TEC | Tecnológico de Costa Rica

Instituto Tecnológico de Costa Rica

Escuela de Ingeniería en Computadores
(Computer Engineering School)

Programa de Licenciatura en Ingeniería en Computadores
(Licentiate Degree Program in Computer Engineering)

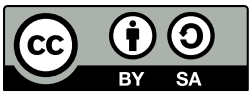
TEC | Tecnológico de Costa Rica

Aplicación de Técnicas de Aprendizaje Automático para Generación de Circuitos Aproximados

(Application of Machine Learning Techniques for the Generation of Approximate Circuits)

Informe de Trabajo de Graduación para optar por el título de Ingeniero en Computadores con grado académico de Licenciatura
(Report of Graduation Work in fulfillment of the requirements for the degree of Licentiate in Computer Engineering)

Ignacio Elías Vargas Campos
Cartago, junio, 2025



TEC | Tecnológico de Costa Rica

TEC – Escuela de Ingeniería en Computadores (CES)
Acta de Aprobación de Trabajo de Graduación

Con fundamento en lo que establece el "Reglamento de Trabajos Finales de Graduación del Instituto Tecnológico de Costa Rica", el Tribunal Examinador del Trabajo Final de Graduación, nombrado con el propósito de evaluar el proyecto:

Aplicación de Técnicas de Aprendizaje Automático para Generación de Circuitos Aproximados

Habiendo analizado el resultado general del trabajo presentado por el estudiante

Primer Apellido	Segundo Apellido	Nombre	No. de carné
Vargas	Campos	Ignacio Elías	2019053776

Emite el siguiente dictamen:

APROBADO

CALIFICACION: **100** puntos.

○ REPROBADO

○ SE RECOMIENDA ○ NO SE RECOMIENDA

Brindarle una nueva oportunidad para la DEFENSA PUBLICA de su Trabajo Final

NUEVA FECHA: _____

Dando fe de lo aquí expuesto firmamos

LUIS ALBERTO CHAVARRIA ZAMORA (FIRMA)
Firmado digitalmente por LUIS ALBERTO CHAVARRIA ZAMORA (FIRMA)
Fecha: 2025.07.08 14:57:10 -06'00'

MSc. Luis Alberto Chavarría Zamora
Profesor Asesor

RONALD EDUARDO GARCIA FERNANDEZ (FIRMA)
Digitally signed by RONALD EDUARDO GARCIA FERNANDEZ (FIRMA)
Date: 2025.07.04 16:27:46 +02'00'

Lic. Ronald García Fernández
Profesor Lector

Pablo David Garcia Brenes
Digitally signed by Pablo David Garcia Brenes
Date: 2025.06.27 15:34:10 -06'00'

Ing. Pablo García Brenes
Lector

27 de junio de 2025

Dedicatoria

Dedico este trabajo a mis padres,
Ana Beatriz y Pablo.
Gracias a su apoyo y sacrificio
completo esta etapa de vida.

Agradecimientos

A mis camaradas más cercanos que me
ayudaron, apoyaron y enseñaron tanto.
A aquellos profesores que me guiaron
y alimentaron mi curiosidad.

Resumen

Este trabajo integra por primera vez técnicas de aprendizaje automático en la herramienta de síntesis lógica aproximada AxLS para facilitar la generación de circuitos aproximados. Se identificó y justificó la elección de implementar un método basado en árboles de decisión por su rapidez de entrenamiento, facil mapeo a Verilog y relevancia en la literatura. Se extendió AxLS con una clase para el fácil entrenamiento de árboles de decisiones y generación de módulos de Verilog aproximados, posibilitando comparaciones directas en métricas de error, área y tiempo de ejecución. Para evaluar la solución se diseñaron experimentos automáticos sobre un conjunto de circuitos de referencia, cuantificando la tasa de error, la reducción de área y la duración del proceso de síntesis.

Los resultados muestran que el método basado en árboles de decisión es competitivo frente a los métodos de poda existentes en AxLS y puede descubrir soluciones más eficientes en diversos casos. Sin embargo, presenta menor control sobre el error introducido y, dependiendo del circutio, no siempre logra generar aproximaciones más compactas que el diseño original.

Esta implementación no solo amplía las capacidades de AxLS, sino que sienta las bases para incorporar otras técnicas de ML y fomentar la adopción de técnicas ML dentro de flujos de ALS.

Palabras clave Diseño electrónico automatizado, síntesis lógica aproximada, aprendizaje automático, árboles de decisión.

Abstract

This work integrates machine learning techniques into the AxLS approximate logic synthesis tool to enable the generation of approximate circuits. A decision tree based method was selected and justified due to its fast training, straightforward mapping to Verilog, and strong presence in the literature. AxLS was extended with a class for easy training of decision trees and automated generation of approximate Verilog modules, allowing direct comparisons in error, area, and synthesis time metrics. To evaluate the approach, automated experiments were conducted on a set of benchmark circuits, measuring error rates, area reduction, and synthesis duration.

Results show that decision trees are competitive with existing pruning methods in AxLS and can yield more efficient solutions in several cases. However, they provide less control over the introduced error and may not always produce smaller approximations than the original design, depending on the circuit.

This work extends the AxLS framework and opens the door to incorporating a wider range of ML based approaches into ALS workflows.

Key words Electronic Design Automation, Approximate Logic Synthesis, Machine Learning, Decision Trees.

Índice general

Índice de tablas

Índice de figuras

Siglas y acrónimos

- ALS: Síntesis de lógica aproximada (*Approximate Logic Synthesis*)
- ML: Aprendizaje automático (*Machine Learning*)
- DT: Árbol de decisión (*Decision Tree*)
- RF: Bosque aleatorio (*Random Forest*)
- LUT: Tabla de búsqueda (*Lookup Table*)
- DL: Aprendizaje profundo (*Deep Learning*)
- PGNN: Red neuronal de grafos con pesos en los caminos (*Path-weighted Graph Neural Network*)
- MCTS: Búsqueda en árbol de Monte Carlo (*Monte Carlo Tree Search*)
- CGP: Programación genética cartesiana (*Cartesian Genetic Programming*)
- CAD: Diseño asistido por computador (*Computer-aided design*)
- EDA: Diseño electrónico automático (*Electronic Design Automation*)
- SOP: Suma de productos (*Sum of products*)
- RL: Aprendizaje reforzado (*Reinforcement Learning*)
- BDD: Diagrama de decisión binario (*Binary Decision Diagram*)
- ER: Tasa de error (*Error Rate*)
- MHD: Distancia de Hamming promedio (*Mean Hamming Distance*)
- WHD: Peor distancia de Hamming (*Worst Hamming Distance*)
- WCE: Error en el peor caso (*Worst Case Error*)
- MAE: Error absoluto medio (*Mean Absolute Error*)
- MRE: Error relativo medio (*Mean Relative Error*)
- MSE: Error cuadrático medio (*Mean Squared Error*)
- API: Interfaz de interacción programática (*Application Programming Interface*)

- CLI: Interfaz de línea de comandos (*Command Line Interface*)
- SAIF: Formato de Intercambio de Actividad de Conmutación (*Switching Activity Interchange Format*)
- VCD: Registro de Cambio de Valores (*Value Change Dump*)
- SSIM: Medida del índice de similitud estructural (*Structural similarity index measure*)

1 Introducción

Este proyecto consiste en la aplicación de técnicas de aprendizaje automático (Machine Learning, ML) al campo de la Síntesis Lógica Aproximada (Approximate Logic Synthesis, ALS). Este informe detalla el proceso para la integración de una técnica del estado del arte de ML en la herramienta de fuente abierta AxLS.

En este capítulo se presenta el contexto y la motivación detrás del desarrollo del proyecto. Se describen los antecedentes relevantes, el problema identificado, así como los objetivos y alcances planteados para dar respuesta a dicho problema.

1.1. Antecedentes del proyecto

En esta sección se recopila información sobre la organización involucrada, el área temática del proyecto y trabajos relacionados que han abordado problemáticas similares. Esto permite ubicar el proyecto dentro de un marco de referencia académico.

1.1.1. Descripción de la organización

Se realizará en colaboración con la escuela de Ingeniería Electrónica del Instituto Tecnológico de Costa Rica, específicamente con el laboratorio de "Efficient Computing Across the Stack" (ECASLab).

El Tecnológico de Costa Rica es una de las universidades públicas de Costa Rica, con su sede original en Cartago creada en 1971 [1].

1.1.2. Descripción del área de conocimiento del proyecto

Este proyecto trata temas en la intersección de las áreas de computación aproximada y ML. Específicamente en el campo generación de circuitos aproximados, a la que se le llama Síntesis de Lógica Aproximada.

1.1.3. Trabajos similares encontrados

En [2], Castro-Godínez et al. presentan una herramienta de fuente abierta para ALS llamada AxLS. Este framework implementa múltiples técnicas de ALS y permite utilizar los mismos métodos para calcular el umbral de error y métricas de calidad en ellas, lo que permite una mejor comparación de los diferentes enfoques de ALS. Este trabajo es fundamental para este proyecto, ya que se realizó sobre herramienta AxLS, aportando un enfoque de ML a la herramienta.

La intersección de aplicar técnicas de ML para ALS es un nicho que ha surgido principalmente en los últimos 5 años. Dentro de esta área se han identificado 3 categorías principales de aplicación de ML a ALS.

La primera categoría identificada es sobre métodos que buscan entrenar modelo que funcione como "asistente" en otros métodos de ALS, evaluando más rápidamente los efectos que causaría generar cambios en un circuito, usualmente ayudando a evaluar error rápidamente. El beneficio es que en muchos métodos se generan cambios a prueba y error o con heurísticas, pero se debe evaluar el error introducido por cada cambio lo cual puede conllevar simulaciones costosas computacionalmente, pero un modelo puede estimar el error con una exactitud aceptable mucho más rápido. Dentro de esta categoría se encuentran los trabajos de Pasandi et al. [3] [4], en donde aplica técnicas de aprendizaje reforzado y aprendizaje profundo (Deep Learning, DL), respectivamente, para estimar el error generado en un circuito al hacer cambios locales. En [5], Ye et al. no estiman directamente el error con ML, pero entrenan un agente de aprendizaje reforzado para realizar cambios locales en un circuito y los estados de este agente son representados con una red neuronal de grafos con pesos en los caminos (Path-weighted Graph Neural Network, PGNN).

La segunda categoría identificada es sobre métodos que se enfocan en técnicas de ML para realizar exploraciones del espacio de diseño de un circuito. Específicamente suelen emplear una técnica llamada Búsqueda en Árbol de Monte Carlo (Monte Carlo Tree Search, MCTS). El trabajo de Rajput et al. [6] plantea la aplicación de MCTS para ALS, modificando el algoritmo levemente para que pueda explorar nodos más profundos en el árbol y utilizando la reducción de área como recompensa para evaluar nodos en el árbol. También se cuenta con [7], donde Awais et al. también aplican MCTS al problema de ALS utilizando ML para la estimación de error, lo cual lo hace también un ejemplo de la primera categoría de aplicaciones de ML a ALS mencionada.

La tercera categoría identificada conlleva realizar un entrenamiento supervisado sobre las entradas y salidas de un circuito, el modelo de ML entrenado es seguidamente mapeado a un circuito. Ya que los modelos de aprendizaje supervisado aprenden a generalizar una función, al ser mapeados a un circuito se obtiene un circuito que aproxima al original.

Uno de los primeros ejemplos en esta categoría es [8], donde Boroumand et al. desarrollan un método para aprender funciones lógicas de ejemplos y sintetizar circuitos basado en el modelo aprendido, a lo que le llaman síntesis lógica a partir de ejemplos. En [9], De Abreu et al. exploran el uso de árboles de decisiones (Decision Tree, DT) como alternativa tanto para optimizar circuitos exactos como para generar versiones aproximadas de los circuitos. En ambos casos la técnica es exitosa, reduciendo el tiempo de ejecución para optimizar los circuitos exactos al compararse las alternativas comunes ABC y Espresso, así como siendo capaz de generar circuitos aproximados de profundidad y área sumamente reducidos e igual logrando buenos niveles de exactitud. Miayasaka et al. prueban y comparan varios métodos populares de aprendizaje supervisado en [10], incluyendo redes neuronales, árboles de decisiones y redes de tablas de búsqueda (Lookup Table, LUT). Evalúan estos métodos en su capacidad de aproximar circuitos lógicos y aritméticos, notando que estos modelos populares no pueden aprender de manera efectiva ciertos tipos de circuitos aritméticos.

Los resultados del concurso del International Workshop on Logic & Synthesis (IWLS) del 2020 [11] han sido sumamente claves en el avance de esta área. El concurso consistía en implementar 100 funciones booleanas dados ejemplos incompletos de sus tablas de verdad, luego las implementaciones fueron validadas con el set de ejemplos de validación que no fue proporcionado a los concursantes. Entre las técnicas evaluadas se encuentran DT, bosques aleatorios (Random Forest, RF), redes de LUT, Espresso, redes neuronales y programación genética cartesiana (Cartesian Genetic Programming, CGP).

Las principales conclusiones del análisis de resultados son que ninguna técnica dominó todas las pruebas; la mayoría de los equipos, incluyendo el ganador, emplearon un conjunto de técnicas diferentes. Los bosques aleatorios y los árboles de decisión fueron muy populares y constituyen un punto de referencia sólido para ALS. Y muy importantemente, sacrificar un poco de exactitud permite una reducción significativa en el tamaño del circuito.

Zeng et al. [12] explora con mayor profundidad el uso de DT para ALS, utilizando una alteración de la técnica llamada árbol de decisión adaptable, particularmente aplicando variaciones guiadas por una métrica llamada "Shapley Additive Explanations" (SHAP), que busca explicar la importancia de las características de entrada a un modelo. En la misma línea de explorar variaciones a la técnica de DT, Huang y Jiang [13] exploran el uso de grafos de decisión para relajar las limitaciones estructurales de un árbol, como el crecimiento exponencial a medida que aumenta la complejidad. Otra variación novedosa de los DT es la propuesta de Hu y Cai [14], donde aplican una técnica a la que llaman árboles de decisión óptimos, proporcionando una garantía de optimalidad, lo que les permite un mejor control en el balance entre precisión y complejidad del circuito.

También se ha profundizado en la técnica de CGP. Berndt et al. [15] proponen recibir una tabla de verdad que representa el comportamiento deseado y evolucionar circuitos para ajustarse a ese comportamiento, partiendo de circuitos aleatorios o de un circuito previamente especificado. Una ventaja de esta técnica es que puede combinarse con otras, utilizando circuitos generados por otras técnicas como punto de partida en el proceso evolutivo.

Prats Ramos et al. [16] presentan un análisis comparativo entre las técnicas de DT, CGP y un enfoque mixto de ML que fue originalmente presentado por el equipo ganador del concurso de IWLS 2020 [11] y combina el uso de redes neuronales y LUT.

