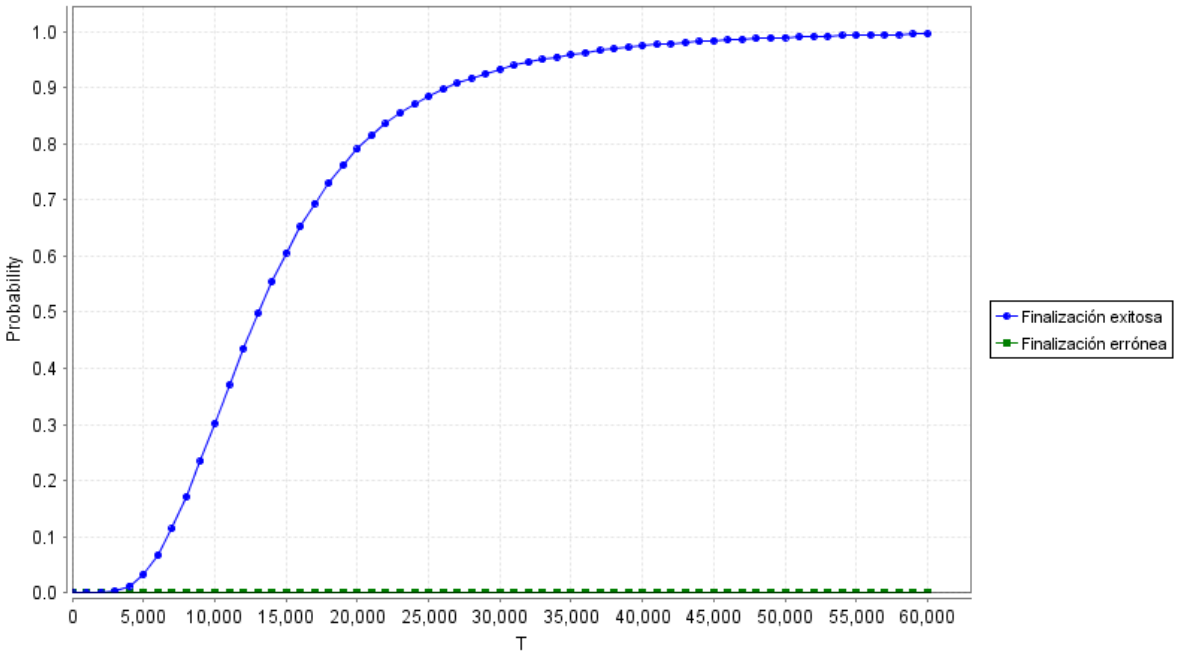


Figura 62: Probabilidad de finalización exitosa vs. errónea del modelo de compuerta XOR.

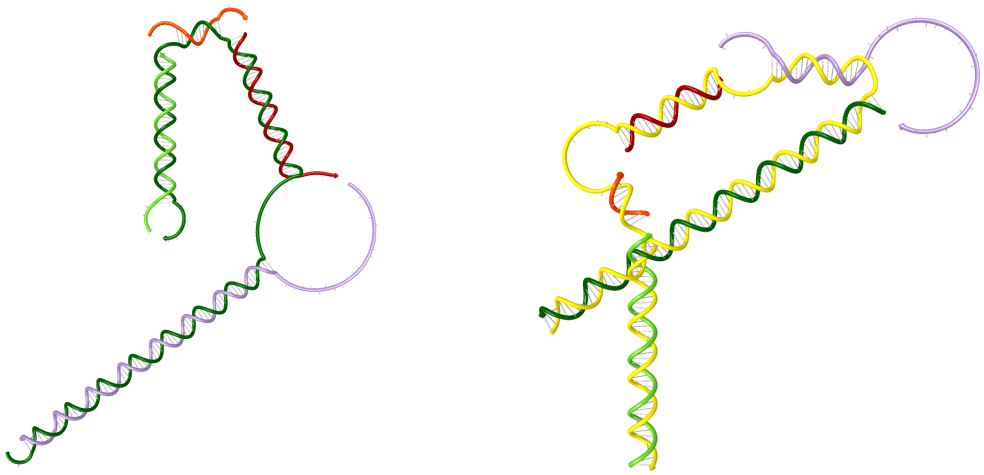
(Figura 64(d)) muestra que la concentración de la variable Y aumentó a lo largo de la simulación e incluso durante unos instantes de tiempo alcanzó un pico de 140,000 moléculas (debido al diseño de la compuerta); sin embargo, dicha concentración disminuyó hasta las 100,000 moléculas para indicar un '1' lógico como salida. Además, las entradas se consumieron junto con los complejos principal y auxiliar mientras que la concentración del complejo XNOR/A y B aumentó, por lo que el modelo se comportó como se esperaba. La fiabilidad para este caso es del 99.9 % dado que de las 100,000 moléculas de la variable de salida Y que se esperaban, se obtuvieron 99,994.

Con respecto al análisis de propiedades, la probabilidad de alcanzar un estado final existoso eventualmente es de 1 mientras que la probabilidad de alcanzar un estado final erróneo es igual a 0 (Figura 65). De aquí se puede inferir que el modelo es estable y funciona de la manera esperada. El tiempo estimado para alcanzar un estado final es de aproximadamente 636,98 unidades de tiempo. En cuanto al análisis del espacio de estados, los resultados obtenidos con Visual DSD y PRISM coincidieron y se pueden consultar en el Apéndice A.7. En las gráficas de los cuatro casos promedio de las simulaciones y de

probabilidades de finalización del modelo se puede observar que los resultados coinciden con los indicados en la tabla de verdad, lo que permitió verificar la correctitud del diseño de la compuerta XNOR.

Tabla 8: Secuencias correspondientes a cada dominio para las simulaciones estocásticas de la compuerta XNOR.

Dominio	Secuencia
t	$5'-$ TATTCC $-3'$
ta	$5'-$ GCTA $-3'$
te	$5'-$ GTCA $-3'$
a	$5'-$ CCCTTTTCTAAACTAAACAA $-3'$
b	$5'-$ CCCTTATCATATCAATACAA $-3'$
sa	$5'-$ CCCAAAACAAAACAAAACAA $-3'$
sb	$5'-$ CCCTTTACATTACATAACAA $-3'$
sy	$5'-$ CCCTTAACTTAACAAATCTA $-3'$
y	$5'-$ CCCTATTCAATTCAAATCAA $-3'$



(a) (b)

Figura 63: Estructura secundaria de la compuerta XNOR generada por NUPACK: (a) complejo principal; (b) complejo auxiliar.

En resumen, los resultados obtenidos en las diferentes simulaciones de los siete modelos de compuertas lógicas propuestos fueron favorables y las propiedades cuantitivas de los modelos se verificaron de manera exitosa. En cuestiones termodinámicas, se puede decir que todos los complejos son estables. Con respecto a los resultados de las

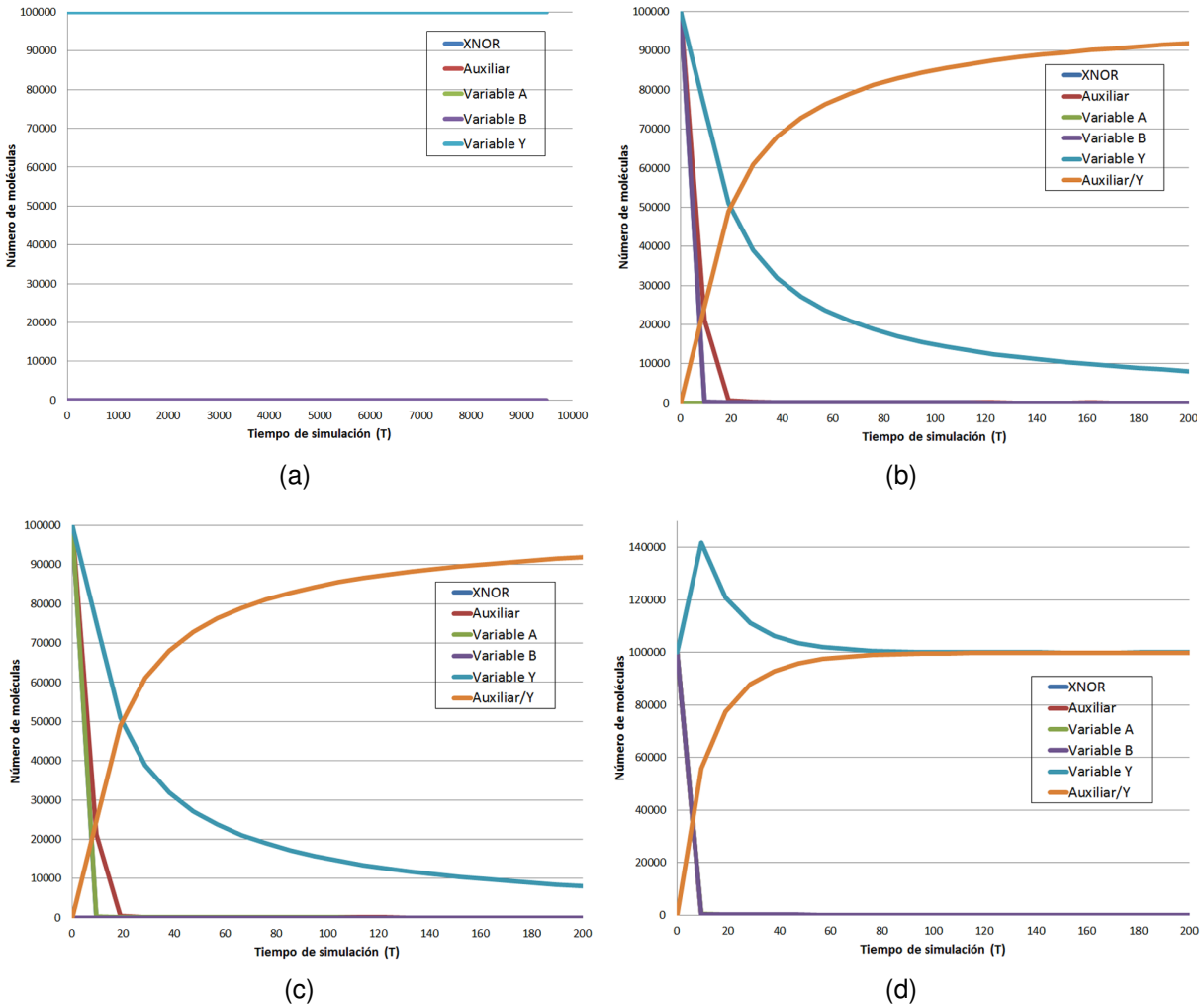


Figura 64: Simulación promedio de la compuerta XNOR con 100,000 moléculas de entrada: (a) caso 1 (A = 0, B = 0); (b) caso 2 (A = 0, B = 1); (c) caso 3 (A = 1, B = 0); (d) caso 4 (A = 1, B = 1).

simulaciones estocásticas, se pudo observar que todas las compuertas presentan una fiabilidad mayor al 98 %.

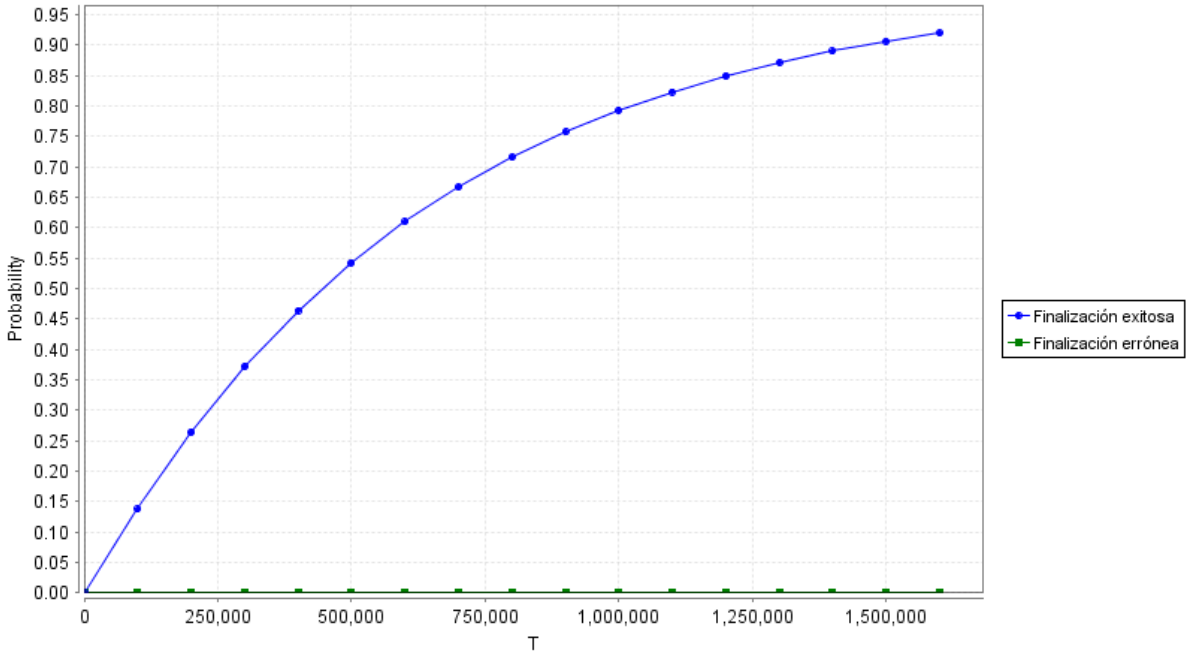


Figura 65: Probabilidad de finalización exitosa vs. errónea del modelo de compuerta XNOR.

Capítulo 7. Aplicaciones conceptuales de compuertas lógicas

Tal como se mencionó en capítulos anteriores, el ADN es un material ideal para la realización de operaciones de cómputo ya que admite la construcción de algunos dispositivos tales como compuertas lógicas y circuitos. Esto se debe principalmente a que su naturaleza biológica soporta aplicaciones tecnológicas *in vivo*, su síntesis química es sencilla y facilita experimentos prácticos *in vitro*, además de que su estructura combinatoria proporciona suficiente espacio para el diseño de las secuencias y el principio de complementariedad de Watson-Crick permite un comportamiento molecular predecible. El diseño de sistemas lógicos moleculares, aparte de ser reconfigurable, es aplicable en áreas de considerable importancia como ciencias de la vida o medicina, ya que su tamaño tan pequeño admite la detección y/o reconocimiento de marcadores moleculares especialmente en el medio intracelular, lo que eventualmente permitiría la realización de diagnósticos inteligentes. En este capítulo se describen algunas aplicaciones conceptuales para los diseños de compuertas lógicas propuestos.

7.1. Detección de células de mama cancerosas

Algunos miARNs tienen la particularidad de comportarse como oncogenes, éstos promueven o inhiben el crecimiento de tumores al sobre-expresarse o sub-expresarse según sea el caso. Estos marcadores moleculares se pueden detectar mediante microarreglos y PCR cuantitativo o en tiempo real, entre otras técnicas. Las concentraciones tisulares de microARNs específicos se han asociado a invasividad tumoral, potencial metastásico y otras características clínicas de varios tipos de cáncer como leucemia linfocítica crónica, de mama, colorrectal, hepático, de pulmón, pancreático y de próstata, entre otros. Los perfiles de expresión de microARNs son útiles para la clasificación, diagnosis y prognosis de tumores malignos (Jain, 2010). Además, los miARNs circulantes que se encuentran en fluidos corporales (sangre, orina, saliva, etc.) al ser estables, detectables y cuantificables, funcionan como marcadores no invasivos en pacientes con enfermedades como el cáncer. Con base en lo anterior, es posible formular una expresión booleana compleja cuyas entradas o variables correspondan a marcadores de enfermedad (miARNs y mARN),

de tal forma que la resolución de dicha expresión indique el diagnóstico (positivo o negativo) de alguna patología, por lo que también se le puede denominar como regla de diagnóstico molecular. Kahan-Hanum *et al.* (2013) proponen una expresión de este tipo para determinar si una célula mamaria es cancerosa, donde las variables a considerar son los microARNs miR31, miR21, miR125b y el ARNm c-myc:

$$\begin{aligned} \textit{célula de mama cancerosa} \; = & (miR31 \; \textbf{AND} \; miR21) \; \textbf{OR} \\ & (miR31 \; \textbf{ANDNOT} \; miR125b) \; \textbf{OR} \; (c-myc). \end{aligned} \tag{21}$$

Básicamente, esta regla indica que una célula mamaria es cancerosa si se cumple al menos una de las siguientes condiciones:

1) Existe una combinación de la presencia de un microARN específico de cáncer de mama (miR31) y un microARN canceroso (miR21).
2) Existe la presencia de un microARN específico de cáncer de mama (miR31) y el microARN indicativo de salud (miR125b) está ausente.
3) Un marcador de ARNm canceroso (c-myc) u oncogén está presente.

Se decidió resolver dicha expresión con el modelo de compuertas lógicas propuesto para verificar el funcionamiento de las compuertas en cascada y al mismo tiempo ejemplificar una aplicación potencial para la biblioteca de compuertas. Para realizar esto, es necesario representar cada uno de los marcadores moleculares con una variable distinta, reescribiendo la expresión de la siguiente manera:

$$\textit{célula de mama cancerosa}=(A \; \textbf{AND} \; B) \; \textbf{OR} \; (A \; \textbf{ANDNOT} \; C) \; \textbf{OR} \; (D). \tag{22}$$

donde A = miR31, B = miR21, C = miR125b y D = c-myc. Antes de resolver la expresión booleana con el modelo de compuertas lógicas propuesto es necesario transformar cada uno de los marcadores moleculares en variables de entrada que sigan la convención propuesta; para ello se requieren compuertas traductoras, cuyo esquema se ilustra en la Figura 66 y su diseño está basado en la propuesta de Seelig *et al.* (2006). Una compuerta traductora debe contener la secuencia complementaria al marcador molecular en la

cadena inferior y una hebra de protección que corresponda a la variable que éste representará en el sistema (variable X). De este modo, la cadena de entrada (miARN) desplaza a la cadena correspondiente a la variable X al hibridar con la hebra inferior completamente complementaria de la compuerta traductora.

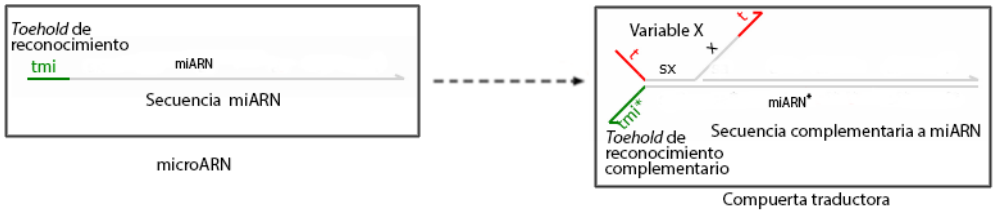


Figura 66: Esquema de una compuerta traductora.

En la Figura 67(a) se puede observar el circuito lógico equivalente a la expresión booleana, donde la entradas (variables A, B, C y D) representan los marcadores moleculares y la variable de salida (I) corresponde al diagnóstico de la muestra. En la Figura 67(b) se puede observar el circuito lógico en términos del modelo propuesto. El circuito se enumera por etapas para hacer más fácil su seguimiento: si se divide la expresión booleana (Ecuación 22) en 3 cláusulas separadas por la operación OR, la primera cláusula (de izquierda a derecha) se resuelve en la etapa 1 del circuito, la operación ANDNOT se ve representada en las etapas 2 y 3 del circuito. La etapa 4 resuelve la operación OR entre las dos primeras cláusulas y finalmente, la etapa 5 corresponde a la operación OR de los resultados de la etapa 4 con la tercera cláusula, que es donde se obtiene el valor de salida del circuito. La tabla de verdad para la expresión booleana corresponde a la Tabla 9, donde la columna diagnóstico nos indica si la célula de mama es cancerosa, siendo '1' si éste es positivo y '0' de lo contrario.

Para ejemplificar la resolución de la expresión con las compuertas lógicas propuestas se toma como ejemplo el caso 2 de la tabla de verdad (Tabla 9), donde sólo se encuentran presentes los marcadores miR31 y miR21 (representados por las variables A y B, respectivamente). El valor de salida esperado es un '1' lógico el cual se representa con la presencia de la variable I y la emisión de fluorescencia por parte del complejo detector indicando un diagnóstico positivo. En la Figura 69 se presenta la resolución esquemática de este caso. La primera etapa corresponde a la compuerta AND cuyas entradas son las

Tabla 9: Tabla de verdad correspondiente a la expresión booleana para detección de células de mama cancerosas Ecuación 22.

miR31 (variable A)	miR21 (variable B)	miR125b (variable C)	c-myc (variable D)	Diagnóstico (variable I)
1	1	1	0	1
1	**1**	**0**	**0**	**1**
0	1	1	0	0
0	1	0	0	0
1	0	1	0	0
1	0	0	0	1
0	0	1	0	0
0	0	0	0	0
1	1	1	1	1
1	1	0	1	1
0	1	1	1	1
0	1	0	1	1
1	0	1	1	1
1	0	0	1	1
0	0	1	1	1
0	0	0	1	1

variables A y B, esperando la variable E como salida, ya que la operación resulta en un ‘1’ lógico. La segunda etapa comprende la compuerta NOT cuya entrada (variable C) no está presente, por lo que el resultado de esta etapa es un ‘1’ lógico, es decir, la presencia de la variable F. La tercer etapa consiste en la operación AND con las entradas A y F, ambas presentes, dando como resultado un ‘1’ lógico y por consiguiente la presencia de la variable G. La cuarta etapa corresponde a la compuerta OR con las entradas E y G. Dado que ambas están presentes, se obtiene la presencia de la variable H indicando un ‘1’ lógico. La quinta etapa corresponde a la compuerta OR con la entrada D (este marcador no está presente) y la entrada H (liberada en la etapa anterior). En una compuerta OR basta que una de las dos variables de entrada esté presente para que la compuerta regrese un ‘1’ lógico, por lo que se libera la variable de salida I, la cual es la variable final del circuito. El último paso del modelo consiste en la detección de dicha variable por medio del complejo detector: la variable I hibrida con el complejo detector separando la hebra de protección que contiene el fluoróforo emitiendo fluorescencia e indicando un diagnótico positivo (‘1’ lógico) coincidiendo así el resultado con el caso 2 de la tabla de verdad (Tabla 9).

Tabla 10: Secuencias correspondientes a cada dominio para el diseño del circuito de detección de células cancerosas.

Dominio	Secuencia
t	$5'-$ TATTCC $-3'$
tea1	$5'-$ GCTA $-3'$
tea2	$5'-$ TACCAA $-3'$
ten	$5'-$ GTCA $-3'$
teo1	$5'-$ CATCG $-3'$
teo2	$5'-$ ACTACAC $-3'$
a	$5'-$ CCCTTTACATTACATAACAA $-3'$
b	$5'-$ CCCTTAACTTAACAAATCTA $-3'$
c	$5'-$ CCCTAATCTAATCATAACTA $-3'$
d	$5'-$ CCTATACCTTAACTTAACAA $-3'$
e	$5'-$ CCCTATTCAATTCAAATCAA $-3'$
f	$5'-$ CCCATTTCAAATCAAAACTT $-3'$
g	$5'-$ CCCATATCTATACATTACAA $-3'$
h	$5'-$ CCCAATTCTTAACATATCAA $-3'$
i	$5'-$ CCCAATACTATTCATAACAT $-3'$
sa	$5'-$ CCCTTTTCTAAACTAAACAA $-3'$
sb	$5'-$ CCCTTATCATATCAATACAA $-3'$
sc	$5'-$ CCCTATACTATACAATACTA $-3'$
sd	$5'-$ CCCAAATCTTAACTATACTA $-3'$
se	$5'-$ CCCAAAACAAAACAAAACAA $-3'$
sf	$5'-$ CCCTAAACTTATCTAAACAT $-3'$
sg	$5'-$ CCCATTACTAATCAATTCAA $-3'$
sh	$5'-$ CCCATAACTATTCTAAACTA $-3'$
si	$5'-$ CCATATCCATAACTTTACAA $-3'$

La simulación estocástica de este ejemplo con la herramienta Visual DSD permitió verificar el funcionamiento y desempeño del modelo propuesto. Es importante mencionar que ésta es una simulación computacional ideal, ya que en una situación real no se conocen las concentraciones de los miARNs dentro de una muestra, por lo que no existiría la misma concentración de moléculas de entrada y compuertas, y por consiguiente tampoco de moléculas de salida. Tomando en cuenta lo anterior, para realizar una simulación más realista de esta aplicación de diagnóstico sería necesario refinar su diseño, por lo que se plantea como trabajo a futuro o como un posible campo de oportunidad. Para la simulación realizada se utilizaron los siguientes parámetros: 100,000 moléculas de entrada de cada una de las diferentes especies, salvo de la variable A, cuya concentración se duplicó (200,000 moléculas) debido a que funciona como entrada para dos etapas

del circuito. El tiempo de simulación fue de 6,000 unidades de tiempo con 1,000 puntos de muestra. Las secuencias de nucleótidos utilizadas para cada uno de los dominios se muestran en la Tabla 10. La Figura 68 muestra la gráfica de las primeras 2,000 unidades de tiempo de simulación del caso 2 de la tabla de verdad (Tabla 9), ya que en este lapso es donde ocurren la mayoría de las reacciones y permite apreciar mejor el comportamiento de las moléculas; en ella se puede observar cómo se consumen las entradas iniciales A y B (que representan los marcadores). También se puede observar que la concentración de la variable de salida I aumenta para después disminuir a causa de su hibridación con el complejo detector, liberando así la hebra de protección de este complejo que contiene el fluoróforo e indicando un diagnóstico positivo, siendo acorde con la tabla de verdad. En la Tabla 11 se muestran los resultados de la simulación estocástica con el número final de moléculas de las entradas, salidas y las compuertas con sus correspondientes entradas hibridadas. En nuestra simulación, idealmente el resultado esperado es de 100,000 moléculas de la hebra con el fluoróforo, obteniéndose un total de 99,970 moléculas, por lo que se puede decir que la fiabilidad del circuito diseñado con el modelo propuesto para este caso es del 99.9 %.

Tabla 11: Resultados de la simulación del caso 2 de la tabla de verdad del circuito para detección de células de mama cancerosas.

Molécula	Concentración final
Variable A	13
Variable B	7
Variable I	1
Hebra con fluoróforo	99,970
AND CON A y B	99,993
AND CON A y F	99,994
OR con E y G	99,985
OR con H	99,984
NOT	100,000

7.2. Sumador completo

Como se mencionó anteriormente, todas las operaciones aritméticas que se realizan en las computadoras se llevan a cabo en términos de lógica. Un sumador es un circuito que calcula la operación de la suma empleando el sistema binario y en las computadoras

convencionales dicho circuito se encuentra en la unidad aritmético lógica. Para sumar dos cadenas binarias se necesita una serie de sumadores completos tan larga como la longitud de las cadenas que se desean sumar. El acarreo de cada sumador completo debe ir hacia la tercera entrada del siguiente sumador completo. El primero de los sumadores sólo requiere dos entradas, ya que no existen pasos previos por lo que tampoco existen acarreos de operaciones anteriores; el acarreo del último de los sumadores indica "desbordamiento", esto es, que el número es demasiado grande para ser manipulado por el circuito.

El circuito de un sumador completo tiene tres entradas: un bit A del cosumando, un bit B del sumando y un bit C_{ENT} de acarreo. El sumador además produce dos salidas: un bit de acarreo (C_{SAL}) y uno de suma (S). Este circuito está formado por un medio sumador que pasa su suma a un segundo medio sumador con los dos dígitos de acarreo recibidos por una compuerta OR de tres entradas. Un medio sumador está compuesto por una compuerta AND y una XOR en paralelo para procesar los dos bits de entrada de modo que la salida de la compuerta AND sea el dígito de acarreo y la salida de la XOR sea el dígito de la suma. Por consiguiente, el sumador completo está compuesto por seis compuertas lógicas: dos XOR, tres AND y una OR de tres entradas. Debido a que existen dos salidas, la circuitería de cada una se diseña de manera individual. La expresión para salida S se define como sigue:

$$S = A \oplus [B \oplus C_{ENT}]. \tag{23}$$

Considerando ahora la salida C_{SAL}, su expresión simplificada corresponde a:

$$C_{SAL} = BC_{ENT} + AC_{ENT} + AB. \tag{24}$$

Las expresiones 23 y 24 se representan con el circuito de la Figura 70 y su tabla de verdad (Tabla 12) tiene ocho casos que representan las posibles combinaciones entre las tres entradas.

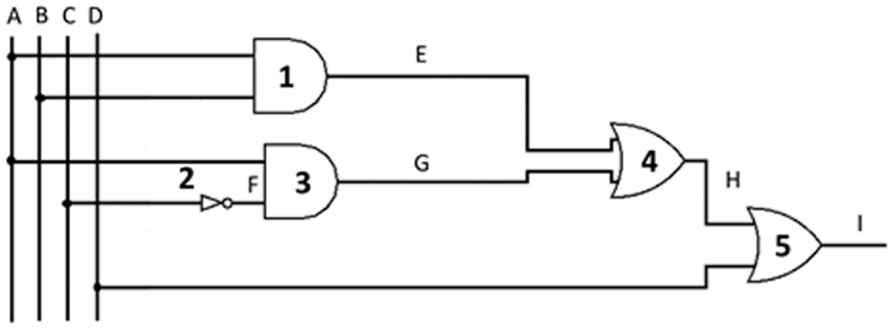
Para probar la robustez del modelo de compuertas propuesto, se decidió implementar un circuito sumador completo cuyo esquema se puede apreciar en la Figura 70(b). El cir-

Tabla 12: Tabla de verdad para un circuito sumador completo.