1.2. Planteamiento del problema

En esta sección se describe el problema central que motiva este proyecto. Se presenta el contexto en el que surge, su justificación desde una perspectiva investigativa, y se formula claramente el problema que se busca resolver.

1.2.1. Contexto del problema

Los investigadores del Instituto Tecnológico de Costa Rica enfrentan limitaciones en el uso de técnicas de ALS basadas en ML. Actualmente, la herramienta disponible para

ellos, AxLS, no cuenta con las capacidades técnicas necesarias para implementar estas técnicas, lo que restringe sus posibilidades de experimentación y análisis.

1.2.2. Justificación del problema

La capacidad de utilizar técnicas de ALS basadas en ML permitiría a los investigadores verificar y comparar resultados obtenidos por otros grupos de investigación. Además, facilitaría el diseño y desarrollo de técnicas novedosas de ALS basadas en ML, permitiéndoles contribuir de manera más significativa en la investigación en ALS.

1.2.3. Enunciado del problema

Los investigadores del Instituto Tecnológico de Costa Rica carecen de una herramienta que les permita aplicar técnicas de ALS basadas en aprendizaje automático. La herramienta disponible, AxLS, no posee las capacidades técnicas necesarias para ello, lo que impide la validación, comparación y desarrollo de nuevas técnicas en este campo.

1.3. Objetivos del proyecto

1.3.1. Objetivo General

Formular una técnica del estado del arte de aprendizaje automático para ALS en la herramienta AxLS, de manera que permita ser comparada con otras técnicas implementadas dentro de la herramienta así como con otras implementaciones diferentes de las técnicas seleccionadas.

1.3.2. Objetivos Específicos

1. Evaluar al menos 3 técnicas de ALS basadas en aprendizaje automático determinando cuál es la más apropiada para ser integrada en la herramienta AxLS.

2. Adaptar la herramienta AxLS para que sea posible la implementación de la técnica escogida, con la consideración de que permita más fácilmente la implementación de futuras técnicas de aprendizaje automático.

3. Evaluar los resultados de la técnica de aprendizaje automático implementada en AxLS con respecto a resultados obtenidos mediante las técnicas ya existentes en la herramienta.

1.4. Alcances, entregables y limitaciones del proyecto

El alcance de este proyecto incluye investigar y evaluar las técnicas del estado del arte, escoger e implementar una en la herramienta AxLS y documentar el proceso. Los entregables asociados a este alcance, las técnicas y herramientas a emplear así como sus estrategias de verificación se presentan en la Tabla 1.1.

Como limitación, la implementación escogida está restringida a las capacidades actuales de AxLS, incluyendo las métricas y conjunto de pruebas con las que cuenta actualmente. Este proyecto no plantea agregar más pruebas a la herramienta. Otra limitación del proyecto es la falta de acceso a máquinas con capacidades computacionales de alto rendimiento, por lo que la técnica seleccionada no puede requerir un entrenamiento demasiado pesado. Debe ser factible su entrenamiento en poco tiempo en un computador personal.

Tabla 1.1: Entregables del proyecto.

Entregable	Técnicas / Herramientas	Estrategias de verificación	Objetivo Específico
Documento de diseño que justifica la escogencia de la técnica de aprendizaje automático a implementar en AxLS. Compara todas las técnicas que estén bajo consideración inicialmente de aquellas encontradas en trabajos relacionados del estado del arte.	Zotero para manejo de referencias.	Documento detalla y escoge una técnica en específico, da razones de por qué se escogió tomando en cuenta la factibilidad y relevancia en el estado del arte.	1
Documento de diseño que explica el plan de implementación de la técnica escogida dentro de la herramienta AxLS.	AxLS, Python, Yosys.	Documento detalla todos los cambios de API necesarios en AxLS y la API para utilizar la técnica nueva, con ejemplos de uso y un plan de pruebas. Detalla el proceso de crear los datos de entrenamiento, entrenamiento del modelo y sintetización del circuito.	2
Código adaptado de herramienta AxLS con técnica de ML y adaptaciones necesarias integradas.	AxLS, Python, Yosys, git, Github, biblioteca ML.	La herramienta AxLS permite ejecutar cualquiera de las técnicas previamente existentes y la técnica ML nueva. La ejecución correcta se validará con un plan de pruebas.	2
Informe final sobre el proyecto realizado y los resultados encontrados al comparar la técnica de ML con las ya existentes en la herramienta AxLS	LaTeX, IEEE, AxLS	El informe describe cómo se llevó a cabo el proyecto, la técnica de ML empleada y presenta los resultados de esta técnica comparados con los de las otras técnicas ya existentes en la máquina con gráficos.	3

2 Marco de referencia teórico

Este capítulo introduce los conceptos teóricos necesarios para respaldar este proyecto, el cual se sitúa en la intersección entre las áreas de ALS y ML. La integración de estas disciplinas permite explorar nuevos métodos para la generación de circuitos digitales que intercambien exactitud por eficiencia y complejidad.

Se revisan los conceptos básicos de la computación aproximada y las técnicas tradicionales de ALS, con el objetivo de contrastarlas con los enfoques basados en ML. Luego se introducen conceptos fundamentales de ML, necesarios para entender los mecanismos mediante los cuales un modelo puede ser útil en la generación de circuitos aproximados, siendo especialmente relevante su capacidad para aprender y generalizar funciones booleanas. Finalmente, se abordan las técnicas de ML más valiosos en la aplicación de ALS, junto con los métodos para transformar los modelos de ML a circuitos lógicos, paso comúnmente necesario para su aplicación práctica dentro del marco de ALS.

La Figura 2.1 mapea los conceptos tratados en este capítulo en un mapa conceptual.

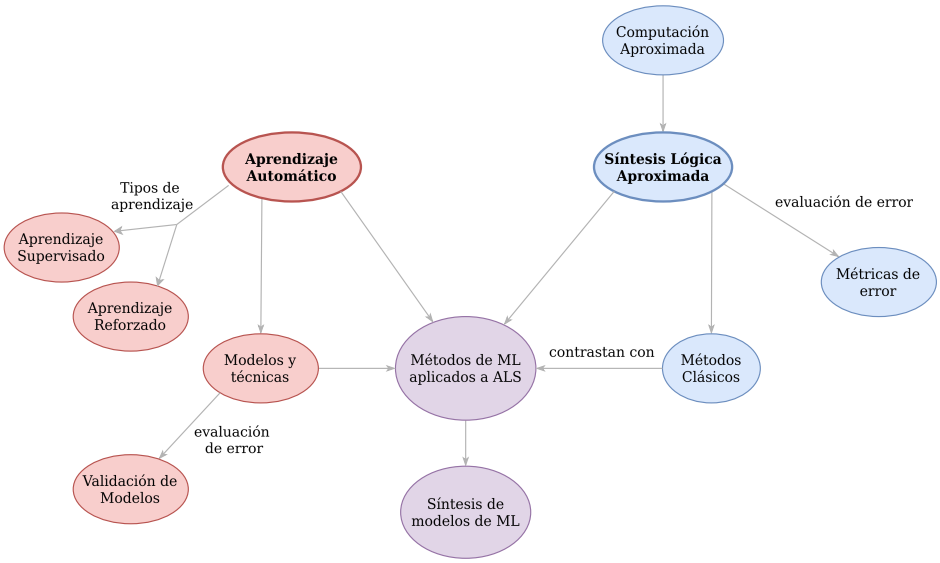


Figura 2.1: Mapa conceptual de los temas abordados en el marco teórico. Las burbujas rojas corresponden a temas asociados a ML, las burbujas azules corresponden a temas asociados a ALS y las burbujas moradas corresponden a temas que integran las 2 áreas.

2.1. Computación aproximada

En esta sección se explican los conceptos de computación aproximada más fundamentales para este trabajo.

La computación aproximada surgió como un paradigma para diseñar sistemas digitales más eficientes en consumo de energía y rendimiento. Muchos sistemas y aplicaciones son tolerantes a errores en los resultados computados; por ejemplo, el procesamiento multimedia (audio, video, gráficos e imágenes), el reconocimiento de patrones y la minería de datos. Al relajar la necesidad de realizar operaciones exactas y determinísticas, las técnicas de computación aproximada permiten mejorar significativamente la eficiencia de los circuitos. [17]

Se denomina circuito aproximado a este tipo de circuito digital que sacrifica precisión en sus resultados a cambio de una mayor eficiencia en su rendimiento, área, consumo de potencia o una combinación de estas métricas.

2.1.1. Síntesis lógica aproximada

La construcción automática de circuitos digitales aproximados se realiza mediante herramientas de diseño asistido por computador (Computer-Aided Design, CAD), disponibles en entornos de diseño electrónico automatizado (Electronic Design Automation, EDA). ALS se enfoca en mejorar el comportamiento de un circuito (ya sea en consumo de potencia, área o rendimiento) sin necesidad de conocer sus detalles de implementación. Este proceso se lleva a cabo de forma sistemática, modificando la funcionalidad especificada sin exceder un umbral de error predefinido [18].

Existen métodos para crear circuitos aproximados manualmente, que no caen dentro de la categoría de ALS, ya que no son automáticos. Este tipo de diseño requiere un profundo conocimiento del comportamiento del circuito para poder hacer aproximaciones que no degraden demasiado su precisión. Este nivel alto de pericia requerida sobre la topología del circuito a aproximar es la principal desventaja de estos métodos y el porqué en los últimos años la investigación se ha enfocado hacia métodos de ALS. [18]

Los métodos de optimización de circuitos digitales se suelen dividir en métodos para dos o para múltiples niveles de profundidad lógica. En este caso la profundidad lógica hace referencia a la cantidad máxima de compuertas lógicas AND u OR por las que debe pasar una señal de la entrada a la salida del circuito [19].

Ya que los circuitos de dos niveles tienen una profundidad lógica fija, su optimización se enfoca en reducir la cantidad de operaciones en la representación de suma de productos (Sum Of Products, SOP) del circuito. Para circuitos de múltiples niveles, el objetivo de optimización depende de la estructura booleana utilizada para representarlos y optimizarlos, pero los métodos suelen buscar reducir el número de compuertas lógicas y la profundidad del circuito. Los métodos de ML se destacan en este contexto, ya que pueden manejar circuitos de profundidad arbitraria y son particularmente útiles para optimizar circuitos complejos, lo que los convierte en una herramienta valiosa para la optimización multi-nivel [18].

En este contexto, en los últimos años se ha explorado con gran éxito la aplicación

de técnicas de ML para ALS [3], [11], [16], [20]. Gracias a su capacidad de generalizar funciones, el aprendizaje de funciones booleanas ha sido una de las técnicas de ML más exploradas y útiles para la generación de circuitos aproximados. Esta efectividad se debe a que, al aprender una función, el modelo busca cubrir todas las posibles soluciones de manera eficiente. Así, logra un buen equilibrio entre precisión y rendimiento, lo que permite crear circuitos eficientes sin perder demasiada exactitud.

2.1.2. Métricas de error

Independientemente de la metodología utilizada para ALS, es necesario definir una manera de calcular el error insertado en la síntesis.

Existen dos categorías principales de métricas: de propósito general, que están relacionadas con la ocurrencia del error, y métricas aritméticas, que están relacionadas con la magnitud del error.

Algunos métodos de ALS calculan el error introducido de manera exacta, mientras que otros lo estiman de manera aproximada. Los métodos para calcularlo de manera exacta suelen ser más complejos y se pueden basar en el problema de satisfaciabilidad booleana, diagramas de decisión binarios (Binary Decision Diagram, BDD), álgebra simbólica computacional o incluso revisar todas las posibles combinaciones de entradas, con tablas de verdad o simulaciones exhaustivas. La estimación aproximada del error se suele implementar aplicando simulaciones Monte Carlo, pero existen otras técnicas [18]. De particular interés para este trabajo, se han aplicado técnicas de ML para la estimación de error de manera aproximada; por ejemplo, en [4] entrenan un modelo de DL para predecir el error en las salidas primarias de un circuito al quitar o reemplazar compuertas lógicas en la red del circuito.

La decisión de cuál metodología utilizar para el cálculo de error debe tomar en cuenta el balance entre la calidad de los resultados y el esfuerzo computacional. Un cálculo de error preciso suele resultar en mejores circuitos aproximados, porque lleva a mejores decisiones de cuáles aproximaciones aplicar durante el proceso de ALS. Sin embargo, un cálculo preciso puede presentar costos computacionales altos, especialmente al ejecutarlos sobre cada posible modificación a la topología del circuito [18]. Además, el costo computacional suele crecer exponencialmente con la complejidad y la cantidad de entradas del circuito.

El resto de esta sección introducirá algunas de las métricas de error más comúnmente utilizadas para ALS. Se definen a continuación la notación y variables:

- n: Cantidad de bits de entrada del circuito.
- N: Cantidad de combinaciones de entradas consideradas.
- $f(x)$: Salida del circuito original.
- $f'(x)$: Salida del circuito aproximado.
- $f_i(x)$: Bit de salida en la posición i del circuito original.
- $f'_i(x)$: Bit de salida en la posición i del circuito aproximado.

Tasa de error

La tasa de error (Error Rate, ER) corresponde con la probabilidad de observar uno o más bits erróneos en la salida. ER es la métrica más comúnmente utilizada en la aproximación de circuitos, siendo aplicada en circuitos tanto generales como aritméticos [18]. El cálculo del ER se presenta en la ecuación (2.1),

$$ER = \frac{1}{N} \sum_{\forall x \in N} f(x) \neq f'(x). \tag{2.1}$$

Distancia de Hamming promedio

La distancia de Hamming promedio (Mean Hamming Distance, MHD), también llamada tasa de error de bits, representa la cantidad de bits de salida erróneos en promedio. Esta tasa fue propuesta como una variación de ER para evaluar aproximaciones con múltiples salidas. El cálculo del MHD se presenta en la ecuación (2.2),

$$MHD = \frac{1}{N} \sum_{\forall x \in N} \sum_{i=0}^{n-1} f_i(x) \oplus f'_i(x). \tag{2.2}$$

Peor distancia de Hamming

La peor distancia de Hamming (Worst Hamming Distance, WHD) representa la mayor distancia de Hamming entre las salidas originales y las aproximadas. Se utiliza para verificar cuál es el peor caso en cantidad de bits de salida erróneos. El cálculo del WHD se presenta en la ecuación (2.3),

$$WHD = \max_{\forall x \in N} \sum_{i=0}^{n-1} f_i(x) \oplus f'_i(x). \tag{2.3}$$

Error en el peor caso

El error en el peor caso (Worst Case Error, WCE) representa la mayor diferencia absoluta en magnitud entre las salidas del circuito original y el aproximado. Es la métrica más comúnmente utilizada en el diseño aritmético [18]. El cálculo del WCE se presenta en la ecuación (2.3),

$$WHD = \max_{\forall x \in N} \left| f(x) - f'(x) \right|. \tag{2.4}$$

Error absoluto medio

El error absoluto medio (Mean Absolute Error, MAE) representa la diferencia en magnitud media entre las salidas del circuito original y el aproximado. El cálculo del MAE se presenta en la ecuación (2.5),

$$MAE = \frac{1}{N} \sum_{\forall x \in N} \left| f(x) - f'(x) \right|. \tag{2.5}$$

Error relativo medio

El error relativo medio (Mean Relative Error, MRE) es similar al MAE, pero el resultado se presenta como un porcentaje y es relativo a las salidas del circuito original. Formalmente, representa la diferencia en magnitud media relativa entre las salidas del circuito original y el aproximado. La ventaja de utilizar un método de error relativo es que, como el valor de error es un porcentaje, es posible analizar circuitos con cantidades de entradas diferentes.