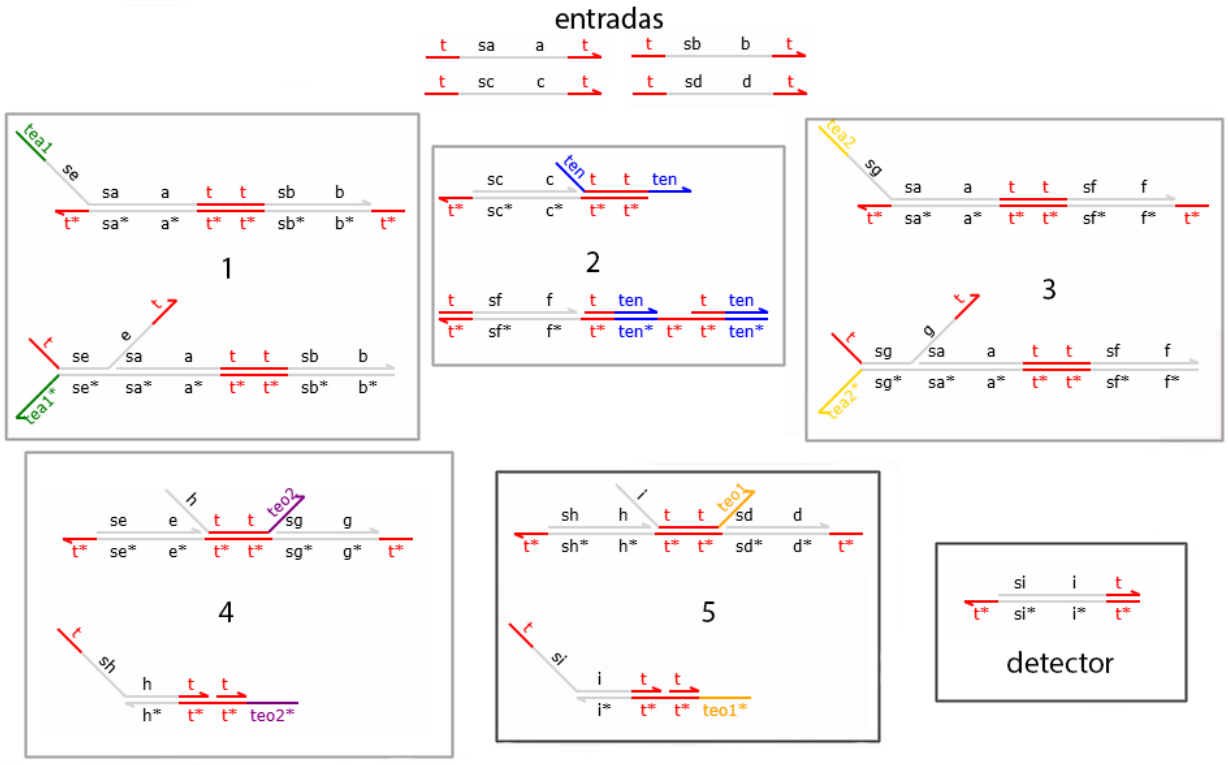
A	B	$C_{\{ENT\}}$	$C_{\{SAL\}}$	S
0	0	0	0	0
0	**0**	**1**	**0**	**1**
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

cuito (Figura 70) se dividió en 7 etapas que corresponden a cada una de las operaciones realizadas por las 7 compuertas que lo componen. Observe que el circuito original se compone de 6 compuertas; sin embargo, es necesario utilizar dos compuertas OR en el diseño propuesto puesto que no se cuenta con una compuerta OR de tres entradas. En la Figura 71 se pueden observar las etapas de resolución del caso 2 de la tabla de verdad del circuito, donde sólo está presente la entrada C_{ENT} y se espera como salida solamente la variable S. La primera etapa consiste en la operación de la compuerta XOR cuyas entradas son las variables C_{ENT} (presente) y A (ausente), por lo que se espera la variable D como salida ya que la operación de OR exclusivo da como resultado un '1' lógico. En la segunda etapa se tiene la operación AND con las variables de entrada A y B (ninguna de las dos variables están presentes) por lo que se espera la ausencia de la variable de salida E. Como tercer paso se tiene la compuerta AND con las entradas C_{ENT} y A. Dado que sólo la primera está presente, no se espera la variable de salida F. La cuarta etapa corresponde a la compuerta AND, cuyas entradas son las variables C_{ENT} (presente) y B (ausente), por lo que la variable de salida G debe estar ausente. En la quinta etapa se tiene una compuerta XOR con las variables de entrada D y B, dado que sólo se encuentra presente la variable D (resultado de la etapa 1) se libera la variable de salida S, indicando un '1' lógico. En la sexta etapa se realiza la operación OR con las variables de entrada E y F, pero como ninguna está presente no se libera la salida H. La séptima etapa corresponde a la compuerta OR con las entradas H y G, que tampoco están presentes, por lo que la salida C_{SAL} no se libera (indicando un '0' lógico) coincidiendo de esta manera con la tabla de verdad. La simulación estocástica de este ejemplo con la herramienta Visual DSD

permitió verificar el funcionamiento y demostrar el desempeño del modelo propuesto. Los parámetros de la simulación fueron: 100,000 moléculas de cada una de las diferentes compuertas involucradas en el circuito, así como 300,000 moléculas de la variable C_{ENT}. La concentración de esta molécula se triplicó debido a que funciona como entrada para tres etapas del circuito. El tiempo de simulación fue de 1,000 unidades y se utilizaron 1,000 puntos de muestra; las secuencias de nucleótidos utilizadas para cada uno de los dominios se muestran en la Tabla 13. La Figura 72 muestra la gráfica de la simulación del caso 2 de la tabla de verdad del circuito sumador completo. Dado que el comportamiento individual de cada compuerta se comprobó en el capítulo anterior, solamente se muestra el comportamiento de las variables de entrada (A = 0, B = 0, C_{ENT} = 1) y las variables de salida (S y C_{SAL}). En la gráfica se puede observar cómo se consume la entrada C_{ENT} y aumenta la concentración de la variable de salida S indicando un '1' lógico para la salida S y un '0' lógico para C_{SAL}. En la Tabla 14 se muestran los resultados de la simulación estocástica con el número final de moléculas de la entrada C_{ENT}, de las compuertas y de las salidas C_{SAL}. Dado que el resultado esperado es de 100,000 moléculas de la variable de salida S y 0 moléculas de la variable C_{SAL} y se obtuvieron 99,743 moléculas de la primera y 0 moléculas de la segunda, se puede decir que la fiabilidad del circuito diseñado con el modelo propuesto es del 99.7 % para este caso en particular.



(a)



(b)

Figura 67: Circuito lógico para la resolución de la regla de diagnóstico: (a) circuito lógico que implementa la Ecuación 22 y (b) circuito bioquímico equivalente.

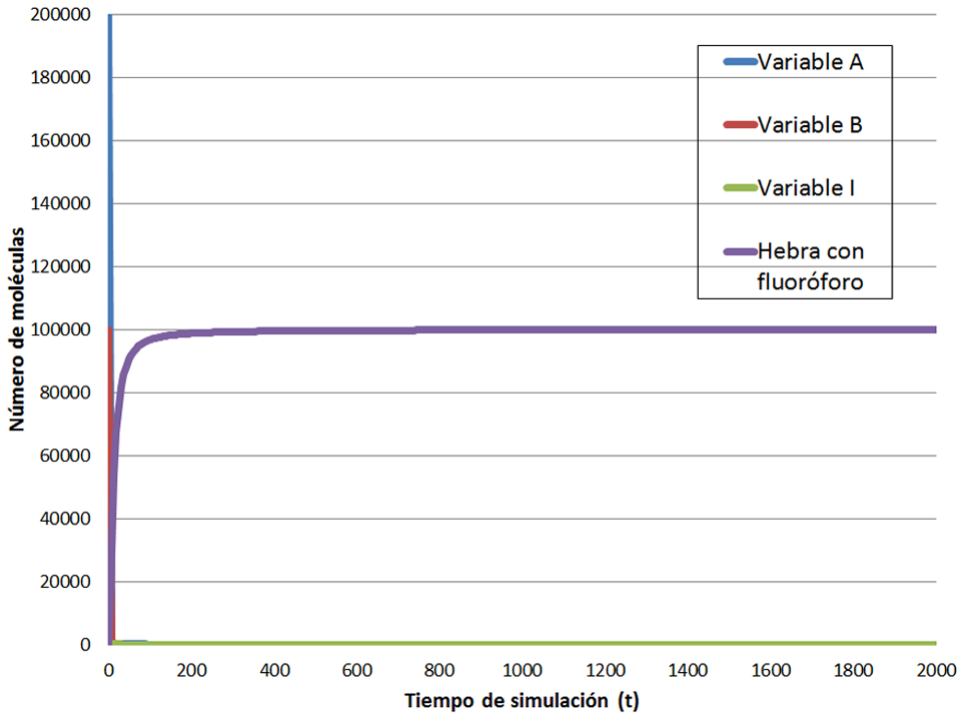


Figura 68: Simulación promedio para el caso 2 de la tabla de verdad del circuito para detección de cáncer de mama.

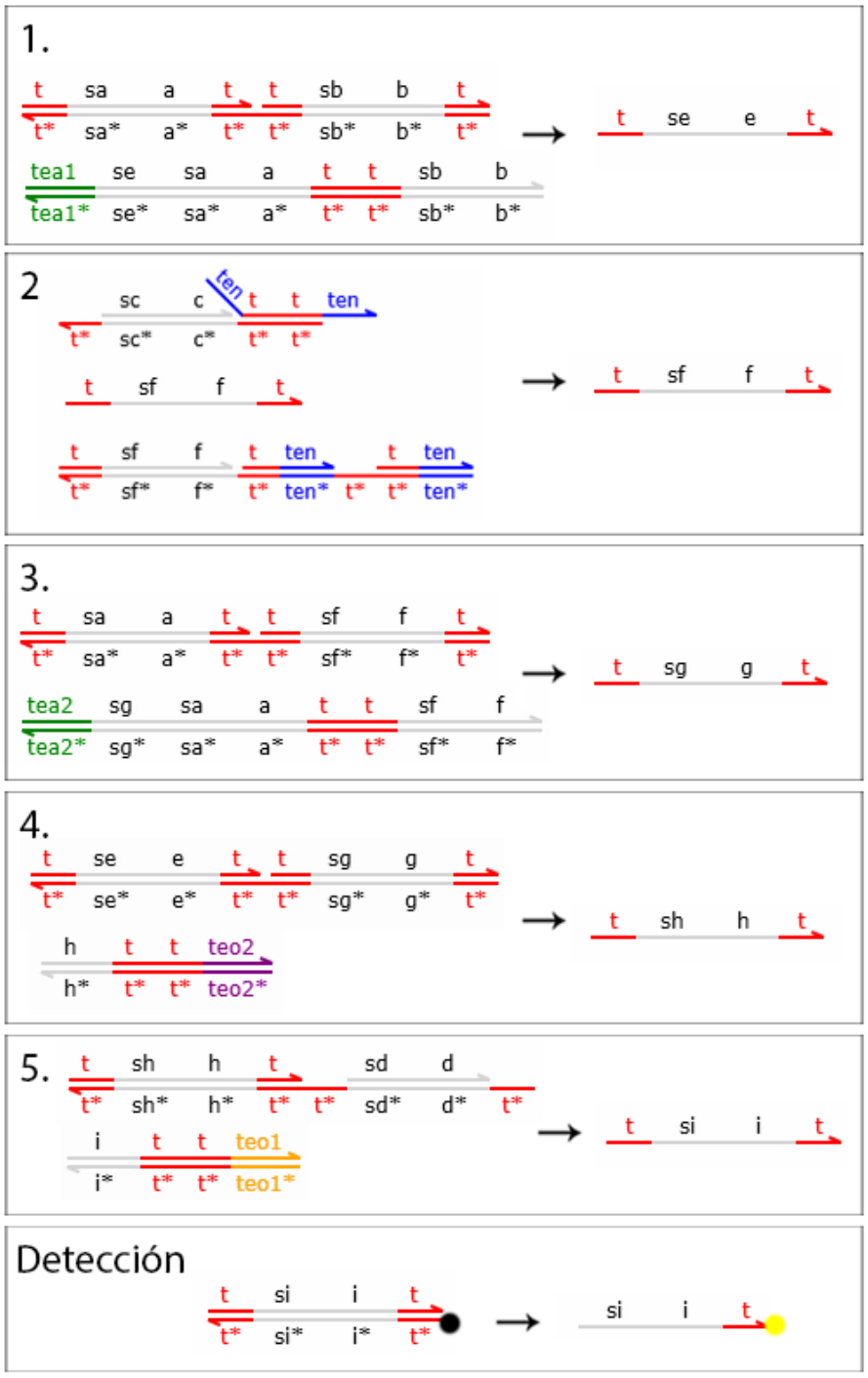
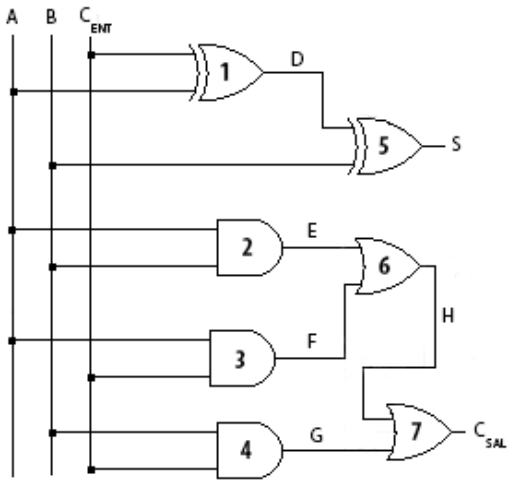
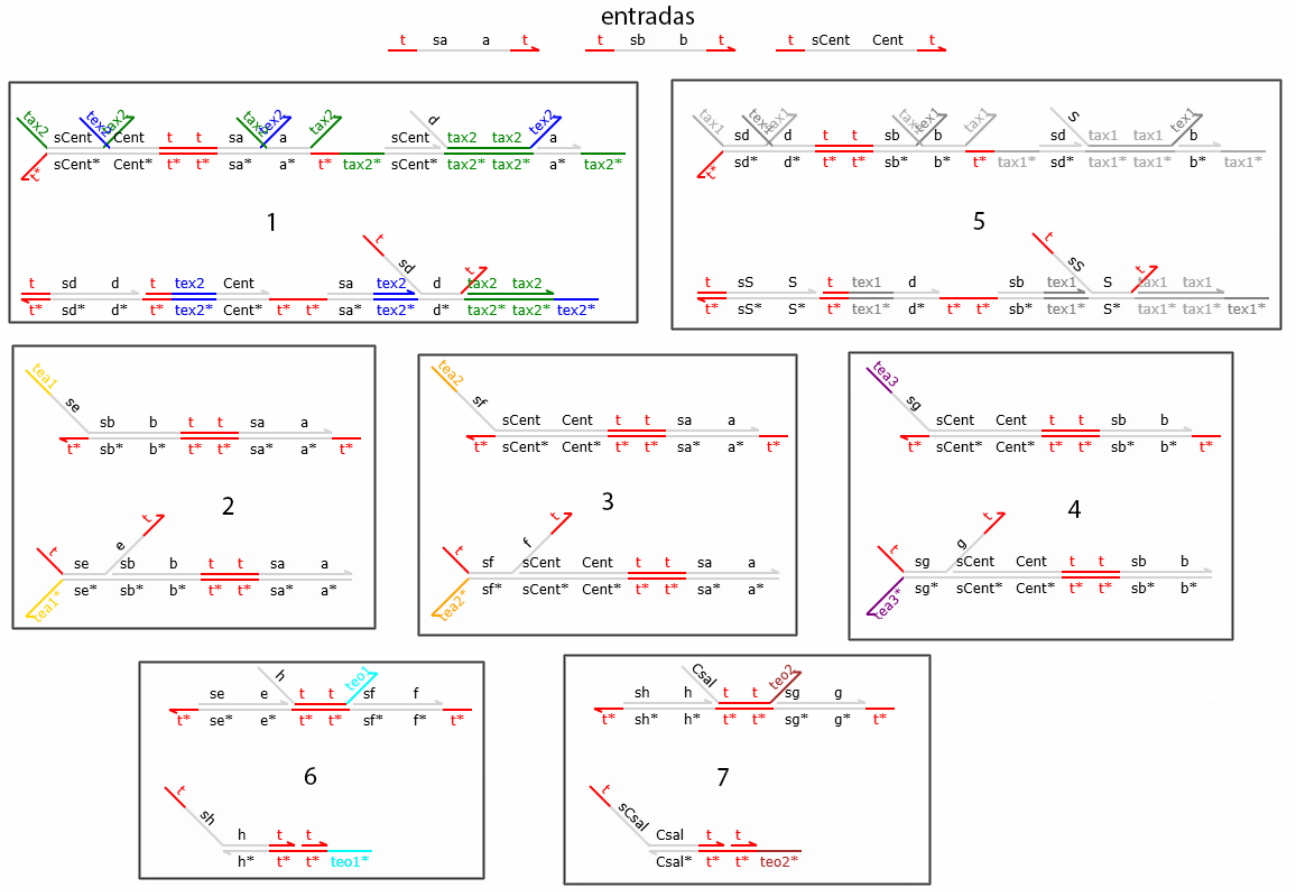


Figura 69: Resolución esquemática del caso 2 de la tabla de verdad (Tabla 9) con el modelo propuesto.



(a)



(b)

Figura 70: Circuito lógico para un sumador completo: (a) circuito lógico (Ecuación 23) y (b) circuito bioquímico equivalente.

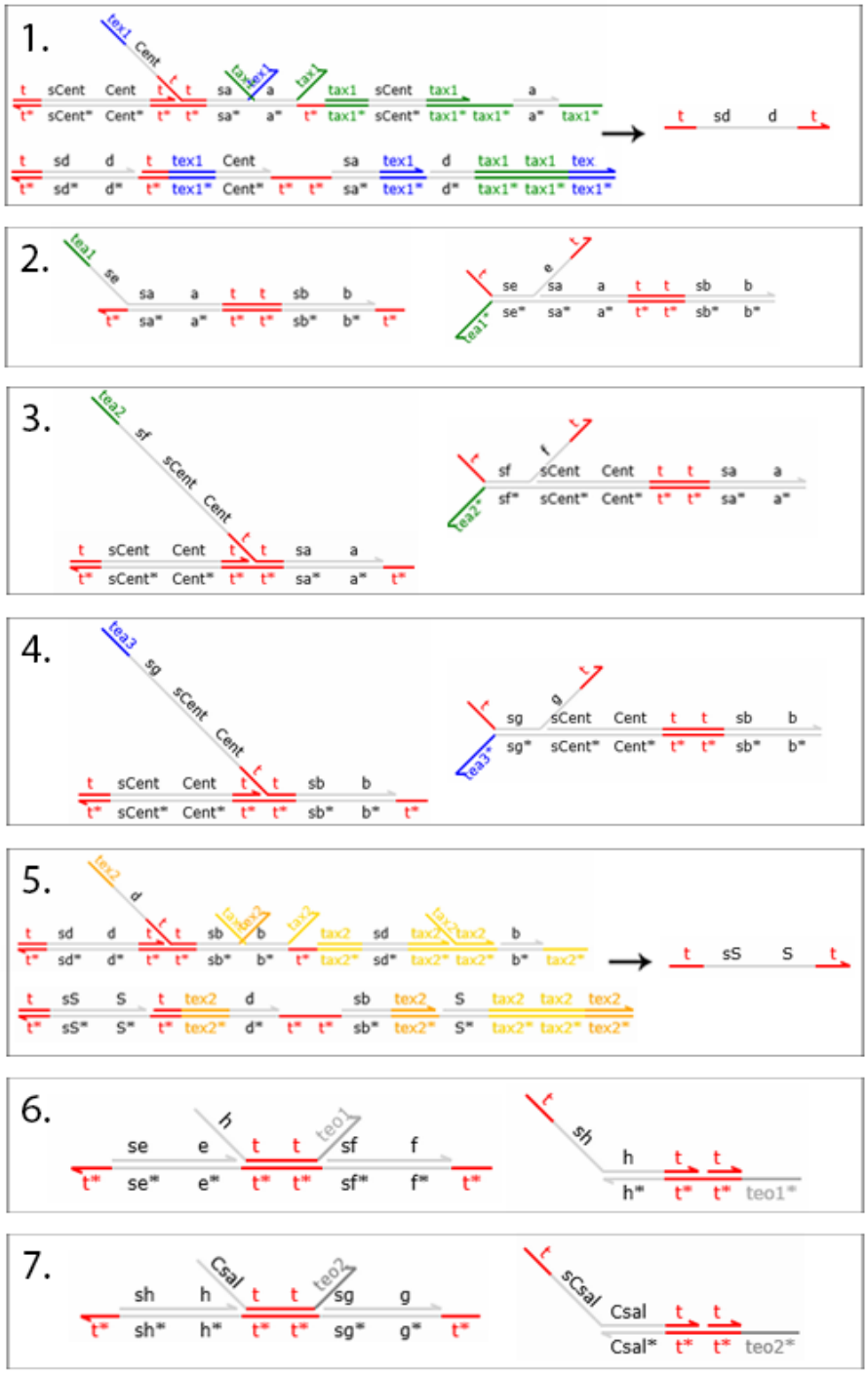


Figura 71: Resolución esquemática para el caso 2 de la tabla de verdad de un sumador completo con el modelo propuesto.

Tabla 13: Secuencias correspondientes a cada dominio para las simulaciones estocásticas del circuito sumador completo.

Dominio	Secuencia
t	$5'-$ TATTCC $-3'$
tax1	$5'-$ GCTA $-3'$
tax2	$5'-$ TACCAA $-3'$
tea1	$5'-$ ACTACAC $-3'$
tea2	$5'-$ CTCAG $-3'$
tea3	$5'-$ CTCAATC $-3'$
teo1	$5'-$ CCTACG $-3'$
teo2	$5'-$ TCTCCA $-3'$
tex1	$5'-$ GTCA $-3'$
tex2	$5'-$ CATCG $-3'$
Cent	$5'-$ CCCTTTTCTAAACTAAACAA $-3'$
Csal	$5'-$ CCATAACCTATACTTATCAA $-3'$
S	$5'-$ CCCTAAACTTATCTAAACAT $-3'$
a	$5'-$ CCCTTATCATATCAATACAA $-3'$
b	$5'-$ CCCTAATCTAATCATAACTA $-3'$
d	$5'-$ CCCTTAACTTAACAAATCTA $-3'$
e	$5'-$ CCCATATCTATACATTACAA $-3'$
f	$5'-$ CCCAATTCTTAACATATCAA $-3'$
g	$5'-$ CCCAAATCTTAACTATACTA $-3'$
h	$5'-$ CCTATACCTTAACTTAACAA $-3'$
sCent	$5'-$ CCCAAAACAAAACAAAACAA $-3'$
sCsal	$5'-$ CCATTTCCTTTTCTTAACTA $-3'$
sS	$5'-$ CCCATTTCAAATCAAAACTT $-3'$
sa	$5'-$ CCCTTTACATTACATAACAA $-3'$
sb	$5'-$ CCCTATACTATACAATACTA $-3'$
sd	$5'-$ CCCTATTCAATTCAAATCAA $-3'$
se	$5'-$ CCCATTACTAATCAATTCAA $-3'$
sf	$5'-$ CCCATAACTATTCTAAACTA $-3'$
sg	$5'-$ CCCAATACTATTCATAACAT $-3'$
sh	$5'-$ CCATATCCATAACTTTACAA $-3'$

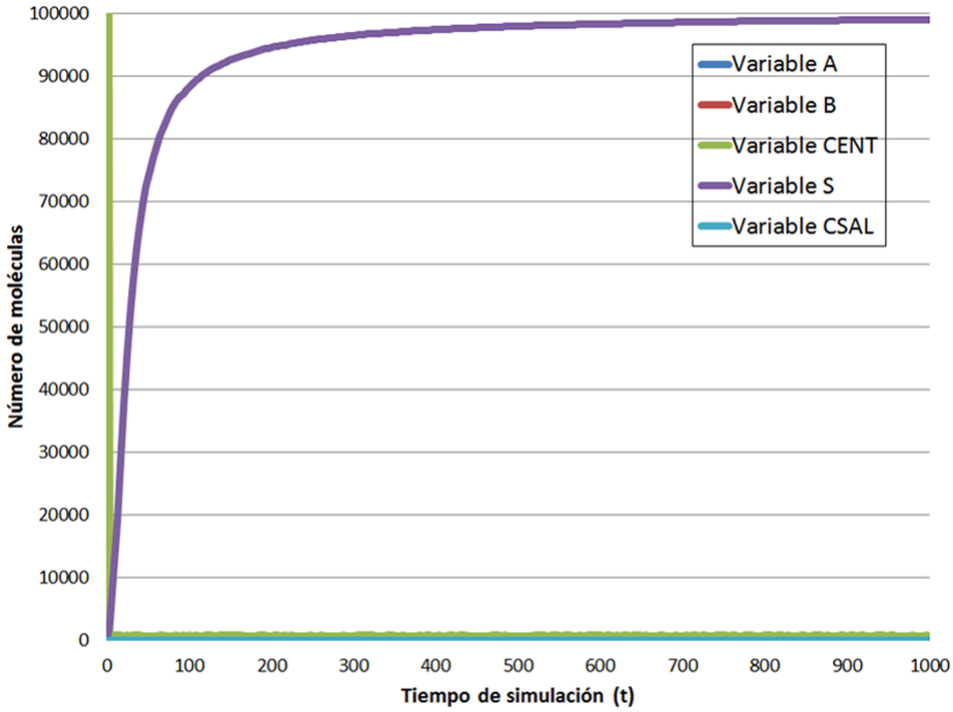


Figura 72: Simulación promedio para el caso 2 de la tabla de verdad del circuito para un sumador completo.

Tabla 14: Resultados de la simulación del caso 2 de la tabla de verdad del circuito para un sumador completo.

Molécula	Concentración final
Variable C_{ENT}	729
Variable C_{SAL}	0
Variable S	99,743
Variable D	1
XOR 1	0
AND 2	100,000
AND 3	33,530
AND 4	33,633
XOR 5	0
OR 6	100,000
OR 7	100,000

Capítulo 8. Conclusiones

8.1. Sumario

En este trabajo se presenta una biblioteca de compuertas lógicas (AND, OR, NOT, NAND, XOR, NOR, XNOR) basada en las propiedades termodinámicas del ADN mediante el enfoque de desplazamiento de cadenas. Para el diseño, simulación y verificación de los modelos propuestos se utilizaron las herramientas computacionales: Visual DSD, PRISM y NUPACK. Los experimentos *in silico* consistieron de simulaciones estocásticas con el algoritmo de Gillespie implementado en el software Visual DSD. Además, se analizaron las redes de reacciones y el espacio de estados con esta misma herramienta.

Las propiedades cuantitativas de los modelos fueron verificadas utilizando la herramienta PRISM, la cual junto con Visual DSD permitió corroborar que los modelos propuestos fueran correctos y presentaran el comportamiento esperado.

El análisis de las propiedades termodinámicas consistió básicamente en obtener la energía libre de cada uno de los complejos y se llevó a cabo por medio de la herramienta NUPACK. Esto se realizó principalmente con el fin de verificar la estabilidad termodinámica de las estructuras correspondientes a las compuertas.

Por último, se sugieren dos posibles aplicaciones de la biblioteca de compuertas lógicas para mostrar su potencial y comprobar su escalabilidad a circuitos más complejos que incluyan varias etapas de reacciones en cascada. Las aplicaciones presentadas consistieron en un circuito para detección de cáncer de mama en un modelo simplificado utilizando marcadores moleculares como entrada y un circuito sumador completo.

8.2. Conclusiones

- Los resultados obtenidos con las herramientas de simulación indican que en cuestiones termodinámicas todos los complejos son estables. Además, los resultados de las simulaciones estocásticas indican que todas las compuertas presentan una fiabilidad arriba del 98 %.

- El modelo propuesto permite implementar las siete compuertas lógicas básicas. Dado que cualquier expresión booleana se compone de combinaciones entre las operaciones básicas AND, OR y NOT es posible realizar cómputo universal. Adicionalmente, las compuertas NAND y NOR también se consideran compuertas universales ya que al combinarlas por sí solas realizan las tres operaciones booleanas básicas.

- Es importante mencionar que, a pesar de que algunas compuertas pueden formarse a partir de otras (por ejemplo NAND a partir de AND y NOT), al contar con los diseños específicos de éstas se requieren menos complejos y por consiguiente, al existir menos cadenas que reaccionan entre sí los resultados son menos propensos a errores principalmente causados por el ruido o interferencia.

- Otra de las ventajas que presenta contar con los diseños de las compuertas XOR, XNOR, NAND y NOR es la simplificación de circuitos, ya que es posible utilizarlas para implantar un circuito que sea equivalente al original pero que contenga menos compuertas y conexiones; por ejemplo, la propiedad de la compuerta XNOR de regresar una salida alta cuando ambas entradas tienen el mismo valor la hace ideal para la construcción de circuitos comparadores.

- Asimismo, con las aplicaciones conceptuales presentadas se comprueba que los diseños de las compuertas lógicas pueden utilizarse para implementar circuitos mucho más complejos. En los casos simulados, a pesar de existir varias etapas de reacciones en cascada, la atenuación de la señal de salida no fue significativa; sin embargo, es necesario continuar con la simulación de los casos restantes y con simulaciones de circuitos más complejos en el sentido del número de compuertas y reacciones en cascada para comprobar que el comportamiento de las compuertas siga siendo el mismo.

- El modelo presenta la ventaja de ser sencillo pero a la vez robusto ya que puede ser visto como bloques de construcción para circuitos más complejos tal como se comprobó en en el trabajo de Qian y Winfree (2011). Los resultados son fáciles de intepretar y, al no utilizar enzimas, su desempeño puede ser evaluado *in silico* con herramientas computacionales especializadas como Visual DSD, PRISM y

NUPACK, sin ser estrictamente necesaria su comprobación en laboratorio.

8.3. Trabajo futuro

A pesar del avance realizado, existe la posibilidad de mejorar este trabajo en cuestiones de optimización de los complejos de las compuertas. Como trabajo a futuro se desea considerar el tema de la reversibilidad de las compuertas, ya sea para implementar esta función directamente en los modelos propuestos o generar nuevas compuertas que la admitan.

Con respecto a la evaluación de las estructuras, sería conveniente realizar experimentos preliminares en laboratorio para corroborar la fiablilidad de las estructuras propuestas, además de analizar otras propiedades termodinámicas que se pueden obtener por medio de la herramienta computacional NUPACK, tales como el defecto de ensamble de las estructuras y la probabilidad de ensamble de los complejos, entre otras.

Desde el punto de vista de diseño, es conveniente realizar pruebas para determinar el grado de atenuación de las señales de salida de las compuertas, lo cual es particularmente importante para realizar experimentos en laboratorio. Para diferenciar claramente las señales de salida entre un ‘1’ y ‘0’ lógicos se pueden diseñar compuertas “umbral”dependiendo del grado de la atenuación (Seelig *et al.*, 2006).

Por último, también es deseable tratar de construir circuitos combinatorios más complejos como pueden ser multiplexores, decodificadores, flip-flops o algunos circuitos aritméticos como multiplicadores o divisores. Con relación a las dos aplicaciones conceptuales presentadas, es conveniente simular todos los casos de la tabla de verdad de cada uno de los circuitos para determinar si es correcto el comportamiento de las compuertas en cascada.

Lista de referencias

Adleman, L. (1994). Molecular computation of solutions to combinatorial problems. *Science*, **266**(5187): 1021–1024.

Adleman, L. M. (1998). Computing with DNA. *Scientific american*, **279**(8): 34–41.

Alur, R. y Henzinger, T. (1999). Reactive modules. *Formal Methods in System Design*, **15**(1): 7–48.

Barbuti, R., Maggiolo-Schettini, A., Milazzo, P., y Troina, A. (2005). An alternative to gillespie's algorithm for simulating chemical reactions. *Computational Methods in Systems Biology (CMSB'05), Edinburgh*.

Benenson, Y., Paz-Elizur, T., Adar, R., Keinan, E., Livneh, Z., y Shapiro, E. (2001). Programmable and autonomous computing machine made of biomolecules. *Nature*, **414**(6862): 430–434.

Berg, J., Stryer, L., Tymoczko, J., y Macarulla, J. (2008). *Bioquímica*. Reverté. p. 1152.

Chandran, H., Gopalkrishnan, N., Phillips, A., y Reif, J. (2011). Localized hybridization circuits. En: L. Cardelli y W. Shih (eds.), *DNA Computing and Molecular Programming*, Vol. 6937 de *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, pp. 64–83.

Chiniforooshan, E., Doty, D., Kari, L., y Seki, S. (2011). Scalable, time-responsive, digital, energy-efficient molecular circuits using DNA strand displacement. En: *DNA computing and molecular programming*. Springer, pp. 25–36.