El cálculo del MRE se presenta en la ecuación (2.6). Se nota que si $f(x)$ tiene un valor de 0, la ecuación se vuelve indefinida; por lo tanto, para aplicaciones prácticas se recomienda reemplazar el divisor con la función máx $(1, |f(x)|)$.

$$MRE = \frac{1}{N} \sum_{\forall x \in N} \frac{|f(x) - f'(x)|}{|f(x)|} \tag{2.6}$$

Error cuadrático medio

El error cuadrático medio (Mean Squared Error, MSE) representa la diferencia en magnitud cuadrática media entre las salidas del circuito original y el aproximado. Se presenta su cálculo en la ecuación (2.7). Aunque el MAE suele ser más intuitivo de interpretar, el MSE ofrece una visión más completa, ya que corresponde con el concepto de varianza de los errores. Además, el MSE es computacionalmente menos costoso de calcular, lo que lo hace especialmente útil cuando se están manejando grandes cantidades de datos.

$$MSE = \frac{1}{N} \sum_{\forall x \in N} \left(f(x) - f'(x) \right)^2 \tag{2.7}$$

2.2. Aprendizaje automático

Según Russel y Norvig [21], en el aprendizaje automático una computadora analiza datos, construye un modelo basado en esos datos y luego usa ese modelo tanto como una hipótesis sobre el mundo como una herramienta de software para resolver problemas.

En el contexto de ALS, una de las formas que más nos interesa aplicar ML es que la computadora analice los datos de la tabla de verdad de un circuito, de modo que se construya una representación generalizada del comportamiento de dicho circuito. Este modelo actúa como una hipótesis sobre lo que haría el circuito original, y aunque normalmente en otros enfoques de ML se mantendría como una solución en software, en nuestro caso buscamos transformarlo en hardware. De esta forma, el modelo sigue funcionando como una hipótesis de lo que haría el circuito original, pero implementada directamente como un circuito aproximado.

2.2.1. Aprendizaje supervisado

En el aprendizaje supervisado, el agente observa parejas de entradas y salidas y aprende una función que mapea las entradas a sus correspondientes salidas [21].

Es decir, en el aprendizaje supervisado, se dispone de los datos del comportamiento que se desea aprender antes del entrenamiento.

Un ejemplo de esto, en el contexto de ALS, es cuando se entrena un modelo utilizando la tabla de verdad de un circuito, que es un conjunto de datos que muestra las salidas correspondientes a sus entradas.

2.2.2. Aprendizaje reforzado

El aprendizaje reforzado (Reinforcement Learning, RL) es aprender qué hacer o cómo asociar situaciones con acciones, para maximizar una señal de recompensa. No se le dice al agente qué acciones tomar, sino que debe descubrir cuáles acciones dan la mayor recompensa mediante prueba y error [22].

En el contexto de ALS, un agente RL podría ser entrenado para aprender a modificar un circuito, realizando modificaciones y recibiendo una recompensa basada en parámetros como la reducción en área, en tiempo de ejecución, o el aumento en el error del circuito. Así el agente, después de mucho entrenamiento se convertirá en un experto en realizar modificaciones a circuitos para generar versiones aproximadas de ellos.

2.2.3. Generalización

Según Bishop [23], en el contexto de ML, la generalización es la capacidad de un modelo para producir respuestas correctas ante entradas no vistas durante el entrenamiento. En la práctica, el espacio de posibles entradas suele ser tan amplio que el conjunto de entrenamiento solo representa una pequeña fracción del total.

Por ejemplo, un modelo entrenado para distinguir entre fotos de gatos y perros no puede haber visto todas las imágenes de estos animales en la existencia; debe aprender patrones generales (es decir, debe generalizar), para poder clasificar imágenes nuevas.

En el contexto de la síntesis lógica aproximada (ALS), si bien es posible generar todas las combinaciones de entrada/salida de la tabla de verdad para circuitos pequeños, esto se vuelve inviable a medida que el número de entradas crece. Un circuito con 64 bits de entrada tiene 2^{64} combinaciones posibles, aproximadamente $1{,}8 \times 10^{19}$, lo que hace imposible utilizar su tabla de verdad completa. Por tanto, el modelo debe aprender a generalizar a partir de un subconjunto representativo. Incluso en circuitos pequeños, donde es posible entrenar modelos con una tabla de verdad exhaustiva, el modelo no memoriza los datos, sino que generaliza al construir una representación abstracta del circuito basada en patrones presentes en los ejemplos.

2.2.4. Sobreajuste

Se dice que una función aprendida tiene sobreajuste cuando presta demasiada atención al conjunto de datos con el que fue entrenada, lo que provoca un mal desempeño con

datos no vistos [21].

En el contexto de aprendizaje de circuitos, el sobreajuste puede ocurrir si un modelo memoriza las entradas y salidas de un subconjunto limitado de la tabla de verdad, sin capturar el comportamiento general del circuito. Como resultado, puede fallar al predecir correctamente salidas para combinaciones de entrada no vistas, aunque estas sigan la misma lógica.

2.2.5. Validación de modelos

La validación de modelos es el proceso de confirmar que un modelo produce resultados suficientemente correctos para el propósito con el que fue diseñado, dentro del dominio en el que se aplica [24].

En el dominio de ML, esta validación comúnmente se lleva a cabo separando los datos de entrenamiento en 3 conjuntos:

- Un conjunto de entrenamiento, utilizado para entrenar los modelos
- Un conjunto de validación, que se utiliza para evaluar los modelos entrenados y ver que generalicen bien y no tengan sobreajuste. Se puede utilizar para entrenar múltiples modelos similares y escoger el que tenga mejor rendimiento.
- Un conjunto de prueba para realizar una evaluación final e imparcial del modelo [21].

A pesar de los nombres utilizados para los conjuntos, la validación final de la efectividad del modelo tendrá que venir del conjunto de prueba.

En el caso de aprender el comportamiento de un circuito tenemos una situación única que no se da comúnmente en ML. Cuando el circuito es pequeño, se pueden conocer todos sus posibles pares de entrada/salida. Por lo que en estos casos se puede entrenar y validar el modelo con el conjunto completo de entradas/salidas posibles del circuito.

2.2.6. Modelos y técnicas

En esta sección se explican los modelos más populares en aplicaciones de ALS, así como los métodos utilizados para adaptarlos a este dominio.

Árboles de decisión (DT)

Un árbol de decisión es una representación de una función que mapean un vector de atributos a un solo valor de salida, una "decisión". Utilizando términos que generalmente se utilizan para estructuras de grafo en forma de árbol, un árbol de decisión llega a su decisión realizando una secuencia de pruebas, iniciando en la raíz del árbol y siguiendo las ramas apropiadas hasta llegar a una hoja. Cada nodo interno del árbol corresponde a una prueba del valor de uno de sus atributos de entrada, las ramas del nodo son etiquetadas con los posibles valores del atributo, y los nodos hoja corresponden con los valores que puede retornar la función.

En el caso general, los valores de entrada y salida pueden ser discretos o continuos. Cuando las entradas son valores discretos y las salidas solo pueden tener dos valores, falso o verdadero, se le llama clasificación Booleana [21].

En la Figura 2.2 se puede observar la estructura de un árbol de decisión booleano.

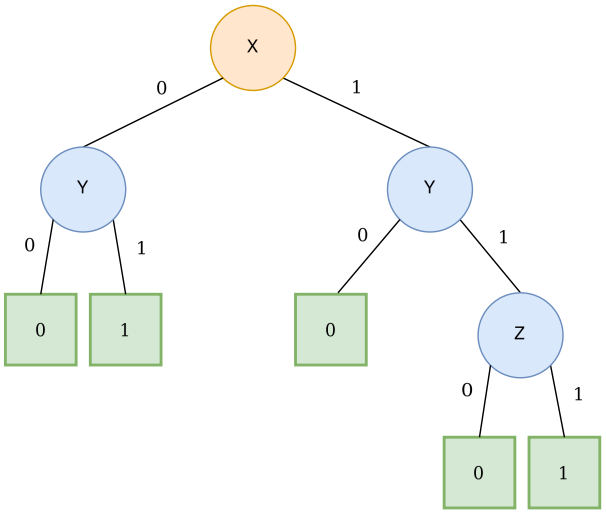


Figura 2.2: Árbol de decisión booleano con tres variables de entrada. El nodo naranja es la raíz del árbol, los nodos azules los nodos intermedios y los nodos verdes son las hojas del árbol, los cuales representan la decisión a la que llega.

En el caso de aprender el comportamiento de un circuito nos enfrentamos a un problema de clasificación booleana: dependiendo de las entradas del circuito, cuál debe ser la salida. Por esto es que, dentro del contexto de ALS, son de particular interés los árboles de decisión booleanos.

Para transformar un modelo DT booleano a una expresión booleana que puede ser expresada con componentes de un circuito se puede emplear el teorema de expansión de Boole [25]. Aplicando este teorema al árbol de la Figura 2.2, este se puede representar con la expresión booleana $(\bar{X}Y) + (XY\bar{Z})$. Un árbol de decisiones no necesariamente se encontrará en la configuración más óptima para expresar la función booleana que implementa, en el caso de la Figura 2.2 la expresión que obtuvimos se puede simplificar más para obtener $Y(\bar{X} + \bar{Z})$. Por este motivo, se recomienda aplicar técnicas de simplificación a la expresión booleana o al circuito derivado de ella.

Un solo DT booleano tiene una sola salida, por lo que para circuitos con múltiples salidas se debe de aplicar un DT que aprenda la función individual de cada salida del circuito.

Bosque aleatorio (RF)

Es común crear modelos que son colecciones de hipótesis y combinar las predicciones de cada hipótesis, puede escogerse un promedio de las predicciones, tratarlas como un

voto de mayoría o aplicar otras técnicas de ML para obtener el resultado final. A las hipótesis individuales se les llama modelo base y su combinación modelo de conjunto. Crear estos modelos de conjunto puede ser beneficioso para disminuir el sobreajuste y sesgo del modelo base. Los bosques aleatorios son de los modelos de conjunto más comúnmente aplicados, son formados por DT como modelo base [21].

En el caso de un RF conformado por DT booleanos, se determina la salida del bosque con un voto de mayoría por parte de los árboles. Para implementar este voto por mayoría en un circuito se pueden utilizar sumadores y un comparador.

Perceptrón multicapa / Red neuronal

Las redes neuronales tienen sus orígenes en [26], donde con inspiración biológica se intentó modelar la red de neuronas en el cerebro con circuitos computacionales. A pesar de sus orígenes, al utilizar redes neuronales en el contexto de ML el realismo biológico impondría restricciones innecesarias que limitarían su practicabilidad. Por esto en la actualidad las redes neuronales no se tratan como un modelo biológico, sino que son de interés como modelos puramente estadísticos para el reconocimiento de patrones.

Existen muchos modelos de redes neuronales, pero uno de particular interés es el perceptrón multicapa. Este uno de los tipos de red neuronal más conocida y utilizada [27].

La arquitectura de perceptrón multicapa se caracteriza por formar un grafo acíclico dirigido de nodos llamados neuronas artificiales. Los nodos se organizan en capas, como se puede observar en la Figura 2.3. Hay capas de nodos designadas de entrada y de salida, cada nodo le aplica un procesamiento a sus entradas y pasa los valores resultantes a sus sucesores en la red. Estos valores eventualmente se propagan hasta los nodos de salida.

Típicamente, el procesamiento de cada nodo es tomar una suma ponderada de sus nodos predecesores y aplicarle una función de activación, la cual en perceptrones multicapa modernos es continua y no lineal. Sea a_j la salida del nodo j y sea $w_{i,j}$ el factor de ponderación para la conexión entre el nodo i y el nodo j; entonces se tiene

$$a_j = g_j\left(\sigma_i w_{i,j} a_i\right),$$

donde g_j es la función de activación no lineal del nodo j. [21]

En un perceptrón clasificador, los nodos de salida corresponden a las posibles clases en las que se puede clasificar, como en el caso de salidas booleanas. Un MLP con salidas booleanas actúa como un clasificador con posibles clases 1 o 0. El valor que se propaga a las salidas típicamente se interpreta como un nivel de confianza de que un nodo de salida representa la respuesta correcta, y se selecciona el mayor de estos valores.

Para profundizar en el tema del funcionamiento y entrenamiento de perceptrones multicapa, se recomienda consultar el Capítulo 5 de [23].

No es fácil traducir MLP booleanos a circuitos. Un método sencillo es sintetizar módulos aritméticos para realizar las computaciones necesarias, como sumadores y multiplicadores. Sin embargo, esto resulta en circuitos muy grandes [10]. En [11], el equipo 3 utiliza un método donde podan conexiones de baja importancia para reducir el tamaño de la red y cada nodo del MLP es transformado en una LUT a través de cuantización de su

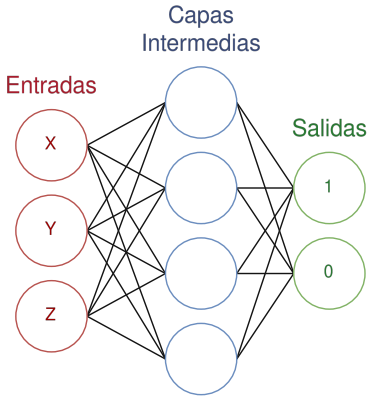


Figura 2.3: Estructura general de un perceptrón multicapa con salidas booleanas. Puede llegar a tener más capas intermedias.

salida, evitando utilizar circuitos aritméticos complejos. Se recomienda leer el apéndice de la referencia para más detalles.

Programación genética cartesiana (CGP)

El término CGP aparece originalmente en [28] como un método de programación genética para aprender funciones booleanas. El cromosoma utilizado por el algoritmo representa una matriz rectangular de funciones primitivas con sus respectivas conexiones y funcionalidades, lo cual constituye una representación de un circuito. Se le denomina "cartesiana" porque organiza los nodos en una rejilla bidimensional identificada mediante coordenadas cartesianas.

Un gran beneficio de CGP es que el proceso evolutivo puede empezar desde un circuito aleatorio o uno predefinido, lo que permite trabajar a partir de circuitos encontrados por otros métodos [15]. Esto lo hace fácil de combinar con técnicas existentes. Otra ventaja es que su cromosoma ya representa directamente un circuito, por lo que convertirlo en uno es un proceso simple.

3 Marco metodológico

Este capítulo detalla la metodología seguida para el desarrollo del proyecto. Busca que el lector comprenda cómo se estructuró el proceso de diseño, implementación y validación de la solución propuesta.

El proyecto contó con las siguientes etapas principales:

1. Experimentación y familiarización con AxLS.
2. Selección del método ML a implementar dentro de AxLS.
3. Implementación de técnica de ML dentro de AxLS.
4. Diseño de experimentos y recolección de resultados.

3.1. Estrategia para la solución

En alineación con los objetivos específicos, este proyecto cumplió con las siguientes tareas clave:

- Escoger una técnica de ML para ALS lo más del estado del arte posible.
- Implementar esta técnica como un método de ALS disponible dentro de la herramienta de fuente abierta AxLS.
- Utilizar AxLS para comparar los resultados obtenidos por la nueva técnica ML con los de otras técnicas ya previamente disponibles dentro de la herramienta.

Cada tarea involucró tanto actividades de diseño como de ejecución, incluyendo revisión bibliográfica e implementación de código.

Adicionalmente, se designó un tiempo en las semanas iniciales para experimentación y familiarización con AxLS, con el único objetivo de que el autor se familiarizara con los flujos de la herramienta y obtuviera una mejor claridad en los pasos que llegaran a ser necesarios en la implementación e integración de una técnica ML. Este fue un periodo puramente exploratorio por lo que no se definió ningún entregable o actividades claras como tal.

La Figura 3.1 contiene un diagrama que muestra las diferentes etapas, sus actividades y productos principales. En el resto de esta sección se entra en más detalles sobre cada una de las tareas clave, las actividades realizadas y productos generados. También, la Tabla 3.1 muestra un resumen de las principales actividades asociadas a cada objetivo específico y los entregables producidos.

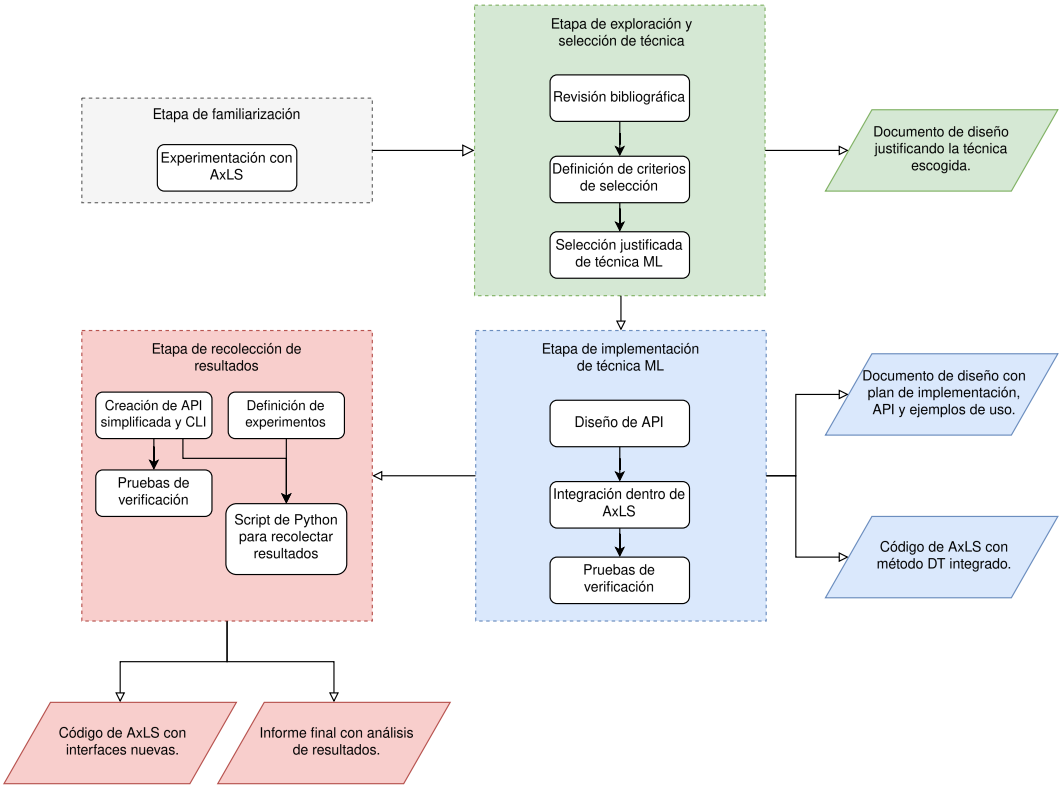


Figura 3.1: Diagrama de la secuencia de pasos seguidos y los productos de cada etapa.

Selección de técnica ML

Para la selección de la técnica ML, primero se realizó un estudio exhaustivo de la literatura disponible sobre aplicación de métodos ML en ALS. Estas referencias son las mencionadas en la sección 1.1.3. Para este estudio bibliográfico se empleó la herramienta Zotero para un manejo más fácil de las referencias utilizadas.

Seguidamente, se definieron los criterios para la selección de la técnica ML más apropiada a implementar en AxLS. Basado en estos criterios de selección, se elaboró un documento de diseño evaluando los métodos identificados en el estudio bibliográfico y concluyendo cuál era el más apropiado.

A continuación se detallan los criterios definidos para la selección del método.

Viabilidad

- No se cuentan con computadores de alto rendimiento para llevar a cabo entrenamientos extensivos.
- AxLS en sí no tiene restricciones que impidan implementar métodos que requieran tiempos de ejecución altos. Sin embargo, como es de fuente abierta, la idea es que

cualquier persona pueda usar la herramienta completa, incluso en un computador personal (tipo laptop común), sin que las ejecuciones tarden más de 1 hora. Que más usuarios puedan probar la mayor cantidad de métodos facilita su adopción y, en consecuencia, puede generar más contribuciones y mayor utilidad.

- Si es posible ejecutar las pruebas rápidamente, eso es altamente preferible para facilitar la verificación del funcionamiento del método y su ejecución por cualquier usuario.
  - Se definió arbitrariamente, como un valor de tiempo bajo, que la ejecución del método seleccionado preferiblemente dure menos de 5 minutos en un computador personal.
- También se consideró la viabilidad de la implementación, ya que este proyecto cuenta con poco tiempo para desarrollarse.

Estado del arte

- Es preferible implementar un método lo más moderno posible para asegurar una mayor relevancia en el estado del arte.
- Es preferible implementar un método con amplia base de estudios previos, como heurística de su relevancia y para contar con más puntos de comparación en la literatura.

Resultados

- Se dará menor importancia a los resultados obtenidos por la técnica en la literatura, ya que este trabajo tiene un rol más fundacional: implementar técnicas de ML en AxLS, y no necesariamente busca obtener los mejores resultados posibles. Sin embargo, en caso de similitud entre métodos en los demás criterios, se podrá considerar cuál ha obtenido mejores resultados en otros estudios.

3.1.1. Implementación de técnica ML

Para la implementación de la técnica ML dentro de AxLS se redactó un documento de diseño detallando la API para utilizar esta técnica, ejemplos de uso y diagramas visuales para ayudar la comprensión del flujo para utilizarla.

El flujo debe de tomar en cuenta los siguientes pasos:

- Obtención de los datos de entrenamiento.
- Entrenamiento del modelo ML.
- Utilización del modelo ML dentro de AxLS para obtener circuitos aproximados.

- Dependiendo de la técnica seleccionada, este paso puede implicar mapear el modelo ML directamente a un circuito. En ese caso, se genera un archivo Verilog que luego puede ser procesado por AxLS. Alternativamente, el modelo puede integrarse directamente en AxLS como parte de un flujo más complejo, por ejemplo, para estimación de error o exploración del espacio de diseño.

Para verificar la correcta operación de la implementación, se deberá diseñar una prueba enfocada en validar dos aspectos: primero, que el modelo ML logra entrenarse y modelar el comportamiento deseado; segundo, que su integración en AxLS permite generar circuitos aproximados funcionales y comparables en métricas de área y error introducido frente a otros métodos existentes. Esta prueba será de escala reducida y con fines de verificación interna, distinta de los experimentos usados para el análisis cuantitativo presentado en este informe. Su diseño dependerá de la técnica ML seleccionada.

Para realizar esta implementación y su validación se utilizará el lenguaje Python, ya que es el lenguaje en el que está escrito la herramienta AxLS.

3.1.2. Diseño de experimentos y recolección de resultados

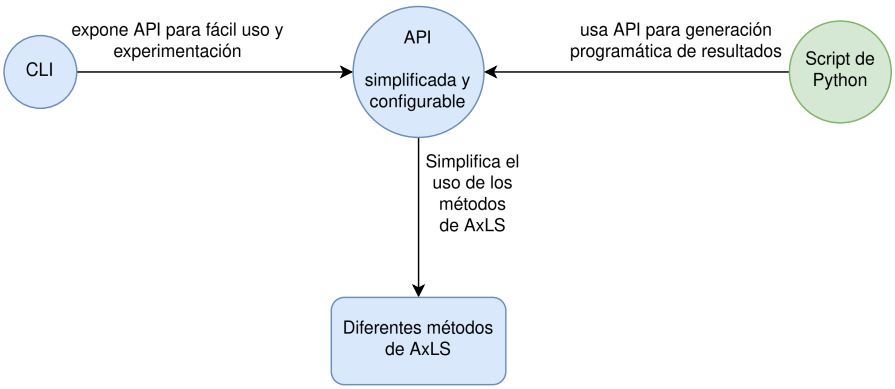


Figura 3.2: Componentes creados para la ejecución de métodos de AxLS. Los bloques azules corresponden a elementos que son parte de AxLS, el script de Python para recolección de resultados sería externo a la herramienta.

Como parte de esta tarea se definieron los resultados necesarios para comparar la técnica de ML implementada con los métodos ya disponibles en AxLS.

Aunque los resultados específicos dependerían de la técnica seleccionada, por ejemplo en caso de requerir variaciones en hiperparámetros, se estableció desde el inicio que las métricas clave serían: error introducido al circuito, reducción de área, y tiempo de ejecución.

AxLS no cuenta con una interfaz de ejecución automatizada, sino que funciona como una biblioteca que debe integrarse manualmente mediante código en Python. Esto representa un reto al momento de ejecutar múltiples pruebas con distintos métodos, parámetros y benchmarks.

Por ello, se decidió diseñar una API interna que permitiera configurar y ejecutar cada técnica de forma simplificada. Esta API fue pensada para facilitar su exposición a través de una interfaz de línea de comandos (CLI), lo que permite una experimentación más ágil sin necesidad de escribir código adicional. Además de facilitar los experimentos, esta interfaz puede fomentar la adopción de AxLS al reducir la curva de aprendizaje para nuevos usuarios.

Con esta infraestructura, fue posible generar todos los resultados de forma automatizada mediante un script sencillo en Python. Además, la CLI resultó fundamental para la ejecución de pruebas de verificación y la exploración ágil de distintos escenarios.

La Figura 3.2 muestra cómo se relacionan los distintos componentes desarrollados.

Para verificar el funcionamiento correcto de la API y la CLI, se implementaron scripts de validación que utilizan directamente la biblioteca de AxLS y se compararon sus resultados con los generados por la nueva interfaz. Dado que algunos métodos de AxLS no son determinísticos, se consideró válido que las ejecuciones tengan diferencias menores, tanto en tiempo como en resultados. Se aceptó un margen de variación del 5 % en métricas de área y error.

Tabla 3.1: Actividades principales del proyecto. La Tabla 1.1 resume las herramientas utilizadas para cada entregable y sus estrategias de verificación.

Entregable	Principales actividades	Objetivo Específico
Documento de diseño que justifica la selección de la técnica ML a implementar en AxLS.	• Revisión bibliográfica. • Definición de criterios de selección. • Selección justificada de técnica ML.	1
Documento de diseño que explica el plan de implementación de la técnica escogida dentro de la herramienta AxLS.	• Diseño de API para técnica ML.	2
Código de herramienta AxLS con técnica de ML integrada.	• Implementación de técnica ML en AxLS. • Verificación de método ML.	2
Código adaptado de herramienta AxLS con adaptaciones para fácil ejecución.	• Implementación de API simplificada. • Exponer API simplificada con una CLI. • Verificación de API simplificada y CLI.	3
Informe final sobre el proyecto realizado y los resultados encontrados al comparar la técnica de ML con las ya existentes en la herramienta AxLS	• Creación de script para generar resultados. • Presentación de datos con elementos gráficos. • Análisis de resultados.	3

4 Descripción del trabajo realizado

Este capítulo consta de dos secciones principales. La primera presenta el proceso seguido en función del marco metodológico: actividades realizadas, retos encontrados, decisiones de diseño tomadas. La segunda sección presenta el análisis de los resultados obtenidos por el proyecto, enfocado en comparar el método de ML integrado en AxLS con otros métodos previamente existentes.

4.1. Descripción del proceso de solución

El proceso para llegar a la solución siguió la estructura planteada en el marco metodológico, que la Figura 3.1 resume visualmente.

4.1.1. Experimentación con AxLS

El objetivo de esta actividad fue familiarizar al autor con la herramienta AxLS, sus módulos existentes y formar una mejor idea de lo que tomaría el proyecto. Se decidió intentar crear una prueba de concepto de lo que sería la verdadera implementación de ML dentro de la herramienta, como ejercicio de familiarización. En el lapso de una semana se intentaría implementar un DT. Se escogió utilizar un DT porque es una técnica fácil de utilizar correctamente, cuya implementación ya existe en la biblioteca *scikit-learn* y que se entrena rápidamente. La prueba de concepto fue exitosa, se logró realizar una síntesis aproximada de uno de los circuitos de benchmark y se identificó cómo realizar los pasos necesarios para la implementación de la técnica:

1. Sintetización del circuito original.
2. Simulación del circuito original para obtener los datos de entrenamiento.
3. Lectura de los datos de entrada y salida del circuito original para entrenar el árbol de decisiones.
4. Entrenamiento del árbol de decisiones con implementación de *scikit-learn*.
5. Mapeo del árbol de decisiones a un circuito Verilog a través de expansión de Boole.
6. Sintetización y simulación del circuito aproximado.
7. Evaluación de métricas de error.