Crick, F. H. C. (1958). On protein synthesis. *The Symposia of the Society for Experimental Biology*, **12**: 138–163.

de Silva, A. P. y Uchiyama, S. (2007). Molecular logic and computing. *Nature Nanotechnology*, **2**(7): 399–410.

Fehlberg, E. (1969). Low-order classical runge-kutta formulas with stepsize control and their application to some heat transfer problems. *NASA, USA: Technical Report NASA-TR-R-315*, p. 43.

Gamerman, D. y Lopes, H. F. (2006). *Markov Chain Monte Carlo: Stochastic Simulation for Bayesian Inference*. Chapman and Hall/CRC. p. 344.

Genot, A. J., Zhang, D. Y., Bath, J., y Turberfield, A. J. (2011). Remote toehold: a mechanism for flexible control of dna hybridization kinetics. *Journal of the American Chemical Society*, **133**(7): 2177–2182.

Gibbons, A., Amos, M., y Hodgson, D. (1996). Models of DNA computation. En: W. Penczek y A. Szałas (eds.), *Mathematical Foundations of Computer Science 1996*, Vol. 1113 de *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, pp. 18–36.

Gillespie, D. T. (1977). Exact stochastic simulation of coupled chemical reactions. *The Journal of Physical Chemistry*, **81**(25): 2340–2361.

Gonzalez, T., Diaz-Herrera, J., y Tucker, A. (2014). *Computing Handbook: Computer Science and Software Engineering*, Vol. 1. CRC Press. p. 3816.

Hagiya, M., Arita, M., Kiga, D., Sakamoto, K., y Yokoyama, S. (1997). Towards parallel evaluation and learning of boolean
mu-Formulas with molecules. En: *Proceedings of Third Annual Meeting on DNA Based Computers*. Vol. 48, pp. 105–114.

Hamano, R., Ishii, H., Miyata, H., Doki, Y., y Mori, M. (2011). Role of microRNAs in solid tumors. *Journal of Nucleic Acids Investigation*, **2**(1): 5–15.

Harju, T. y Margenstern, M. (2005). Splicing systems for universal turing machines. En: C. Ferretti, G. Mauri, y C. Zandron (eds.), *DNA Computing*, Vol. 3384 de *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, pp. 149–158.

Head, T. (1987). Formal language theory and DNA: An analysis of the generative capacity of specific recombinant behaviors. *Bulletin of Mathematical Biology*, **49**(6): 737–759.

Ignatova, Z., Martínez-Pérez, I., y Zimmermann, K. (2008). *DNA Computing Models*. Advances in information security. Springer. p. 288.

Jain, K. K. (2010). *The Handbook of Biomarkers*. Humana Press, primera edición. p. 462.

Kahan-Hanum, M., Douek, Y., Adar, R., y Shapiro, E. (2013). A library of programmable DNAzymes that operate in a cellular environment. *Scientific reports*, **3**: 1535.

Koehler, R. T. y Peyret, N. (2005). Thermodynamic properties of DNA sequences: characteristic values for the human genome. *Bioinformatics*, **21**(16): 3333–3339.

Kuramochi, J. y Sakakibara, Y. (2006). Intensive *in vitro* experiments of implementing and executing finite automata in test tube. En: A. Carbone y N. Pierce (eds.), *DNA Computing*, Vol. 3892 de *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, pp. 193–202.

Kwiatkowska, M., Norman, G., y Parker, D. (2011). PRISM 4.0: Verification of Probabilistic Real-time Systems. En: *Proceedings of the 23rd International Conference on Computer Aided Verification*, Berlin, Heidelberg. Springer-Verlag, CAV'11, pp. 585–591.

Lakin, M. R. y Phillips, A. (2011). Modelling, simulating and verifying turing-powerful strand displacement systems. En: *Proceedings of the 17th International Conference on DNA Computing and Molecular Programming*, Berlin, Heidelberg. Springer-Verlag, DNA'11, pp. 130–144.

Lakin, M. R., Parker, D., Cardelli, L., Kwiatkowska, M., y Phillips, A. (2012a). Design and analysis of DNA strand displacement devices using probabilistic model checking. *Journal of the Royal Society Interface*, **9**(72): 1470–1485.

Lakin, M. R., Paulevé, L., y Phillips, A. (2012b). Stochastic simulation of multiple process calculi for biology. *Theor. Comput. Sci.*, **431**: 181–206.

Lakin, M. R., Youssef, S., Cardelli, L., y Phillips, A. (2012c). Abstractions for DNA circuit design. *Journal of The Royal Society Interface*, **9**(68): 470–486.

Lee, C. S., Davis, R. W., y Davidson, N. (1970). A physical study by electron microscopy of the terminally repetitious, circularly permuted DNA from the coliphage particles of escherichia coli 15. *Journal of Molecular Biology*, **48**(1): 1 – 22.

Li, W., Yang, Y., Yan, H., y Liu, Y. (2013). Three-input majority logic gate and multiple input logic circuit based on DNA strand displacement. *Nano Letters*, **13**(6): 2980–2988.

Lipton, R. J. (1995). DNA solution of hard computational problems. *Science*, **268**(5210): 542–545.

Macdonald, J., Li, Y., Sutovic, M., Lederman, H., Pendri, K., Lu, W., Andrews, B. L., Stefanovic, D., y Stojanovic, M. N. (2006). Medium scale integration of molecular logic gates in an automaton. *Nano letters*, **6**(11): 2598–2603.

Martínez-Pérez, I. M., Zimmermann, K., y Ignatova, Z. (2009). An autonomous DNA model for finite state automata. *Int. J. Bioinformatics Res. Appl.*, **5**(1): 81–96.

Modi, S., Swetha, Goswami, D., Gupta, G. D., Mayor, S., y Krishnan, Y. (2009). A DNA nanomachine that maps spatial and temporal pH changes inside living cells. *Nature Nanotechnology*, **4**(5): 325–330.

Nelson, B. L. (2002). *Stochastic modeling - analysis and simulation*. Dover Publications. pp. I–XIV, 1–321.

Ouyang, Q., Kaplan, P. D., Liu, S., y Libchaber, A. (1997). DNA solution of the maximal clique problem. *Science*, **278**(5337): 446–449.

Parzen, E. (1999). *Stochastic Processes*. Classics in Applied Mathematics. Society for Industrial and Applied Mathematics. p. 338.

Phillips, A. y Cardelli, L. (2009). A programming language for composable DNA circuits. *Journal of the Royal Society Interface*, **6**(S4): 419–436.

Polanski, A. y Kimmel, M. (2007). *Bioinformatics*. Springer. p. 396.

Prokup, A., Hemphill, J., y Deiters, A. (2012). DNA computation: a photochemically controlled and gate. *Journal of the American Chemical Society*, **134**(8): 3810–3815.

Qian, L. y Winfree, E. (2009). A simple DNA gate motif for synthesizing large-scale circuits. En: *DNA computing*. Springer, pp. 70–89.

Qian, L. y Winfree, E. (2011). Scaling up digital circuit computation with DNA strand displacement cascades. *Science (New York, N.Y.)*, **332**(6034): 1196–1201.

Reynaldo, L. P., Vologodskii, A. V., Neri, B. P., y Lyamichev, V. I. (2000). The kinetics of oligonucleotide replacements. *Journal of Molecular Biology*, **297**(2): 511 – 520.

Rothemund, P. W. (1996). A DNA and restriction enzyme implementation of turing machines. *DNA based computers*, **6**: 75–120.

Rothemund, P. W., Papadakis, N., y Winfree, E. (2004). Algorithmic self-assembly of DNA Sierpinski triangles. *PLoS biology*, **2**(12): e424.

Roweis, S., Winfree, E., Burgoyne, R., Chelyapov, N. V., Goodman, M. F., Rothemund, P. W., y Adleman, L. M. (1998). A sticker-based model for DNA computation. *Journal of Computational Biology*, **5**(4): 615–629.

SantaLucia, J. y Hicks, D. (2004). The thermodynamics of DNA structural motifs. *Annual review of biophysics and biomolecular structure*, **33**: 415–440.

Schmitz, U., Wolkenhauer, O., y Vera, J. (2013). *MicroRNA Cancer Regulation: Advanced Concepts, Bioinformatics and Systems Biology Tools*. Advances in Experimental Medicine and Biology. Springer.

Seelig, G., Soloveichik, D., Zhang, D. Y., y Winfree, E. (2006). Enzyme-free nucleic acid logic circuits. *Science*, **314**(5805): 1585–1588.

Soloveichik, D., Seelig, G., y Winfree, E. (2010). Dna as a universal substrate for chemical kinetics. *Proceedings of the National Academy of Sciences*, **107**(12): 5393–5398.

Stojanovic, M. N. y Stefanovic, D. (2003). A deoxyribozyme-based molecular automaton. *Nature biotechnology*, **21**(9): 1069–74.

Sugimoto, N., Nakano, S.-i., Katoh, M., Matsumura, A., Nakamuta, H., Ohmichi, T., Yoneyama, M., y Sasaki, M. (1995). Thermodynamic parameters to predict stability of RNA/DNA hybrid duplexes. *Biochemistry*, **34**(35): 11211–11216.

Tocci, R. J. y Widmer, N. S. (2003). *Sistemas digitales: principios y aplicaciones*. Pearson Educación. p. 881.

Waterman, M. S. (1995). *Introduction to Computational Biology: Maps, Sequences and Genomes*. Chapman & Hall. London.

Watkins, N. E., Kennelly, W. J., Tsay, M. J., Tuin, A., Swenson, L., Lee, H.-R., Morosyuk, S., Hicks, D. A., y SantaLucia, J. (2011). Thermodynamic contributions of single internal rA·dA, rC·dC, rG·dG and rU·dT mismatches in RNA/DNA duplexes. *Nucleic Acids Research*, **39**(5): 1894–1902.

Win, M. N. y Smolke, C. D. (2008). Higher-order cellular information processing with synthetic RNA devices. *Science (New York, N.Y.)*, **322**(5900): 456–60.

Winfree, E. (1998). Whiplash PCR for O(1) computing. En: *University of Pennsylvania*. pp. 175–188.

Winfree, E., Yang, X., y Seeman, N. C. (1996). Universal computation via self-assembly of DNA: Some theory and experiments. En: *DNA Based Computers II, volume 44 of DIMACS*. American Mathematical Society, pp. 191–213.

Xia, T., SantaLucia, J., Burkard, M. E., Kierzek, R., Schroeder, S. J., Jiao, X., Cox, C., y Turner, D. H. (1998). Thermodynamic parameters for an expanded nearest-neighbor model for formation of RNA duplexes with Watson−Crick base pairs. *Biochemistry*, **37**(42): 14719–14735.

Yurke, B., Turberfield, A. J., Mills, A. P., Simmel, F. C., y Neumann, J. L. (2000). A DNA-fuelled molecular machine made of DNA. *Nature*, **406**(6796): 605–608.

Zadeh, J. N., Steenberg, C., Bois, J. S., Wolfe, B. R., Pierce, M. B., Khan, A. R., Dirks, R. M., y Pierce, N. A. (2011). Nupack: Analysis and design of nucleic acid systems. *Journal of Computational Chemistry*, **32**(1): 170–173.

Zhang, C., Yang, J., y Xu, J. (2010). Circular DNA logic gates with strand displacement. *Langmuir*, **26**(3): 1416–1419. PMID: 19957974.

Zhang, D. Y. y Seelig, G. (2011). Dynamic DNA nanotechnology using strand-displacement reactions. *Nature chemistry*, **3**(2): 103–113.

Zhang, D. Y., Turberfield, A. J., Yurke, B., y Winfree, E. (2007). Engineering entropy-driven reactions and networks catalyzed by DNA. *Science*, **318**(5853): 1121–1125.

Apéndice A. Información adicional de simulaciones de las compuertas

La herramienta Visual DSD también permite calcular el grafo de todos los posibles estados alcanzables del sistema a partir de las moléculas iniciales. Cada estado alcanzable se representa por un nodo del grafo y cada una de las posibles reacciones como una arista. El estado inicial se indica remarcando el borde con color negro y el estado final (estado a partir del cual ya no se pueden generar más reacciones) con color rojo. También se indican estadísticas como la probabilidad de ocurrencia de cada reacción, número total de estados alcanzables, de estados finales y de transiciones.

A.1. AND

La compuerta AND tiene cuatro estados iniciales (Figura A.1(a)–(d)), los cuales corresponden a las cuatro posibles combinaciones de entradas de la tabla de verdad y dos estados finales (Figura A.2): el primero (Figura A.2(a)) es donde los complejos principal y auxiliar no reaccionan debido a que las entradas no se encontraban presentes (por lo que también es un estado inicial) y el segundo (Figura A.2(b)) se alcanza cuando ambas variables de entrada se encontraban presentes y reaccionaron con el complejo principal generando las reacciones necesarias para liberar la variable de salida a partir del complejo auxiliar.

A.2. OR

La compuerta OR tiene cuatro estados iniciales (Figura 3(a)–(d)), los cuales corresponden a las cuatro posibles combinaciones de entradas de la tabla de verdad y cuatro estados finales (Figura A.4). Cada uno de los estados finales se alcanza mediante su correspondiente estado inicial: el estado inicial donde no están presentes ninguna de las variables (Figura 3(a)) es también un estado final (Figura 4(a)) ya que al no existir entradas no es posible que ocurran reacciones entre los complejos; el estado final 2 (Figura 4(b)) se alcanza a partir del estado inicial donde sólo está presente la variable B (Figura 3(b)); el estado final 3 (Figura 4(c)) se alcanza a partir de estado inicial donde sólo está presente la variable A (Figura 3(c)). Por último, el estado final 4 (Figura 4(d)) se alcanza a

partir del estado inicial donde se encuentran presentes las dos variables (Figura 3(d)). En los estados finales (Figura 4(b)–(d)) se libera la variable de salida, ya que para que la compuerta OR regrese un ‘1’ lógico se requiere que al menos una entrada esté presente.

A.3. NOT

La compuerta NOT tiene dos estados iniciales (Figura A.5) que corresponden a las dos posibles combinaciones de entradas de la tabla de verdad y dos estados finales (Figura A.6). El primer estado inicial (Figura 5(a)) es cuando no hay variable de entrada y éste también corresponde a un estado final (Figura 6(a)) ya que no ocurren reacciones entre los complejos. El segundo estado inicial (Figura 5(b)) es cuando ambas variables de entrada se encuentran presentes y a partir de éste se alcanza el estado final 2 (Figura 6(b)), donde la variable de entrada reacciona con el complejo principal generando las reacciones necesarias para atrapar la variable de salida mediante complejo auxiliar, dando como resultado un ‘0’ lógico.

A.4. NAND

La compuerta NAND tiene cuatro estados iniciales (Figura 7(a)–(d)), los cuales corresponden a las cuatro posibles combinaciones de entradas de la tabla de verdad y cuatro estados finales (Figura A.8). Cada uno de los estados finales se alcanza mediante su correspondiente estado inicial. El estado inicial donde no están presentes ninguna de las variables (Figura 7(a)) es también un estado final (Figura 8(a)), ya que al no existir entradas no es posible que ocurran reacciones entre los complejos; el estado final 2 (Figura 8(b)) se alcanza a partir del estado inicial donde sólo está presente la variable B (Figura 7(b)); el estado final 3 (Figura 8(c)) se alcanza a partir de estado inicial donde sólo está presente la variable A (Figura 7(c)). El estado final 4 (Figura 8(d)) se alcanza a partir del estado inicial donde se encuentran presentes las dos variables (Figura 7(d)). En los estados finales 1-3 (Figura 8(a)–(c)), se libera o se encuentra presente la variable de salida, ya que el único caso en que la compuerta NAND regresa un ‘0’ lógico corresponde al estado final 4 (Figura 8(d)).