Cabe notar que la técnica no estaba realmente integrada dentro de la herramienta, ya que fue una implementación ad hoc. Además, presentaba varios errores. En el momento que se detectaron, se decidió no dedicar tiempo a intentar corregirlos, porque no había certeza de que esa fuera la técnica final que se escogería para agregar propiamente a AxLS.

4.1.2. Escogencia de método ML

Para escoger el método de ML, se siguieron los criterios planteados en el marco metodológico, en la sección 3.1. Después de una revisión bibliográfica exhaustiva sobre técnicas de ML aplicadas a ALS, se realizó un análisis explorando las diferentes técnicas utilizadas en el estado del arte. Finalmente, se justificó cuál era la más apropiada para este proyecto.

Escogencia de categoría

Como se discute en la sección 1.1.3, en la bibliografía se identificaron 3 categorías principales de aplicación de métodos de ML en ALS:

1. Técnicas que entrenan modelos para asistir en otros métodos de ALS, acelerando la evaluación de errores al simular cambios en el circuito.
2. Técnicas de ML aplicadas a la exploración del espacio de diseño de circuitos.
3. Enfoques supervisados donde se entrena un modelo con entradas/salidas de un circuito, y luego se mapea a hardware.

Para el método a implementar en la herramienta AxLS se escogió la categoría 3 debido a las siguientes consideraciones:

- Se descartó completamente la categoría 2 por el criterio de viabilidad, ya que las exploraciones o entrenamientos realizados duran horas y son realizados en estaciones de trabajo considerablemente más potentes que lo que se tiene acceso para este proyecto.
- Luego se evaluaron más detenidamente los criterios de escogencia entre las categorías 1 y 3:
  - Viabilidad: Por lo general los métodos de la categoría 3 tienen un entrenamiento menos extenso y más fáciles de realizar.
  - Estado del arte: Hay considerablemente más trabajos de la categoría 3.
  - Resultados: En ambas categorías hay resultados competitivos, ninguna es considerablemente más exitosa que la otra.

Escogencia de método

Se evaluaron los métodos disponibles dentro de la categoría 3, y sus consideraciones se discuten a continuación. En la sección 1.1.3 se adentra más en los métodos de las otras categorías.

Árbol de decisiones

- Es el método más estudiado en la literatura, 7 de las 10 referencias utilizadas lo estudian [9], [10], [11], [12], [13], [14], [16].
- Es muy viable debido a su fácil implementación, disponibilidad en bibliotecas de alto grado como *scikit-learn* y bajo tiempo de entrenamiento. En efecto, ya se había realizado una implementación ad hoc como una prueba de concepto a inicios del proyecto.
- Tiene resultados competitivos. No siempre los mejores, pero por lo general la literatura lo menciona como una de las técnicas más efectivas o la base que están intentando superar con un método más avanzado.

Se consideró la posibilidad de implementar alguna de las versiones modificadas que se han estudiado, como en [14] o [12], pero se decidió sería mejor implementar el método más general para tener una base de comparación y dejar cualquier mejora para un trabajo futuro.

Bosque aleatorio

- Fue explorada en 2 de las referencias estudiadas, las cuales son del 2021 [10], [11].
- Son fáciles de implementar, ya que están compuestos por DT.
- Sí ha obtenido mejores resultados que los DT en algunas de las tareas evaluadas dentro de las referencias, particularmente a la hora de generalizar circuitos lógicos y para algunos circuitos aritméticos en específico.

Programación genética cartesiana

- Esta técnica usa algoritmos evolutivos para generar circuitos, representando sus elementos como un *genotipo* fácilmente mapeable al circuito final.
- Ha sido estudiada en 2 de las referencias utilizadas, las cuales son bastante modernas con la última siendo del 2024 [15], [16].
- Esta técnica obtiene resultados muy exitosos en términos del intercambio entre área de circuito y error introducido.
- Se descartó por su alto costo computacional: incluso con un clúster mucho más potente que el equipo disponible, en [15] las ejecuciones tomaron hasta un día completo.

Perceptrón multicapa

- Esta técnica es explorada en 4 de las referencias utilizadas, con 3 de ellas siendo del 2021 y la última del 2024 [8], [10], [11], [16]
- Su viabilidad es limitada por la dificultad de traducirlas a circuitos. Suelen requerir módulos complejos como sumadores y multiplicadores, lo que aumenta mucho el tamaño. Reducir esto implica técnicas como podar conexiones poco importantes o convertir cada nodo del MLP en una LUT mediante cuantización.
- Tienen buenos resultados en sus estudios, particularmente con circuitos aritméticos de adición y multiplicación y aprendiendo la operación lógica de XOR, tareas en las que los otros métodos suelen fallar.

Método escogido Tomando en cuenta las consideraciones dadas para cada método se decidió implementar DT, principalmente por su alta viabilidad y relevancia dentro de la literatura.

Se considera que además es una buena base para trabajo futuro en el que se implemente alguna variación de la técnica como RF, o las presentadas en [14], [12] o [13].

4.2. Implementación de técnica de ML

Debido a la implementación previa de la técnica de DT como prueba de concepto, la implementación real fue más fácil. Aun así, fue necesario integrarla correctamente en AxLS, corregir errores y generalizarla para que funcionara con cualquier circuito, no solo uno específico.

Se creó una clase llamada `DecisionTreeCircuit` que abstrae sobre la implementación de DT de *scikit-learn* para dar las utilidades necesarias para realizar ALS. Esta solo expone los siguientes métodos:

- `train(X, y)`: Acepta vectores con los datos obtenidos de la simulación del circuito original que utiliza para entrenar el árbol de decisiones interno.
- `to_verilog_file(topmodule, filename)`: Después de entrenar el árbol de decisiones, este método mapea el árbol entrenado a un circuito de Verilog y lo escribe en un archivo de texto especificado por el usuario. El parámetro de `topmodule` permite al usuario controlar el nombre del módulo de Verilog para el circuito.

Para que el árbol se adapte al circuito que se está aproximando y el módulo de Verilog se genere correctamente, se deben suplir listas con las entradas y salidas del circuito. También se agregó el parámetro `one_tree_per_output`, que permite elegir entre un solo árbol multi-salida o uno por cada salida del circuito. Esto se incluyó tras notar en la prueba de concepto que ambas opciones eran posibles y afectaban el resultado.

La clase, sus propiedades y métodos se muestran en formato UML en la Figura 4.1.

El flujo completo de utilizar esta clase con las utilidades de AxLS conlleva los siguientes pasos, los cuales se representan visualmente en la Figura 4.2.

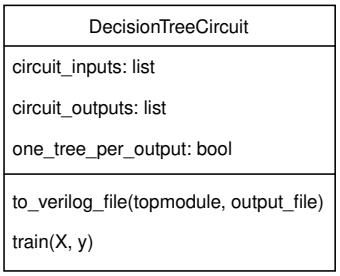


Figura 4.1: Representación de UML de `DecisionTreeCircuit`.

1. Sintetizar circuito original con la clase `Circuit` ya existente dentro de AxLS.
2. Simular el circuito original con un set de datos de entrada, generando las salidas correspondientes.
3. Leer los archivos del set de datos de entrada y sus salidas correspondientes y entrenar el árbol de decisiones con estos.
4. Mapear el árbol de decisiones a un circuito aproximado de Verilog.
5. Sintetizar el circuito aproximado de Verilog con la clase `Circuit`. Con este objeto se pueden realizar simulaciones para obtener las métricas de área y error del circuito aproximado.

Para validar la implementación se hicieron pruebas con un circuito sumador de 9 bits de entrada (dos entradas de 4 bits y 1 bit de acarreo). Las pruebas examinaron los productos intermedios generados y el rendimiento del circuito final para verificar su correcta implementación. Varias de las pruebas se hicieron utilizando un DT sin profundidad máxima, que para este pequeño circuito es capaz de memorizar todos los pares de entradas y salidas, generando resultados exactos.

- Se verificó que las entradas y salidas del circuito de Verilog generado calzaban correctamente a través de inspección manual y revisando que las salidas al simular el circuito coinciden con las salidas al manipular el DT en software directamente.
- Utilizando un DT sin profundidad máxima, se validó en software que las salidas correspondían con lo esperado dependiendo de las entradas. Por ejemplo si las entradas son $3, 4, 1$, la salida sería 8 en binario.
- Utilizando un DT sin profundidad máxima, el circuito aproximado debería tener error de $0\,\%$ y se esperaría que tenga un área mayor al circuito original, ya que memorizar los pares de entradas y salidas es menos eficiente que programar la estructura de un sumador.
- Al reducir la profundidad máxima del DT, el área del circuito generado debía disminuir y el error aumentar.

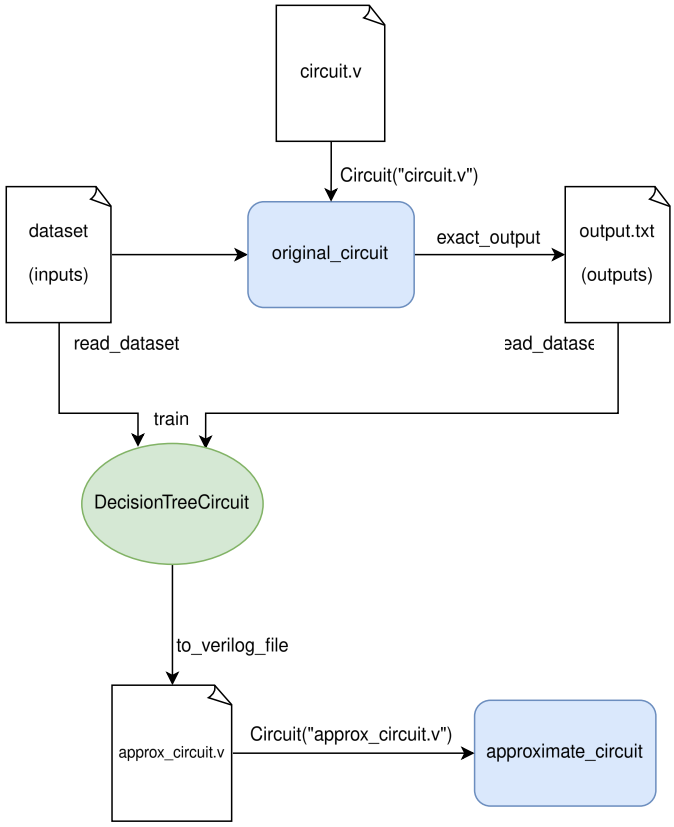


Figura 4.2: Representación visual de cómo utilizar `DecisionTreeCircuit` dentro de AxLS.

4.2.1. Recolección de resultados.

Para la recolección de resultados, como se describió en la sección 3.1.2, se definieron los resultados deseados y los experimentos necesarios, se creó una API para proveer un uso más abstraído de la herramienta AxLS, se creó una CLI que exponía la API para terminal y se escribió software externo a la herramienta para la generación programática de los resultados.

Definición de experimentos

El primer paso que se tomó como parte de la etapa de recolección de resultados fue definir los resultados que se deseaban generar. Este paso era esencial para definir las capacidades que debía exponer la API y los scripts externos que la utilizarían.

En concordancia con el tercer objetivo específico, se buscó recolectar datos de ejecución de la técnica de DT y de otros métodos dentro de la herramienta. Las métricas medidas serían todas para las que la herramienta AxLS tiene soporte:

- Área del circuito.

- Métricas de error del circuito:
  - ER.
  - MHD.
  - WHD.
  - MAE.
  - MSE.

También se decidió medir el tiempo de ejecución de cada método. La herramienta AxLS no proveía soporte para esto directamente, ya que Python ofrece muchas utilidades para medición de tiempo. Pero para la facilidad de los usuarios esta métrica se agregó a la API de ejecución simplificada.

Los resultados recolectados siempre incluirían estas métricas. Se busca obtener los mejores resultados posibles para cada uno de los métodos para realizar una comparación justa. Se compararon los siguientes métodos, considerando las variaciones a probar para sistemática intentar obtener los mejores resultados para cada uno:

- Métodos de poda: Esto incluye el resto de métodos previamente disponibles en AxLS. Para estos métodos se selecciona un umbral de error y se poda el circuito hasta alcanzar el umbral. Se probarían con y sin resíntesis, aunque según [29], la resíntesis casi siempre debería de ser beneficiosa.

- DT: Se decidió variar la máxima profundidad del árbol, si utilizar un solo árbol multi-salida o múltiples árboles con uno por salida y si aplicar resíntesis o no.

La intención era probar todos los métodos de poda disponibles dentro de la herramienta AxLS, pero al experimentar se notó que `ccarving` y `significance` requerirían mucha configuración para obtener resultados óptimos. Esto es debido a que estos métodos se basan en una métrica de "significancia" para escoger cuáles nodos de un circuito podar, la cual se asigna a los nodos del circuito basado en una entrada de las significancias de cada salida. Se decidió que seleccionar manualmente la significancia de las salidas de manera óptima para todos los circuitos que se probarían se sale del alcance de este proyecto. El método `ccarving` también podría utilizar métricas diferentes, pero también se decidió que evaluar diferentes métricas para este método se sale del alcance del proyecto.

Como los métodos de poda tienen un umbral de error controlable, pero el método de DT no, es que se prueban múltiples profundidades máximas para este método. Al permitir mayor profundidad de árbol, se espera que el árbol haga un mejor trabajo de aprender la función, reduciendo su error a pesar de que el circuito final requiera más área.

Creación de API simplificada y CLI

Para la creación de la API simplificada para la ejecución de AxLS, en base con los resultados que se desean generar y en general permitir una ejecución configurable de los métodos de AxLS, se definieron los siguientes requerimientos:

- Poder ejecutar cualquiera de los métodos dentro de AxLS:
  - `ccarving`: Tallado de circuitos, como se presenta en [30].
  - `significance`: Poda a nivel de compuertas basado en significancia, como se presenta en [31].
  - `inconst`: Poda nodos del circuito bajo la suposición de que algunas entradas son constantes. Al ejecutarlo a través de la API se inicia asumiendo que los bits de entrada menos significativos son constantes y al podar todos los nodos sugeridos se asume que el siguiente bit es constante y se continúa hasta alcanzar alguna condición de parada.
  - `outconst`: Igual que `inconst`, pero asumiendo que algunas de las salidas son constantes.
  - `probprun`: Basado en información de temporización adquirida a través de la simulación de circuitos, recomienda podar nodos en función del tiempo que permanecieron en un valor específico, 0 o 1, lo que indica los nodos que asumieron un valor particular la mayor parte del tiempo.
  - `decision_tree`: El método de DT.
- Aceptar todos los parámetros de configuración necesarios de cada método.
  - Parámetros que todos los métodos aceptan: Si utilizar resíntesis, si separar el set de datos en un set de prueba y otro de validación.
  - En común para todos los métodos de poda: Umbral de error, cuántos nodos podar en cada iteración.
  - `ccarving` y `significance`: Una lista de las significancias de cada bit de salida.
  - `decision_tree`: Profundidad máxima del árbol, si utilizar un árbol multisalida o múltiples árboles con uno para cada salida.
- Medir el tiempo de las ejecuciones y ofrecerlo como una de las métricas medibles al ejecutar un método.
- Controlar mostrar el progreso de las simulaciones de los circuitos.
- Capacidad de generar un CSV con los resultados de la ejecución, o agregar los datos a un CSV existente.

Se decidió separar el set de datos en uno de prueba y otro de validación porque, en circuitos grandes, no es posible generar todas las combinaciones posibles de entradas. Esto se debe a que la cantidad de entradas crece de forma exponencial con la fórmula 2^n donde n es la cantidad de bits de entrada. Por eso, se permite dividir el set de datos para usar una parte en validar si el método funciona bien con entradas que no se vieron durante la ejecución.

Esta técnica es común en el dominio de ML, ya que ayuda a saber si un modelo generaliza bien o si está muy ajustado a los datos con los que fue entrenado. En este

caso se usó principalmente para evaluar la capacidad de generalización del método de DT, pero también sirve con los métodos de poda. Ahí ayuda a ver si el umbral de error alcanzado se debe solo al set de prueba o si en realidad representa bien lo que haría el circuito con cualquier entrada. En el caso de `probprun`, que trabaja con datos simulados, el set de validación permite comprobar si el circuito aproximado solo funciona bien con esos datos o si generaliza bien a cualquier entrada.

El requerimiento de poder controlar si se muestra el progreso de las simulaciones se agregó porque AxLS solía siempre mostrar el progreso de las simulaciones a través de la terminal, pero con experimentación se notó que si no mostraba el progreso entonces la simulación se ejecuta más velozmente. Igual se mantuvo la opción de mostrar el progreso si se desea para motivos de depurar errores o simplemente querer saber el progreso de una ejecución.

La opción para agregar los resultados obtenidos a un CSV se agregó por motivos de generación de resultados. Se decidió utilizar el formato CSV porque es ampliamente utilizado, legible por humanos y por máquinas, fácil de entender, fácil de generar y tiene soporte de muchas herramientas.

Todas estas opciones fueron fáciles de exponer a través de una CLI. A la API simplificada no se le agregaron opciones para generación de sets de datos porque AxLS ya previamente proveía una API muy simple para esto, la cual adicionalmente se expuso a través de la CLI para fácil generación de sets de datos para simulación.

Para validar la implementación de la API se probó ejecutar todos los métodos de manera manual con programas de Python que utilizan AxLS como biblioteca y a través de la CLI. Al utilizar el mismo set de datos al aproximar un circuito con el programa y con la CLI, todos generaron circuitos idénticos.

Generación de resultados

Al finalizar la API para ejecutar los métodos de AxLS, se empezó a trabajar en la generación de los resultados deseados.

El primer paso era generar los sets de datos que se utilizarían para cada circuito. Para circuitos pequeños, aquellos con 16 bits de entrada o menos, se pueden generar todas las entradas posibles que pueden tener. Por lo tanto, para los circuitos pequeños se generaron sets de datos exhaustivos, que contienen todas las entradas posibles del circuito. Para lograr esto se tuvo que agregar funcionalidad adicional a AxLS para poder generar sets de datos sin repetición.

Para circuitos grandes se vuelve imposible utilizar sets de datos exhaustivos o representativos. Por ejemplo, en un circuito de 32 bits de entrada, hay $4{,}3 \times 10^9$ posibles entradas al circuito. Aunque se utilice un set de datos de 1 millón de entradas, eso solo representaría un 2,3 % de las entradas posibles del circuito. Para circuitos mayores es mucho peor, debido al crecimiento exponencial de la cantidad de posibles entradas, para un circuito de 64 bits o 128 bits de entrada se vuelve completamente inviable generar un set de datos representativo. Generar un 1 % de sus posibles entradas requeriría más memoria de la que tiene cualquier disco duro moderno.

Otro motivo que complica usar grandes sets de datos es que entre más grande sea el

circuito, más procesamiento requiere simularlo. Un circuito siempre se tiene que simular al final de una ejecución para medir sus métricas de error. Además, el método `probprun` requiere una simulación previa para recolectar datos de temporización. Y posiblemente lo peor es que todos los métodos de poda requieren múltiples simulaciones debido al proceso que siguen:

1. Podar uno o múltiples nodos.
2. Simular el circuito podado para evaluar el error introducido.
3. Si el error introducido es menor al umbral definido, volver al paso 1.
4. Si el error es mayor al umbral definido, se reincorporan los últimos nodos podados hasta que el error vuelva a estar por debajo del umbral.

Este procedimiento también se muestra visualmente en la Figura 4.3.

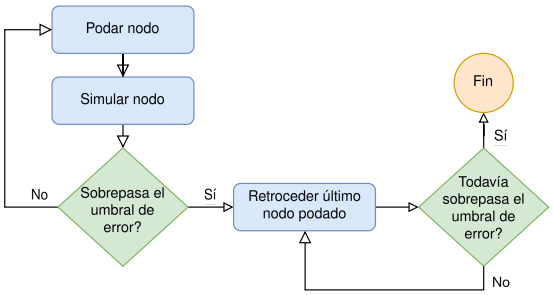


Figura 4.3: Diagrama de flujo del proceso seguido con los métodos de poda.

Tomando en cuenta estas limitaciones, se tuvo que definir de qué tamaño hacer los sets de datos para evitar problemas durante la ejecución. Como la simulación de los circuitos es una de las etapas que más tiempo consume, especialmente en los métodos de poda, se decidió que el tamaño del set de datos dependiera de cuánto tardaba la simulación.

Con eso en mente, se hicieron simulaciones de 1000 elementos para medir cuánto tardaba, en promedio, cada circuito en procesar un solo elemento. Con ese dato se estimó el tamaño adecuado de los sets de datos. Los resultados de esta prueba se muestran en el apéndice, en la Tabla 6.1. Se escogió arbitrariamente un tiempo de simulación de 2 s como un límite corto que permitiera ejecutar los métodos de poda en un tiempo razonable; asumiendo que el proceso podría llegar a requerir alrededor de 1000 iteraciones (estimación tentativa en ese momento), esto daría tiempos de ejecución aproximados de media hora, lo cual se consideró aceptable para el contexto del proyecto.

Para el método `decision_tree` se utilizó un conjunto de datos más grande, ya que requiere menos simulaciones y los métodos de ML se benefician de contar con más datos. Se eligió un set basado en una simulación de 20 s, un tamaño 10 veces mayor que el de los otros conjuntos de datos. No se tenía un análisis formal sobre el impacto en los tiempos de entrenamiento, pero se optó por no aumentarlo más, guiados por la intuición de que

podría alargar el entrenamiento sin aportar beneficios significativos. Sí se sabía que un tiempo de simulación de 20 s implicaba alrededor de un minuto simulando por ejecución.

Basado en esa prueba, se decidió excluir algunos circuitos del resto del proceso de recolección de resultados. La mayoría se descartaron por ser demasiado lentos en simulación; algunos incluso se cancelaron tras 4 horas de ejecución sin progreso aparente. Otros sí completaron la prueba, pero fueron eliminados porque el set de datos que se podía generar para 2 s de simulación era demasiado pequeño (se definió arbitrariamente un mínimo aceptable de 1000 datos) y no podría representar bien el error introducido. También se excluyeron algunos porque no estaban funcionando correctamente, la sintetización del circuito fallaba o no generaba bien las salidas. Finalmente, unos pocos se descartaron aunque sí se podían simular en 2 s, ya que los métodos de poda nunca lograron completarse y fue necesario cancelarlos después de 4 o más horas sin alcanzar el umbral de error. La Tabla 4.1 detalla las razones por las que se excluyó cada circuito.

Tabla 4.1: Circuitos excluidos de la recolección de resultados y sus motivos.

Circuito	Motivo de exclusión
hyp_128b sqrt_128b mul_64b log2_32b	Demasiado lentos en simular, no terminaron la prueba de 1000 datos de entrada.
div_64b square_64b fwrdk2j Mul_32b sin_24b voter RForest	Lentos en simular, el set de datos generado para una simulación de 2 s sería de menos de mil datos.
max_128b adder_128b	Por su complejidad, los métodos de poda duraron demasiado y se cancelaron.
invk2j fir	Tienen errores que no permiten sintetizar o simular correctamente.

Otro cuello de botella identificado es la creación de archivos en formato SAIF [32]. Estos archivos contienen la información de temporización que usa el método `probprun`. Para generarlos, primero se debe simular el circuito deseado. Durante esta simulación, se escribe la información de ejecución en un archivo VCD. Este archivo guarda los cambios de valor de cada nodo del circuito [33], así que su tamaño crece dependiendo del tamaño del circuito y de cuántos datos de entrada se usen en la simulación.

Una vez generado el archivo VCD, este se utiliza como base para crear el archivo SAIF.

Para convertir un archivo VCD a SAIF, AxLS incluye un script en Python llamado `vcd2saif.py`. Se verificó mediante una prueba sencilla que, entre más grande sea el set

de datos, más tiempo toma ejecutar esta conversión. Los resultados de esa prueba se muestran en la Tabla 4.2.

Tabla 4.2: Pruebas de tiempo de ejecución de `vcd2saif.py` en el archivo VCD generado con el circuito BK_16b.

Tamaño de set de datos simulado	Tiempo de ejecución (s)
1000	7,60
4000	29,54

Tomando en cuenta que la simulación de 4000 datos para este circuito dura menos de 200 ms, pero la generación del archivo SAIF toma casi medio minuto, se decidió que valía la pena intentar acelerar este proceso. Al revisar la función en Python, se notó que no hace nada muy complejo, pero sí recorre línea por línea todo el archivo VCD. Estos archivos crecen muy rápido. Por ejemplo, para el circuito BK_16b, con una simulación de 4000 combinaciones de entrada, el archivo VCD llega a tener 724 000 líneas.

Como Python es conocido por ser lento, se decidió reescribir la utilidad `vcd2saif` en un lenguaje más rápido. Los principales candidatos fueron lenguajes compilados como C, Go y Rust. Se eligió Rust porque el autor ya tenía experiencia con él, ofrece buen rendimiento y tiene muchas herramientas útiles para escribir una CLI de forma rápida [34].

Una vez terminada la nueva versión en Rust, llamada `vcd2saif_rs`, se repitió el experimento con el circuito BK_16b y 4000 datos de entrada. La utilidad en Rust tardó solo 1,43 s en una tarea que con la versión en Python tomaba casi medio minuto. Gracias a esto, el tiempo de generación del archivo SAIF se redujo a aproximadamente una vigésima parte del original.

Se cronometraron las ejecuciones de `vcd2saif_rs` para todos los circuitos con archivo SAIF generado; los resultados se incluyen en el apéndice, en la Tabla 6.2. Si se suman esas ejecuciones la generación de archivos SAIF duró hora y media. Asumiendo que el script de Python dura 20 veces más, hubiera durado alrededor de 31 horas. Con estos datos se evidencia que traducir el script a Rust fue sumamente beneficioso para la realización del proyecto.

Finalmente, tras realizar los experimentos definidos previamente, se recolectaron los datos necesarios para evaluar el desempeño de la técnica implementada. A continuación, se presenta un análisis detallado de los resultados obtenidos.

4.3. Análisis de los resultados obtenidos

En esta sección se analizan los resultados de los experimentos descritos previamente. Primero se comparan variantes del método DT: con o sin resíntesis, y usando un solo árbol multi-salida o uno por salida. Luego se contrasta DT contra otros métodos de

AxLS, evaluando métricas de área, error y tiempo. También se discute en qué tipos de circuitos la técnica de DT destaca y en cuáles no funciona.

Variaciones en DT

Resíntesis La primera variación que se evaluó fue si utilizar resíntesis en el circuito aproximado generado por el método `decision_tree` es beneficioso.

La Figura 4.4 muestra una comparación en métricas de error, área y tiempo de ejecución.

El patrón observado en el error es que hay algunas variaciones mínimas, puede ser ligeramente mayor o menor con resíntesis. Como las variaciones son tan pequeñas y no son constantes se atribuyen a un elemento aleatorio en el entrenamiento del DT. Esto es el comportamiento esperado, ya que la resíntesis no debería de afectar las salidas que produce un circuito.

En un entorno práctico no interesarían los casos donde el área del circuito resultante sobrepase el 100 % del área del circuito original, pero igualmente se incluyeron esos casos en la gráfica para mejor estudiar el efecto de resintetizar un circuito. Se observa el mismo patrón que con el error, los valores son muy parecidos en la mayoría de casos con variaciones pequeñas que no son consistentes entre los casos de con y sin resíntesis, por lo que se atribuyen a la aleatoriedad del DT.

En el caso del área sí se esperaba obtener más beneficios de la resíntesis, ya que se esperaba que el DT generara estructuras no tan eficientes y el proceso de resintetizar el circuito pudiera reacomodar los nodos de mejor manera y evitando algunas redundancias. Sin embargo, parece que este no fue el caso, implicando que el resultado del DT no se puede reducir más.

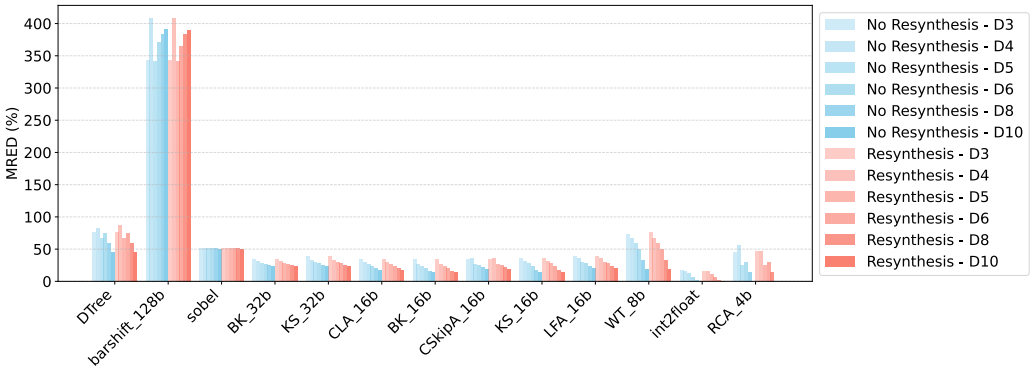
Finalmente, para la comparación de tiempo se tiene un resultado similar, no hay variaciones muy significativas en la mayoría de casos, con la excepción de árboles muy profundos en circuitos muy complejos. Conforme el árbol crece en complejidad, es de esperarse que el tiempo de ejecución aumente considerablemente. Esto se debe a que el tiempo de síntesis crece respecto a la complejidad del circuito. Como se ha mencionado previamente, el DT crece en complejidad exponencialmente con su profundidad. Además, un circuito complejo resulta en tiempos de síntesis muy largos, lo que opaca, en términos de tiempo, el resto de los procesos de la ejecución de ALS.