A.5. NOR

La compuerta NOR tiene cuatro estados iniciales (Figura 9(a)–(d)), los cuales corresponden a las cuatro posibles combinaciones de entradas de la tabla de verdad y cuatro estados finales (Figura A.10). Cada uno de los estados finales se alcanza mediante su correspondiente estado inicial. El estado inicial donde no están presentes ninguna de las variables (Figura 9(a)) es también un estado final (Figura 10(a)), ya que al no existir entradas no es posible que ocurran reacciones entre los complejos y es el único caso donde la variable de salida está presente indicando un '1' lógico. El estado final 2 (Figura 10(b)) se alcanza a partir del estado inicial donde sólo está presente la variable B (Figura 9(b)); el estado final 3 (Figura 10(c)) se alcanza a partir del estado inicial donde sólo está presente la variable A (Figura 9(c)); el estado final 4 (Figura 10(d)) se alcanza a partir de estado inicial donde se encuentran presentes las dos variables (Figura 9(d)). En los estados finales 2-4 (Figura 10(b)–(d),) la variable de salida no se encuentra presente para representar un '0' lógico.

A.6. XOR

La compuerta XOR tiene cuatro estados iniciales (Figura 11(a)–(d)), los cuales corresponden a las cuatro posibles combinaciones de entradas de la tabla de verdad y dos estados finales (Figura A.12). El primero (Figura 12(a)) es donde los complejos principal y auxiliar no reaccionan debido a que las entradas no se encontraban presentes y es también un estado inicial (Figura 11(a)); el segundo estado final (Figura 12(b)) se alcanza mediante el estado inicial (Figura 11(b)) donde ambas variables de entrada se encontraban presentes y reaccionaron con el complejo principal generando las reacciones necesarias para atrapar la variable de salida a partir del complejo auxiliar, puesto que la salida para este caso debe ser un '0' lógico. La compuerta XOR regresa un '1' lógico cuando solamente alguna de las dos variables de entrada está presente; sin embargo, estos dos casos no corresponden a estados finales en el sistema y esto se debe a que los complejos de la compuerta pueden seguir generando reacciones a causa de que existen hebras de protección para monitorear la presencia de ambas variables con el fin de identificar cuando se debe atrapar la variable de salida y obtener así el funcionamiento propio de la operación XOR.

A.7. XNOR

La compuerta XNOR tiene cuatro estados iniciales (Figura 13(a)–(d)), los cuales corresponden a las cuatro posibles combinaciones de entradas de la tabla de verdad y dos estados finales (Figura A.14). El estado inicial (Figura 13(a)) donde no están presentes ninguna de las entradas corresponde también a un estado final (Figura 14(a)), ya que no ocurren reacciones entre los complejos de la compuerta. Para los estados iniciales (Figura 13(b)) y (Figura 13(c)) que corresponden a los casos donde sólo está presente una de las dos variables de entrada, no existen estados finales ya que después de hibridarse con sus partes complementarias y desatar las reacciones para que la variable de salida se hibride en el complejo auxiliar (para regresar un '0' lógico) aún existen hebras que pueden desencadenar reacciones entre los complejos; esto se debe a que se desea monitorear cuando están presentes ambas variables (caso correspondiente al estado inicial de la Figura 13(d)) para liberar la variable de salida y de esta manera regresar un '1' lógico (lo que corresponde al estado final 2 en la Figura 14(b)).

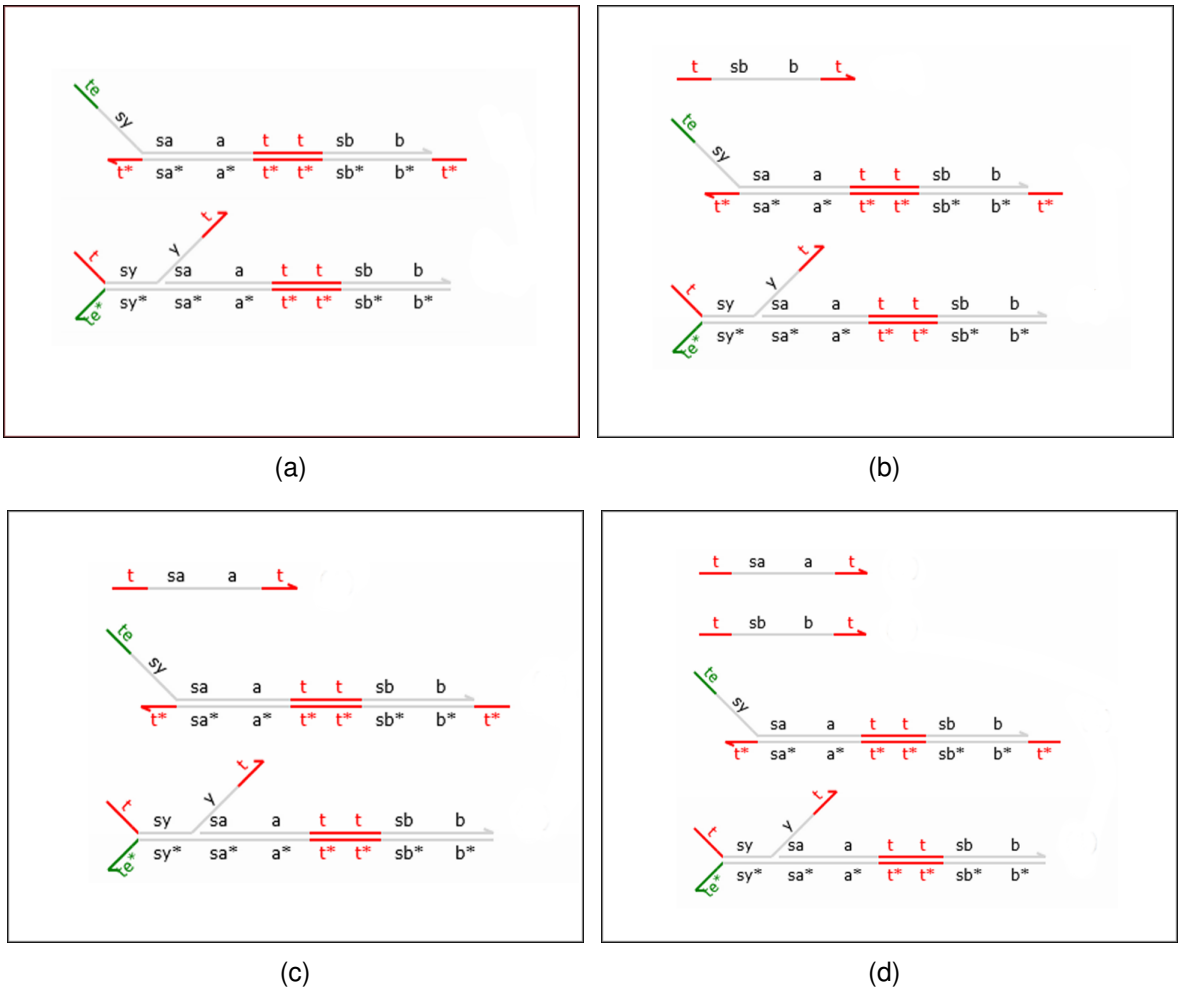


Figura A.1: Estados iniciales de la compuerta AND: (a) sin entradas; (b) variable B presente; (c) variable A presente; (d) variables A y B presentes.

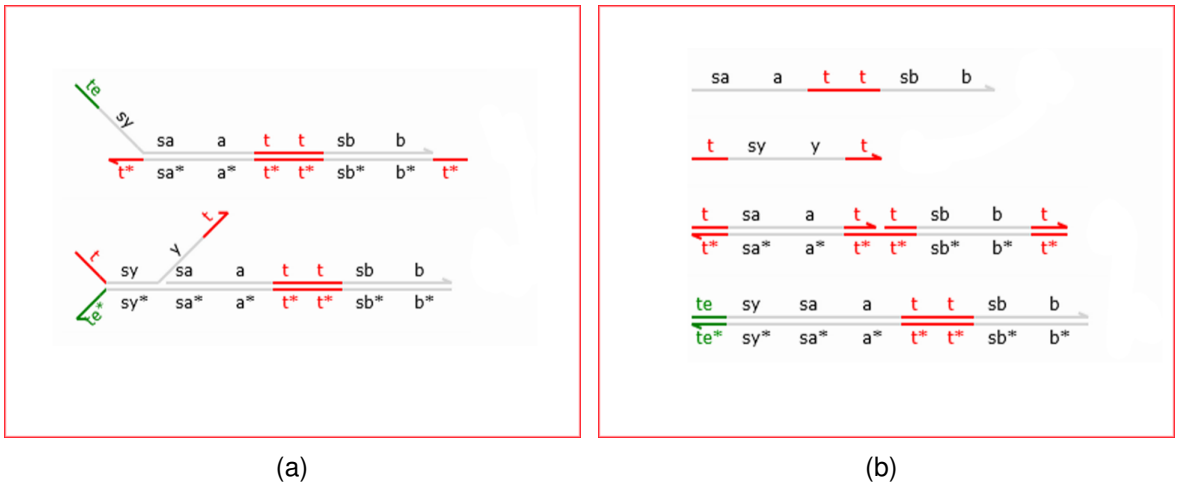


Figura A.2: Estados finales de la compuerta AND: (a) estado final 1 y (b) estado final 2.

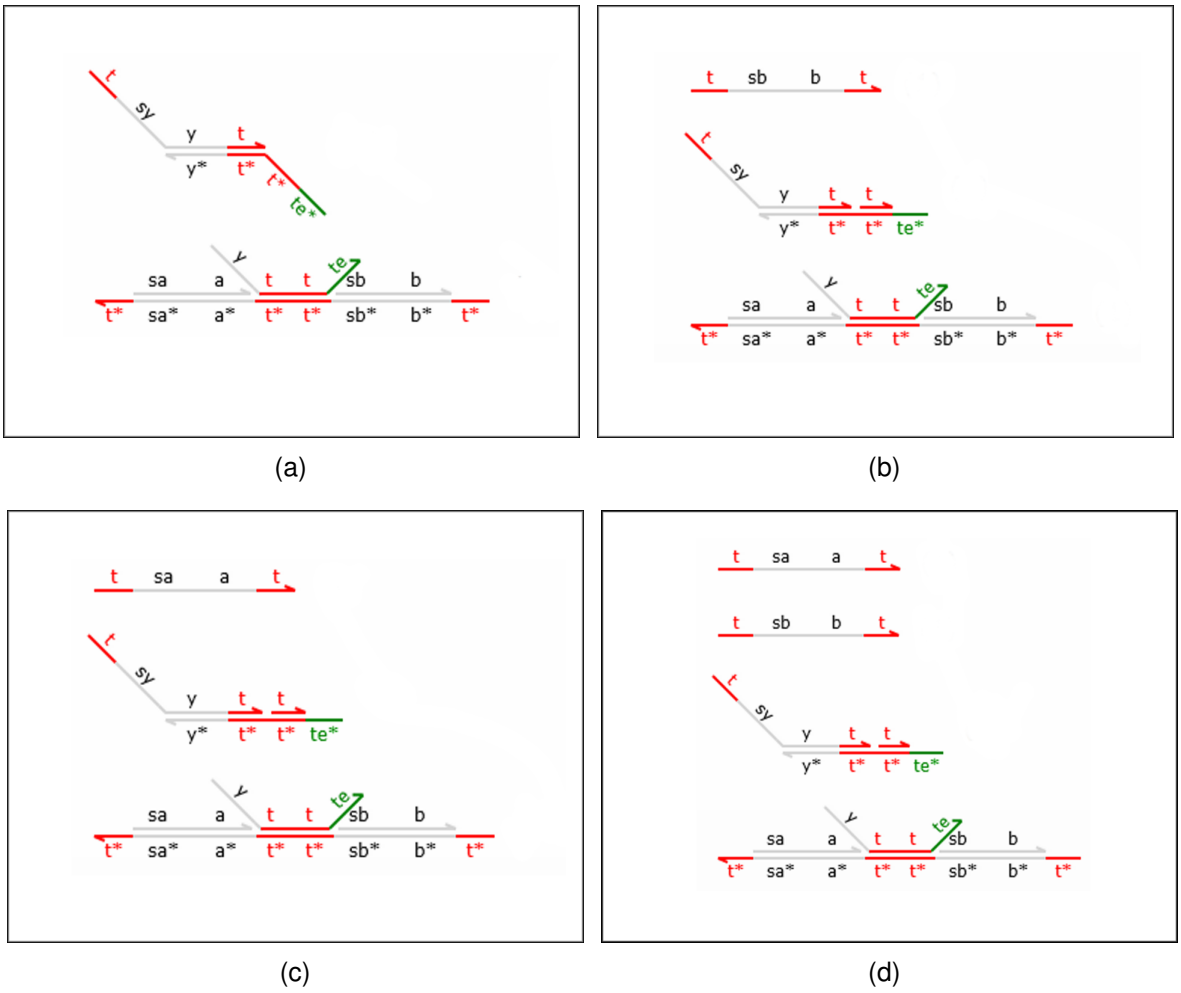


Figura A.3: Estados iniciales de la compuerta OR: (a) sin entradas; (b) variable B presente; (c) variable A presente; (d) variables A y B presentes.

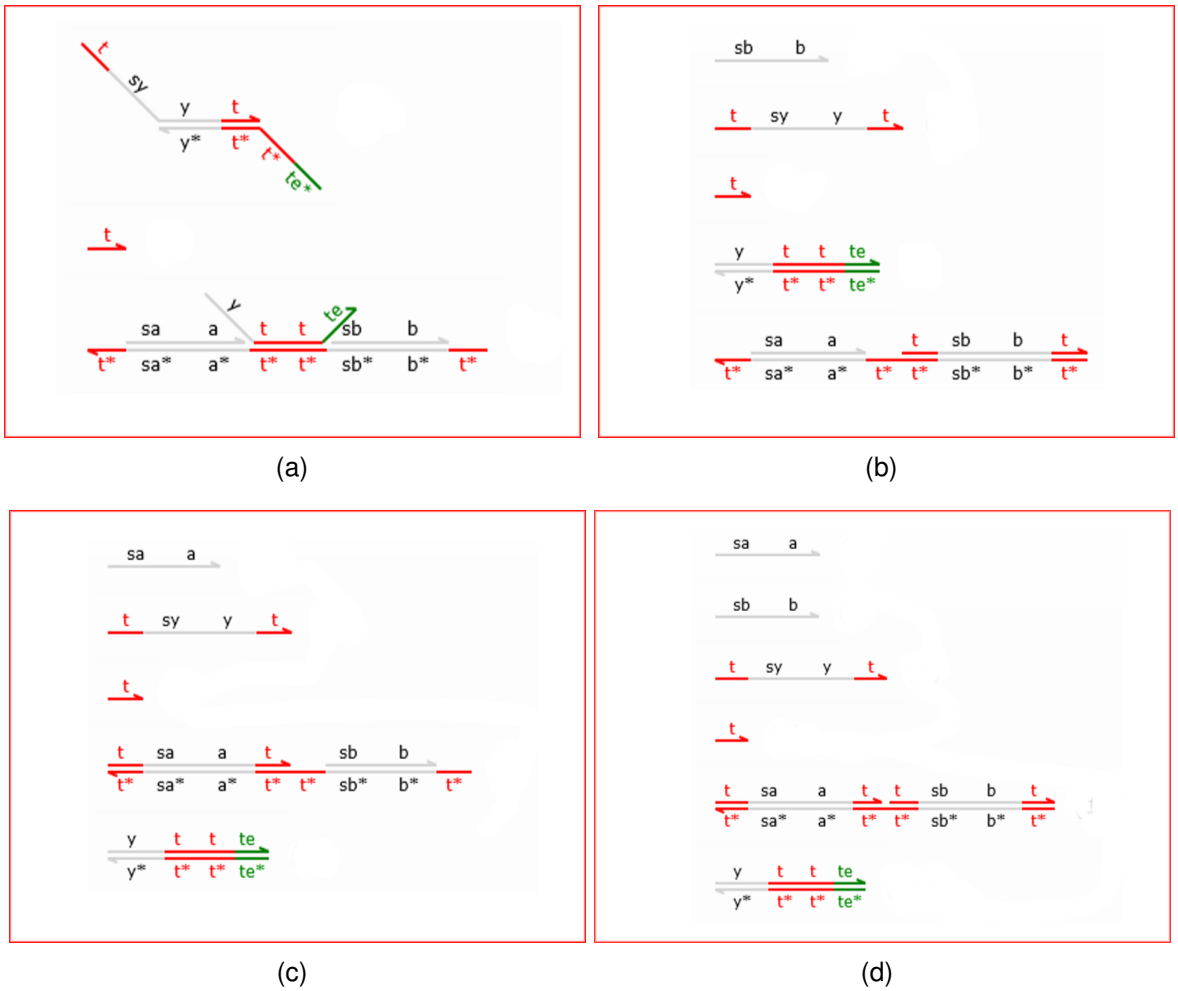


Figura A.4: Estados finales de la compuerta OR: (a) estado final 1; (b) estado final 2; (c) estado final 3 y (d) estado final 4.