Sin embargo, los casos donde la resíntesis tiene un efecto considerable en el tiempo de ejecución son los casos donde el área del circuito resultante sobrepasa el 100 % del área del circuito original, por lo que no son casos de interés práctico.

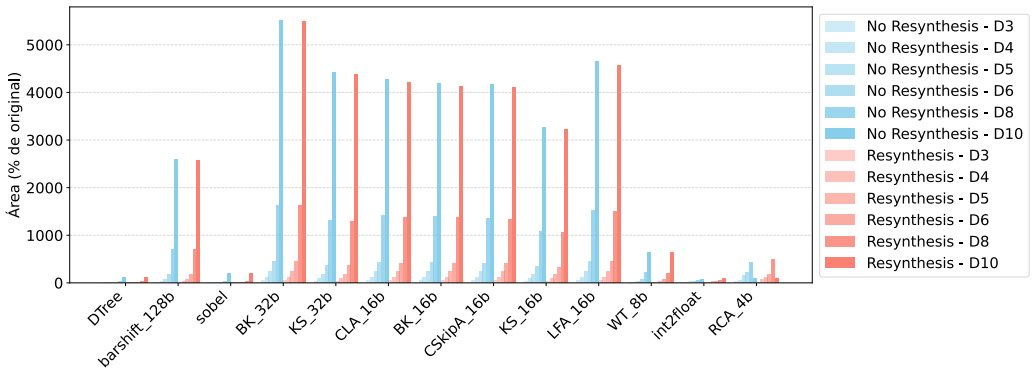
En general, se observó que resintetizar el circuito resultante generado por el método `decision_tree` no tiene efectos significativos, ni positivos ni negativos.

Múltiples árboles La otra variación que se estudió para el método `decision_tree` fue si entrenar un solo árbol multi-salida para representar el circuito completo, o si entrenar un árbol diferente que solo aprendiera una de las salidas del circuito.

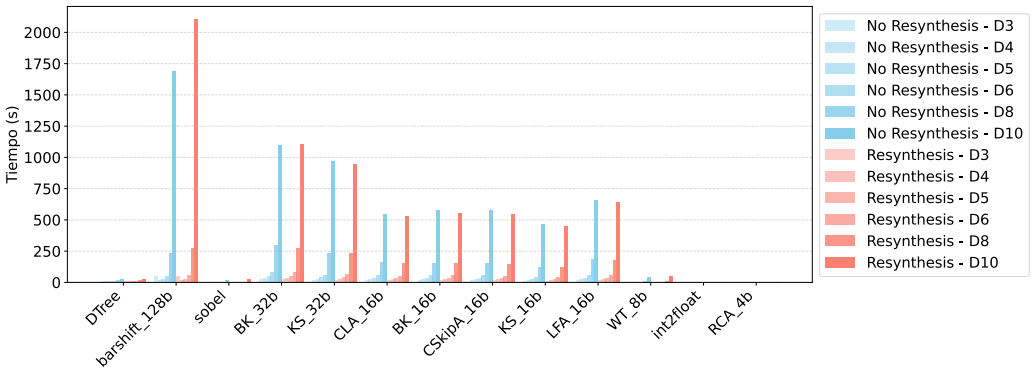
La Figura 4.5 muestra una comparación en métricas de error, área y tiempo de ejecución.



(a) Comparación del error introducido.

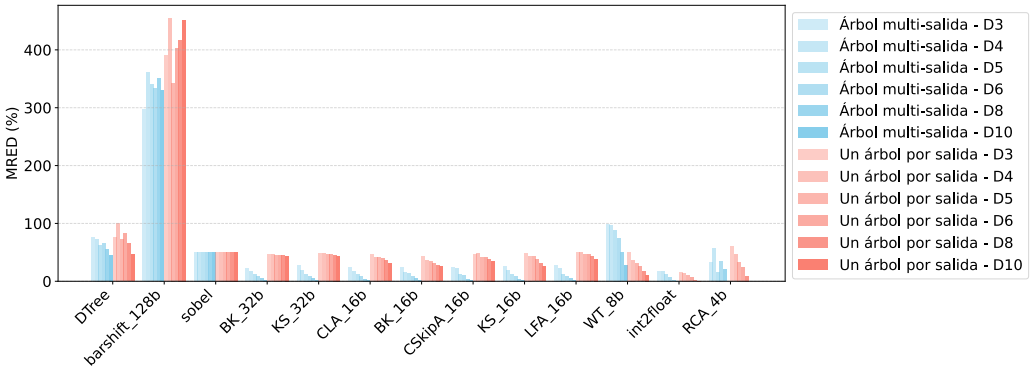


(b) Comparación del área resultante.

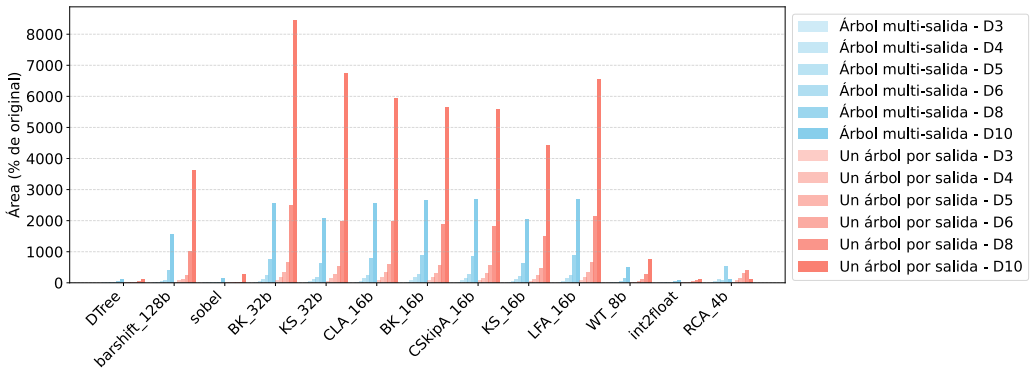


(c) Comparación del tiempo de ejecución.

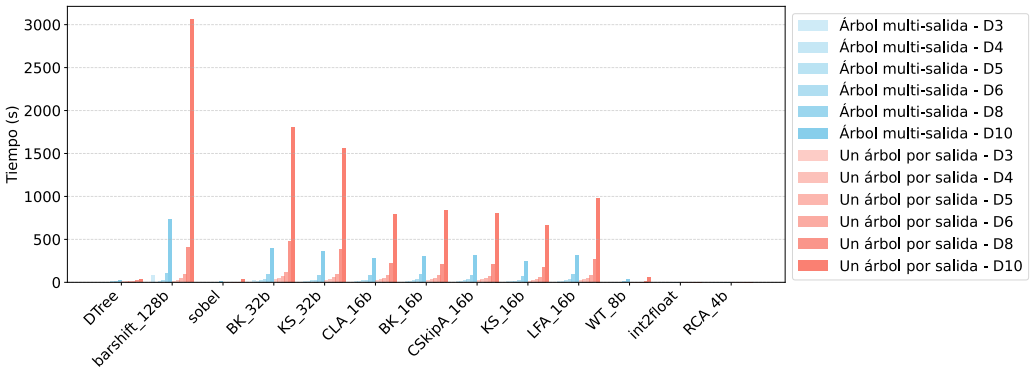
Figura 4.4: Resultados de aplicación de resíntesis con método `decision_tree`.



(a) Comparación del error introducido.



(b) Comparación del área resultante.



(c) Comparación del tiempo de ejecución.

Figura 4.5: Resultados de utilizar un árbol por salida comparado con un solo árbol multi-salida usando el método `decision_tree`.

En la gráfica de error se observa que por lo general produce mejores resultados utilizar un solo árbol multi-salida. Esto es lo que se esperaba, ya que al utilizar un solo árbol este explota mejor las correlaciones entre las salidas del circuito dada una entrada. Si no hubiese correlaciones entre las salidas entonces no habría beneficio, pero los bits de salida de un circuito se esperaría que estén altamente correlacionados entre sí. Cabe notar el caso del circuito `WT_8b`, que es el único que ve una clara mejora al usar árboles separados para cada salida. Este circuito es un multiplicador, lo cual implica que las salidas de la operación de multiplicación no presenta correlación entre sus salidas.

Por otro lado, el área del circuito siempre se ve afectada negativamente, i.e. al utilizar un árbol diferente por salida, el área siempre resulta mayor. Esto incluso con el circuito `WT_8b`, significando que su disminución en error fue un compromiso al costo de mayor área.

Utilizar un árbol por salida resultó en tiempos de ejecución considerablemente más lentos. Esto se atribuye a que se deben entrenar varios DT en vez de solo uno y probablemente producen un circuito más complejo que el que produce un solo árbol resultando en tiempos de síntesis mayores.

DT contra métodos de poda

A continuación se compara el método `decision_tree` agregado a la herramienta contra métodos de poda ya previamente disponibles: `inconst`, `outconst`, `probprun`.

También se discute en qué circuitos `decision_tree` funciona mejor y en cuáles no funciona.

Una característica clave de los métodos de poda en comparación al método de DT es que ellos permiten controlar el error introducido al circuito con un umbral de error definido por el usuario y realizan cambios incrementales en el circuito hasta alcanzarlo. A diferencia del método DT que reemplaza el circuito entero y el área y error del circuito aproximado producido no se puede controlar más allá de variar los parámetros del DT hasta obtener los resultados más cercanos a los deseados.

Por esta limitación del DT, para poder realizar comparaciones lo más equivalentes posible, los métodos de poda se ejecutaron con múltiples umbrales de error diferentes: 1 %, 5 %, 10 %, 25 % y 50 %. Así se pueden escoger las ejecuciones de métodos de poda que tengan un umbral de error similar a los resultados obtenidos por DT.

La Tabla 4.3 muestra los mejores resultados de cada método evaluado con un umbral de error del 1 %. Para esta y próximas tablas, el mejor resultado sería aquel con la menor área entre los resultados con error menor al umbral. En general también solo se incluirán los circuitos en los que el método `decision_tree` logró producir un resultado con error por debajo del umbral y con área menor al 100 % del área original. En el caso de la Tabla 4.3 solo se incluyen los circuitos RCA_4b e int2float porque en ningún otro el método `decision_tree` pudo producir un circuito aproximado con error tan bajo. Esta incapacidad del método de DT de producir resultados con un error controlado limita los circuitos en los que puede ser efectivo. Sin embargo, vemos que los circuitos con los que el método `decision_tree` sí obtiene un error por debajo de 1 % tienen resultados muy positivos en términos de la reducción del área.

Para el circuito RCA_4b todos los métodos encontraron una variación del circuito que tenía 0 % de error, pero tenía área reducida, con el método de DT sobresaliendo en la reducción del área. Cabe tomar en cuenta que este circuito tiene solo 9 bits de entrada, por lo que se utilizó el set de datos exhaustivo de las posibles entradas del circuito para medir el error, lo que significa que realmente el circuito original tenía redundancias.

Con el circuito int2float la diferencia es más marcada, `outconst` y `probprun` reducen el área en 1,46 %, `inconst` en 19,57 % y `decision_tree` en 33,95 %. Esto indica que cuando el método basado en DT sí funciona, obtiene muy buenos resultados en comparación a los métodos de poda. Algo sorprendente en este caso es que los mejores resultados provienen de ejecuciones con un árbol diferente por salida y de mucha profundidad.

Estos dos circuitos son de naturaleza diferente, int2float es un circuito lógico y RCA_4b es un sumador aritmético. Lo que tienen en común es ser de los circuitos más pequeños entre los benchmarks disponibles para AxLS. Solo hay otro circuito de menos de 16 bits de entradas disponible para pruebas: un decodificador de 8 bits con 256 bits de salida. Con el decodificador el DT solo pudo generar un circuito con más área que el original con 0 % de error, o generar un circuito que solo genera valores de 0 en la salida, por lo que no se incluyó en los resultados.

Esto indica que el DT es especialmente efectivo para circuitos pequeños y posiblemente con el decodificador no pudo ser efectivo debido a su gran cantidad de bits de salida. También, se observa que si se tiene la necesidad de mantener un bajo error para estos circuitos pequeños, es posible que permitirle al método de DT formar estructuras más complejas (con mayor profundidad y usando árboles separados por cada salida) le puede permitir encontrar caminos más eficientes para aprender el set de datos limitado.

Tabla 4.3: Comparación de métodos de AxLS para umbral de 1 % de error.

Circuito	`decision_tree`			`inconst`	`outconst`	`probprun`
	Profundidad máxima	Un árbol por salida	Área	Área	Área	Área
RCA_4b	10	Sí	94,78 %	98,51 %	98,51 %	98,51 %
int2float	8	Sí	66,05 %	80,43 %	98,54 %	98,54 %

De aquí saltamos al umbral de 25 %, ya que `decision_tree` no logró generar resultados para nuevos circuitos en los umbrales de 5 % y 10 %. Se presentan los mejores resultados obtenidos en la Tabla 4.4.

Cabe notar que para este umbral, todos los mejores resultados dados por el método `decision_tree` son utilizando un solo árbol multi-salida. Esto se alinea más con lo observado previamente respecto al uso de un árbol multi-salida comparado a múltiples árboles. Para este nuevo umbral el método de DT sigue generando el mejor resultado para int2float, en el resto se mantiene competitivo respecto a `outconst` y `probprun`, pero `inconst` es el ganador por mucho.

El circuito RCA_4b no fue incluido porque la mejor respuesta de `decision_tree` para el umbral de 25 % continúa siendo la misma presentada en la Tabla 4.3.

Tabla 4.4: Comparación de métodos de AxLS para umbral de 25 % de error.

Circuito	`decision_tree`			`inconst`	`outconst`	`probprun`
	Profundidad máxima	Un árbol por salida	Área	Área	Área	Área
int2float	3	No	5,19 %	7,57 %	36,09 %	20,91 %
BK_16b	3	No	26,85 %	10,51 %	49,42 %	84,44 %
KS_16b	4	No	44,15 %	8,00 %	35,41 %	58,07 %
CSkipA_16b	3	No	29,57 %	10,51 %	49,22 %	87,94 %
LFA_16b	4	No	54,86 %	8,75 %	48,44 %	98,83 %
CLA_16b	3	No	29,60 %	8,27 %	43,93 %	73,53 %
BK_32b	3	No	23,22 %	4,28 %	42,06 %	80,88 %
KS_32b	4	No	43,01 %	3,44 %	35,91 %	52,48 %

Todos los circuitos incluidos con los que `decision_tree` logró obtener un error de menos del 25 %, aparte de int2float, son circuitos sumadores que reciben dos entradas de 32 o 16 bits. Los circuitos con los que no logró obtener resultados por debajo de este error son:

- WT_8b: Un multiplicador de 2 entradas de 8 bits.
- sobel: Un circuito que aplica un filtro detector de bordes que recibe una entrada de 81 bits.
- DTree: Un DT entrenado que recibe 30 entradas de 10 bits cada una.
- barshift_128b: Desplazador de barril de 7 bits a la izquierda para entradas de 128 bits.