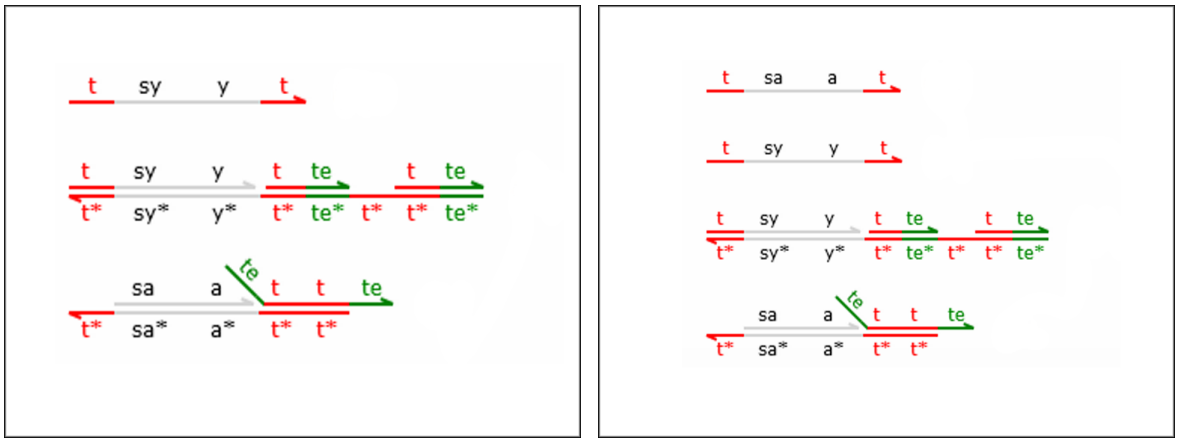
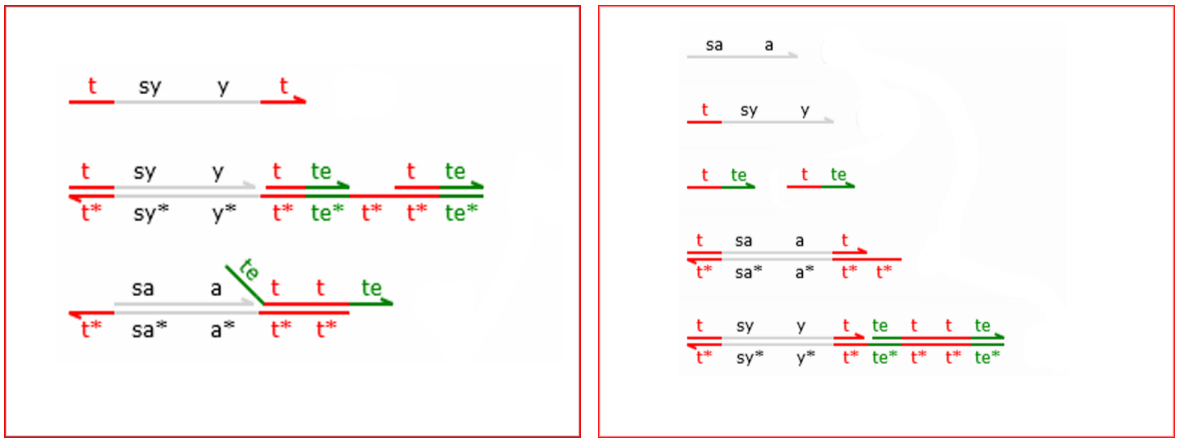
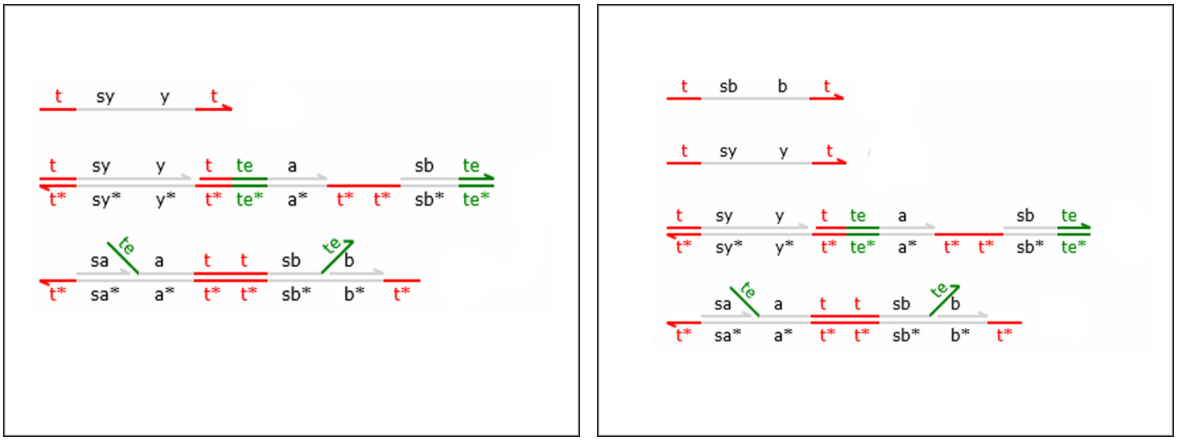


Figura A.5: Estados iniciales de la compuerta NOT: (a) sin entrada; (b) variable A presente.

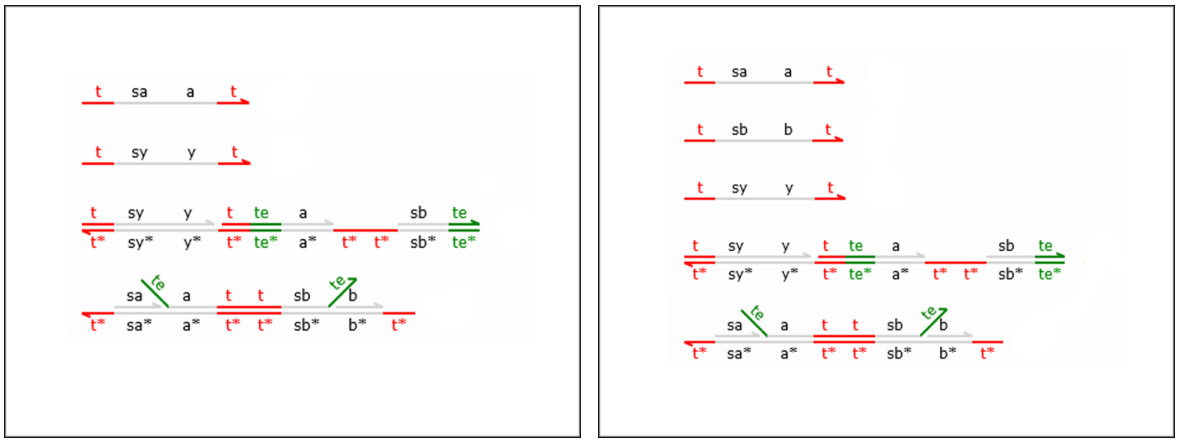


(a) (b)

Figura A.6: Estados finales de la compuerta NOT: (a) estado final 1 y (b) estado final 2.



(a) (b)



(c) (d)

Figura A.7: Estados iniciales de la compuerta NAND: (a) sin entradas; (b) variable B presente; (c) variable A presente; (d) variables A y B presentes.

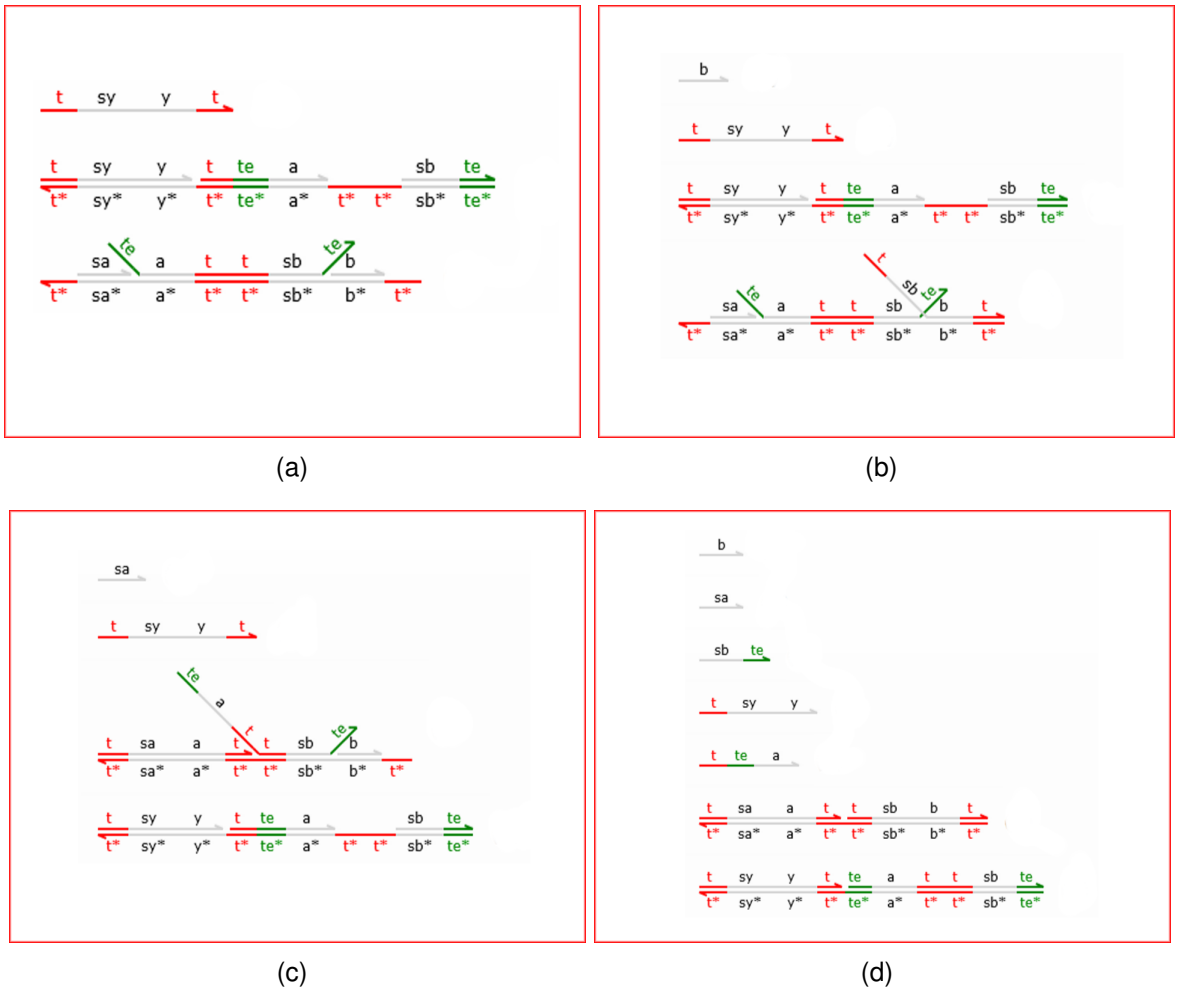


Figura A.8: Estados finales de la compuerta NAND: (a) estado final 1 ; (b) estado final 2 ; (c) estado final 3 y (d) estado final 4.

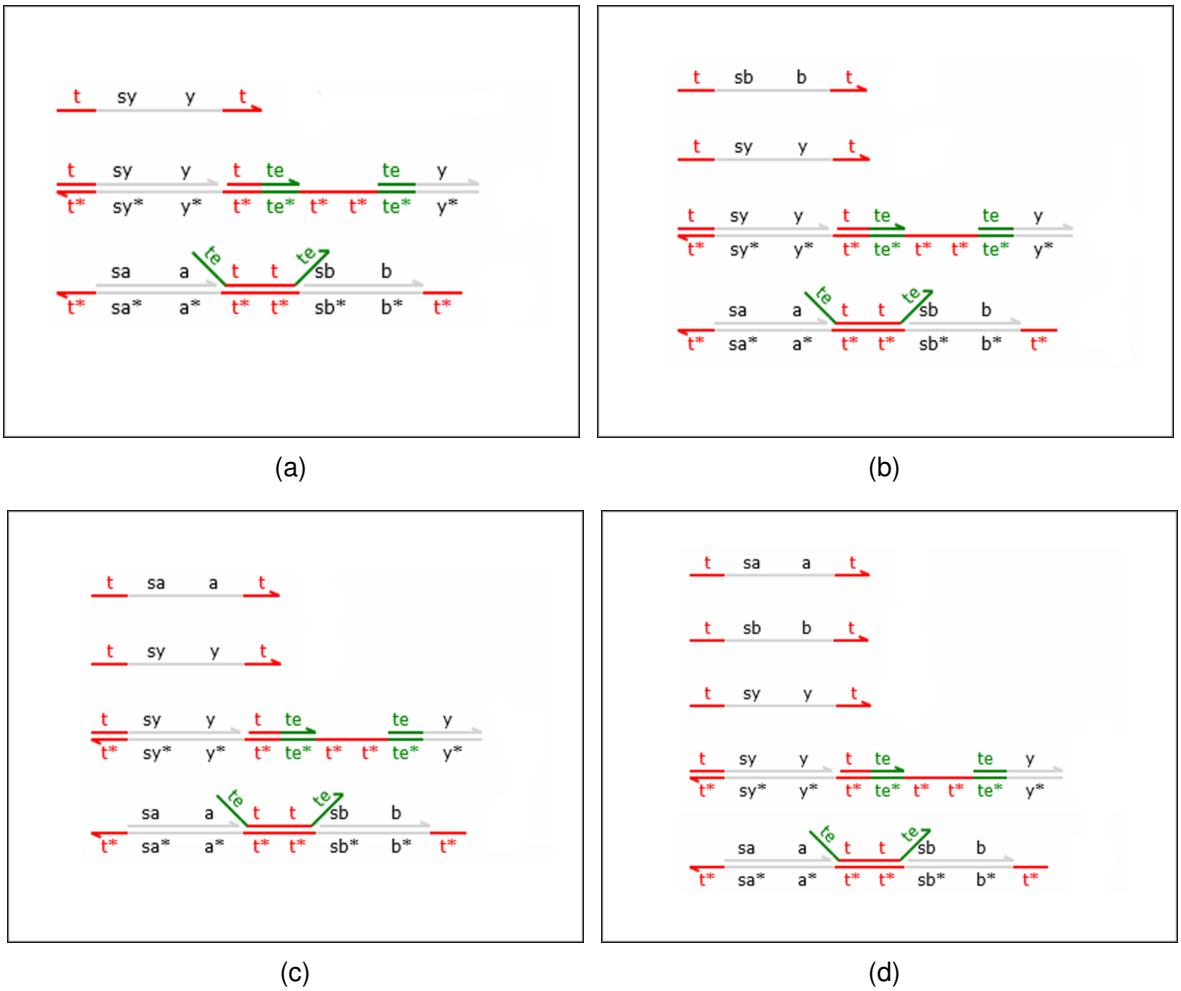


Figura A.9: Estados iniciales de la compuerta NOR: (a) sin entradas; (b) variable B presente; (c) variable A presente; (d) variables A y B presentes.

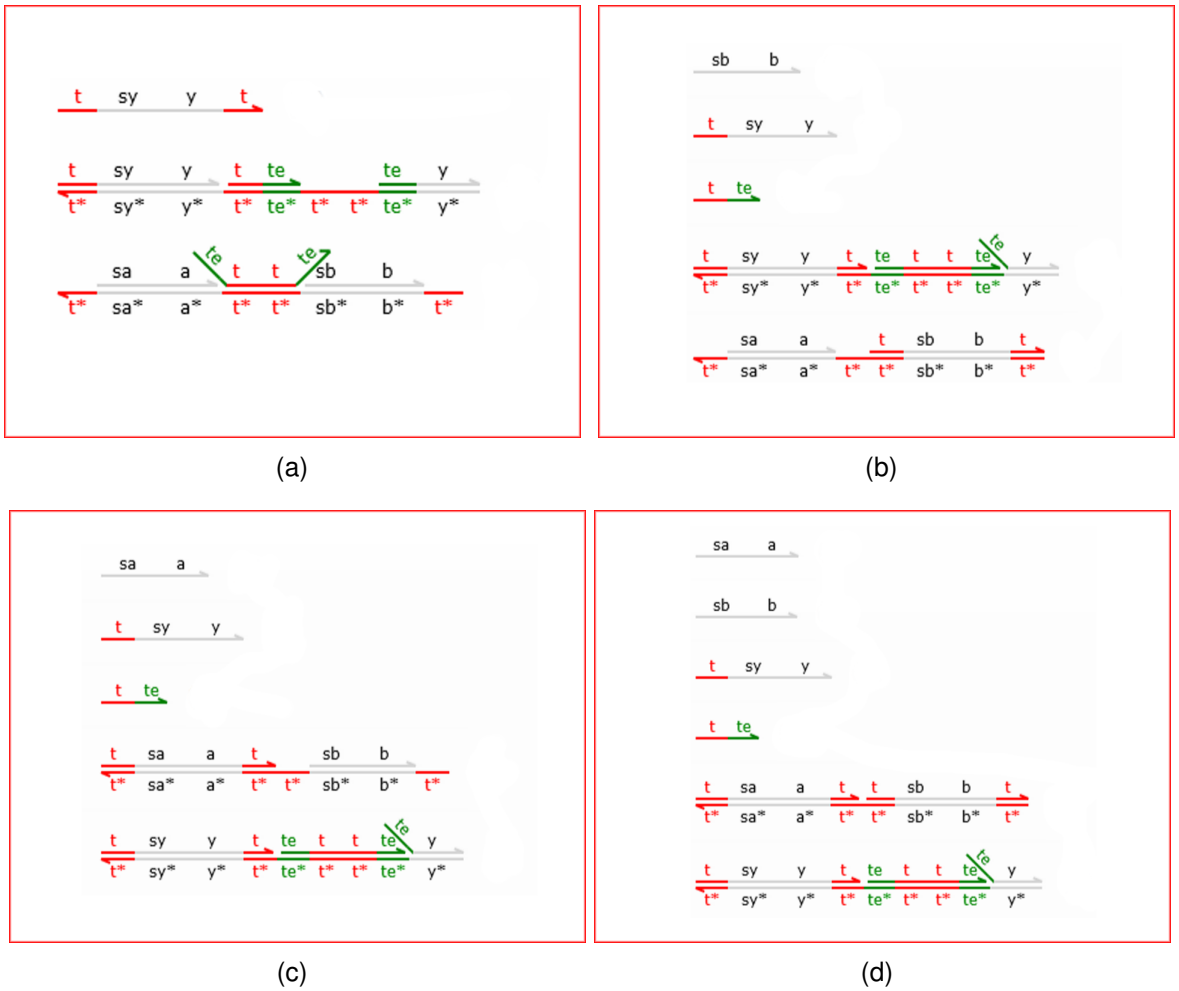
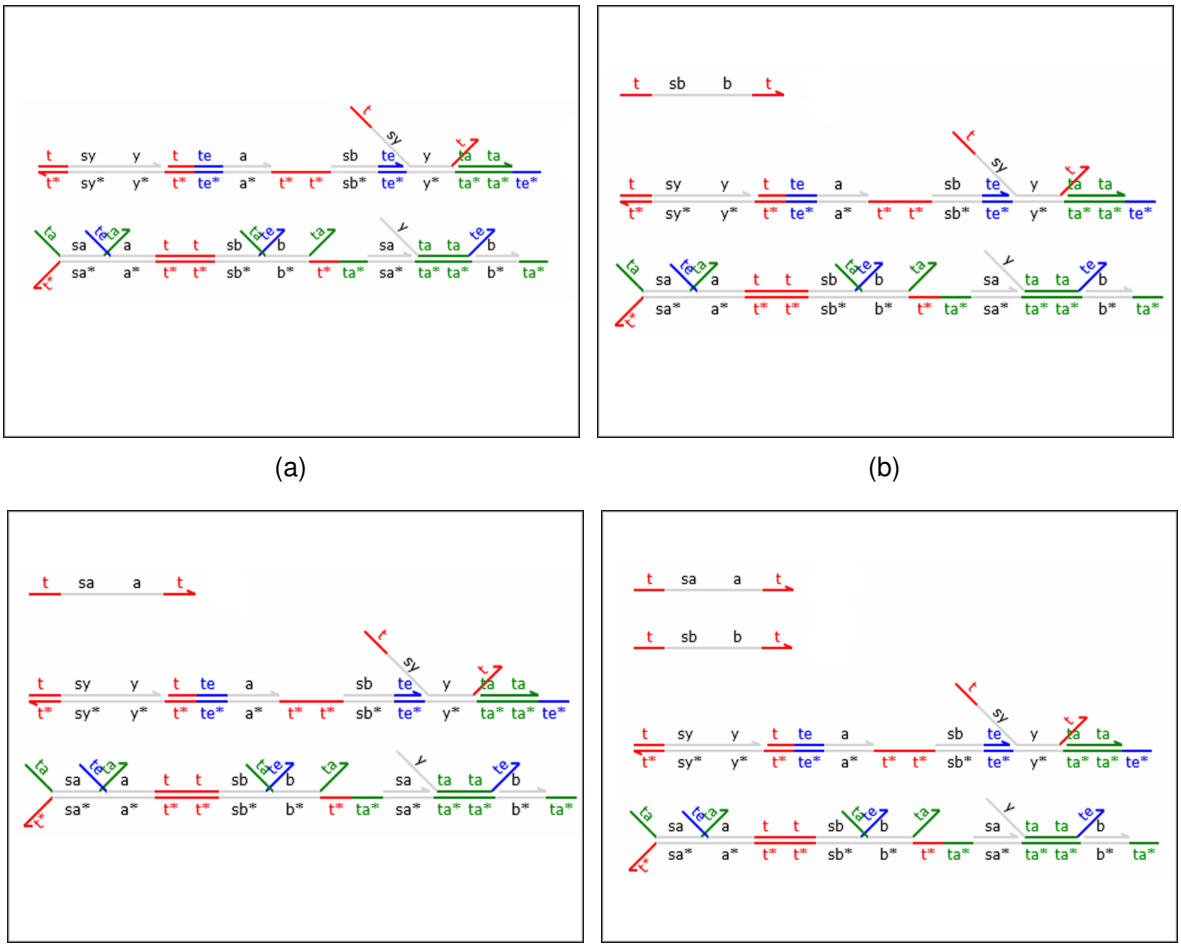


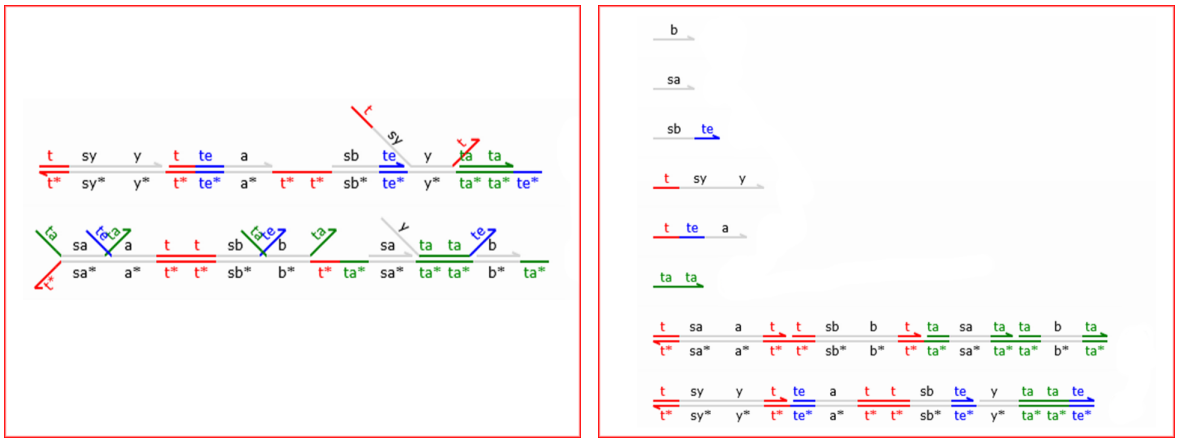
Figura A.10: Estados finales de la compuerta NOR: (a) estado final 1 ; (b) estado final 2 ; (c) estado final 3 y (d) estado final 4.



(a) (b)

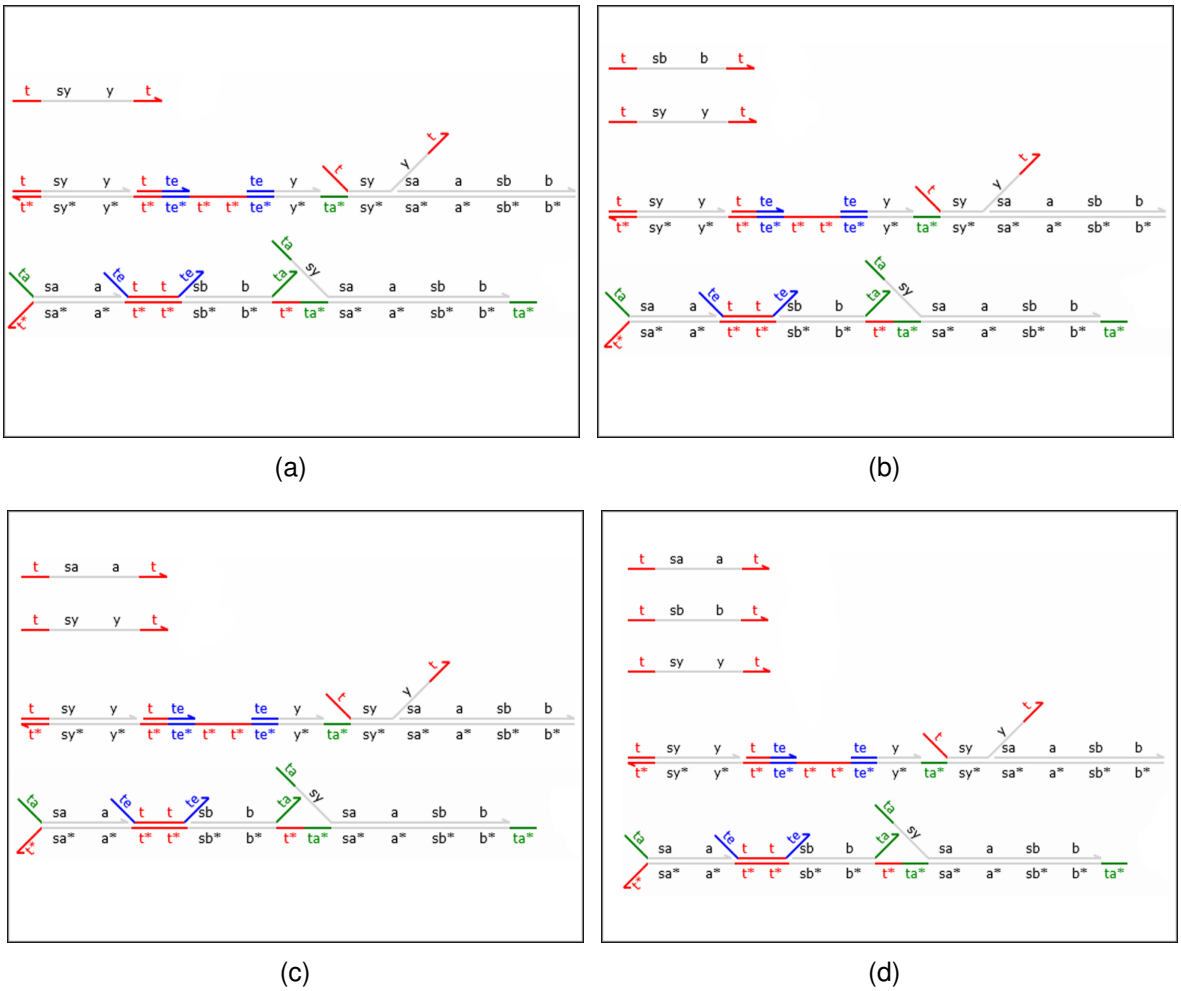
(c) (d)

Figura A.11: Estados iniciales de la compuerta XOR: (a) sin entradas; (b) variable B presente; (c) variable A presente; (d) variables A y B presentes.



(a) (b)

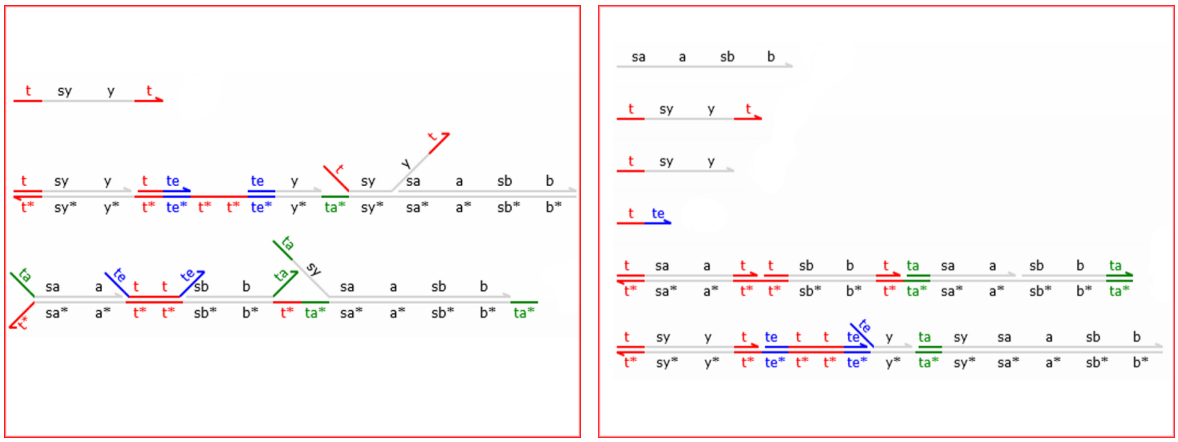
Figura A.12: Estados finales de la compuerta XOR: (a) estado final 1 y (b) estado final 2.



(a) (b)

(c) (d)

Figura A.13: Estados iniciales de la compuerta XNOR: (a) sin entradas; (b) variable B presente; (c) variable A presente; (d) variables A y B presentes.



(a) (b)

Figura A.14: Estados finales de la compuerta XNOR: (a) estado final 1 y (b) estado final 2.