La imposibilidad de generar un resultado para WT_8b que pudiera incluirse en la Tabla 4.4 sugiere que a un árbol de decisión (DT) le cuesta especialmente aprender la operación de multiplicación, incluso cuando se trata de un circuito relativamente pequeño. Esto indica que a un DT le resulta más sencillo aprender sumas que multiplicaciones, ya que los circuitos sumadores que sí aparecen en la tabla manejan entradas mucho más grandes que el WT_8b. Lo reportado en [10] concuerda con esta observación, donde se muestra que los RF, formados por múltiples DT, también fallan al intentar aprender multiplicaciones. Esta dificultad se atribuye a que los bits de salida en una multiplicación suelen depender de operaciones XOR, las cuales presentan poca información mutua entre entradas y salidas, complicando su aprendizaje para modelos como los DT, que se basan precisamente en la información mutua para dividir los datos.

La exclusión de los circuitos sobel, barshift_128b y DTree parece indicar que el método falla en crear circuitos satisfactorios para circuitos muy grandes. La exclusión del método DTree es especialmente sorprendente porque es una estructura de DT y lo estamos aproximando con un DT.

Se podría analizar más el porqué no se logra crear una aproximación útil. Algunos posibles motivos son que se requiere un DT bastante más complejo que los que se probaron, debido a la gran cantidad de información de entrada. También podría ser que faltó utilizar más datos de entrenamiento o que se usó una estrategia diferente para sintetizar el DT original a un circuito, como reutilizar módulos comparadores en vez de convertir el árbol en un SOP, que es la estrategia utilizada por `decision_tree`.

La exclusión de los circuitos sobel, barshift_128b y DTree sugiere que el método no logra generar aproximaciones satisfactorias para circuitos muy grandes. La exclusión del circuito DTree resulta especialmente sorprendente, ya que originalmente es una estructura de DT adaptada a circuito, y estamos intentando aproximarla precisamente con un DT. Esto podría deberse a varios factores: tal vez se requiere un DT más complejo que los evaluados, debido a la alta cantidad de entradas; podría ser que se necesitaban más datos de entrenamiento; o también podría ser que la síntesis del DT original se hizo con una estrategia distinta, como reutilizar módulos comparadores en lugar de convertir el árbol a una forma SOP, que es la estrategia empleada por `decision_tree`.

El circuito barshift_128b es de naturaleza lógica, lo cual refuerza la teoría de que el fallo en aproximarlo se debe a su tamaño, ya que el circuito int2float, con el que el método DT fue altamente exitoso, es el otro circuito que se probó de naturaleza lógica. Pero de igual forma es inconcluso si se debe solo a su tamaño, puede ser que la función de desplazar bits sea especialmente difícil para un DT de aprender. Habría que realizar pruebas con más desplazadores para estudiar el comportamiento del DT aprendiéndolos.

Finalmente, el circuito Sobel tiene un comportamiento bastante único entre los circuitos probados, ya que es el único que aplica un filtro digital. Sin embargo, comparte con los circuitos que no fueron efectivos el hecho de tener muchos bits de entrada.

Para evaluar si los métodos generalizan bien fuera del set de datos que se usó de prueba, se apartó un 20 % de los datos para hacer una evaluación final de validación. Los valores de error de prueba y validación que obtuvieron las ejecuciones de la Tabla 4.4 se presentan en la Tabla 4.5. Nótese que con circuitos pequeños no se usa un set de validación porque se puede evaluar el set de datos completo de entradas posibles al circuito.

Se observa que todos los métodos generalizan bien. Particularmente importante para `decision_tree`, que entrena basado en los datos de prueba, y el `probprun`, que se basa en información de simulación de los datos de prueba.

Finalmente, se evalúan los resultados de reducción de área para un umbral de 50 % en la Tabla 4.6. Solo incluye circuitos para los que `decision_tree` obtuvo algún resultado entre 25 % y 50 % de error, con menor área que el resultado presentado en la Tabla 4.4.

En la mayoría de circuitos incluidos, el método `decision_tree` entra en segundo lugar en reducción de área, con el método `inconst` ganando. El circuito WT_8b es la excepción, lo cual es interesante porque el DT no había logrado obtener un resultado por encima del umbral de las tablas previas. El método `probprun` obtuvo un resultado similar con el circuito WT_8b, pero los métodos `inconst` y `outconst` obtuvieron resultados mucho peores en ese circuito específico. Todos los circuitos incluidos son sumadores excepto por el WT_8b, lo cual ayuda a explicar esta diferencia en comportamiento.

Se incluye la Tabla 4.7, que muestra los valores de error alcanzados por las ejecuciones

Tabla 4.5: Valores de MRED en los sets de prueba y validación para los resultados obtenidos para el umbral de 25 %.

Circuito	decision_tree		inconst		outconst		probprun	
	Prueba	Validación	Prueba	Validación	Prueba	Validación	Prueba	Validación
int2float	17,90 %	—	24,60 %	—	14,60 %	—	21,90 %	—
BK_16b	24,10 %	24,10 %	20,00 %	21,00 %	8,60 %	8,70 %	21,60 %	21,80 %
KS_16b	18,60 %	18,40 %	19,00 %	19,50 %	16,80 %	17,20 %	24,20 %	23,90 %
CSkipA_16b	23,70 %	24,00 %	20,10 %	20,10 %	17,00 %	16,80 %	22,70 %	23,00 %
LFA_16b	22,60 %	23,10 %	12,70 %	12,80 %	16,80 %	16,60 %	9,50 %	9,50 %
CLA_16b	23,40 %	24,10 %	16,30 %	16,20 %	16,70 %	16,30 %	20,80 %	20,50 %
BK_32b	21,80 %	21,80 %	16,40 %	16,30 %	16,90 %	16,90 %	10,10 %	10,00 %
KS_32b	18,90 %	18,90 %	13,80 %	14,80 %	8,60 %	8,60 %	20,50 %	20,50 %

Tabla 4.6: Comparación de métodos de AxLS para umbral de 50 % de error.

Circuito	decision_tree			inconst	outconst	probprun
	Profundidad máxima	Un árbol por salida	Área	Área	Área	Área
RCA_4b	3	No	26,12 %	6,72 %	47,76 %	70,15 %
WT_8b	3	Sí	14,20 %	63,68 %	91,27 %	52,16 %
KS_16b	3	No	20,74 %	2,22 %	30,81 %	53,63 %
LFA_16b	3	No	27,04 %	2,92 %	41,44 %	82,88 %
KS_32b	3	No	17,72 %	1,15 %	35,91 %	50,73 %

de la Tabla 4.6. Se nota que todos los métodos generalizan bien. También se observa que la mayoría de ejecuciones no se acercaron al umbral de 50 %. En el caso de los métodos de poda esto significa que llegaron a un punto donde podar el siguiente nodo del circuito incurriría muchísimo error, haciéndolos sobrepasar el umbral.

Algo interesante a notar es la profundidad máxima de los resultados presentados en las tablas anteriores. Para el método `decision_tree`, los mejores resultados se obtienen con árboles de una profundidad máxima de 3, a pesar de que las entradas son de 16 o hasta 32 bits. Con esta cantidad limitada de niveles, el árbol solo puede dividir el conjunto de datos basado en unos pocos bits de las entradas.

En la Figura 4.6 se muestra la estructura adoptada por el DT que aproximó el circuito KS_16b con un umbral del 50 %. Como se puede observar, el nodo hoja con el valor más pequeño es 26 115, lo que indica que este sumador no es apto para sumar números pequeños. El valor de MRED, que es del 25,4 % para este circuito, no es representativo del error en cualquier suma que se pueda realizar con él; por ejemplo, este circuito aproximado indicaría que $1 + 1 = 26115$. Esto demuestra algo que se podría considerar una falla en la forma en que se llevaron a cabo los experimentos para los circuitos más grandes, que no podían utilizar un conjunto de datos exhaustivo. Dado que se usaron conjuntos de datos con una distribución aleatoria uniforme, los valores grandes, que se acercan a utilizar

Tabla 4.7: Valores de MRED en los sets de prueba y validación para los resultados obtenidos para el umbral de 50 %.

Circuito	`decision_tree`		`inconst`		`outconst`		`probprun`	
	Prueba	Validación	Prueba	Validación	Prueba	Validación	Prueba	Validación
RCA_4b	32,60 %	—	45,60 %	—	30,20 %	—	37,00 %	—
WT_8b	48,40 %	—	42,90 %	—	22,10 %	—	47,70 %	—
KS_16b	25,40 %	25,20 %	28,90 %	29,50 %	32,80 %	32,90 %	42,00 %	41,50 %
LFA_16b	27,50 %	28,10 %	27,20 %	27,20 %	32,80 %	32,50 %	49,90 %	49,50 %
KS_32b	27,60 %	27,60 %	28,50 %	30,10 %	8,60 %	8,60 %	41,20 %	42,40 %

todos los bits de entrada del circuito, están sobrerrepresentados en comparación con los valores pequeños. Si la distribución de los datos que va a recibir el circuito es uniforme respecto al set de entradas posibles, si se tendría el comportamiento deseado. Pero, es más probable que el usuario quiera poder tener un porcentaje de aproximación que sea más consistente a través del rango entero de valores.

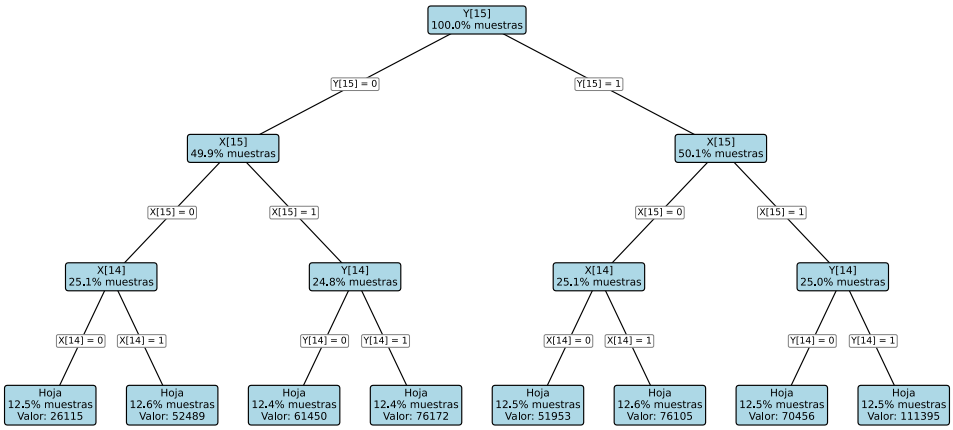


Figura 4.6: Estructura del árbol utilizado para aproximar sumador KS_16b en el resultado de la Tabla 4.6.

Una ventaja del método `decision_tree` es que permite generar un circuito adecuado a cierto rango de datos, dependiendo del conjunto de datos utilizado para el entrenamiento. Con los otros métodos también se puede usar un conjunto de datos representativo de los valores que se desean manejar, lo cual daría una métrica de MRED más alineada con el uso real; sin embargo, tienen mayor probabilidad de mantener partes del circuito que no

son tan útiles para la tarea específica.

Para demostrar una aplicación real del método de DT para la generación de circuitos aproximados, en la Figura 4.7 se presentan ejemplos de su uso en aplicaciones de procesamiento de imágenes. Se muestran los resultados de aplicar un filtro de desenfoque gaussiano y un filtro de detección de bordes Sobel a una imagen, primero con un sumador exacto y luego con dos sumadores aproximados distintos generados con DT. Los porcentajes de área mostrados en la imagen son relativos al circuito BK_16b, que era el sumador de 16 bits de menor área entre los disponibles.

Para estos ejemplos, los DT se entrenaron con un conjunto de datos que contenía todas las parejas de sumas posibles de 8 bits, y luego valores aleatorios seleccionados entre 9 y 12 bits, ya que no era necesario sumar valores mayores. Esto se debe a que los píxeles individuales tienen valores de 8 bits, pero las sumas requeridas alcanzaban un valor máximo de 4080, el cual requiere 12 bits para su representación.

El kernel utilizado para el filtro gaussiano se presenta en la siguiente matriz:

$$K = \begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}.$$

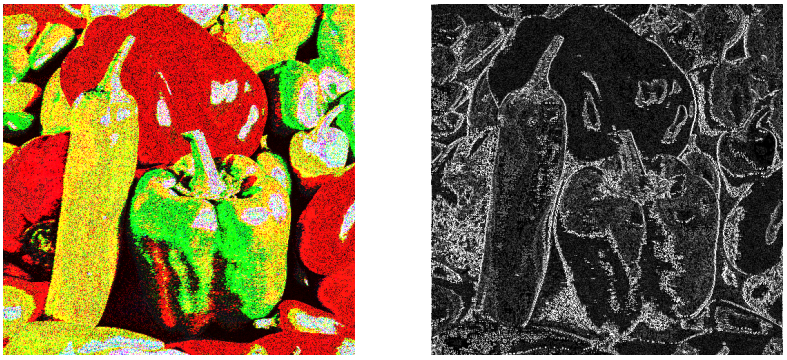
Este kernel fue escogido porque permite aplicar el filtro sin necesidad de un multiplicador, ya que las multiplicaciones pueden realizarse únicamente con corrimientos a la izquierda. Esto lo hace más representativo de una aplicación real donde se utilice un sumador aproximado. Como un multiplicador es considerablemente más complejo que un sumador, usar un sumador aproximado junto con un multiplicador exacto resultaría en ganancias mínimas de área. Para el filtro Sobel también se evitó el uso de multiplicaciones.

Como se puede observar, al utilizar un DT de profundidad 3 con un árbol diferente por salida, se obtiene un resultado visualmente muy distinto al obtenido con un DT de profundidad 4 que usa un solo árbol multi-salida. Esto ocurre a pesar de que ambos circuitos aproximados tienen métricas similares de MRED y MED.

También se incluyó en la figura la métrica SSIM (Índice de similitud estructural) como una medida objetiva de la similitud entre los resultados obtenidos con un sumador aproximado y el resultado exacto. Se aprecia que, aunque el circuito generado por el DT con profundidad máxima 4 tiene un valor de MRED bajo, el resultado del filtro de desenfoque gaussiano presenta un valor de SSIM mucho menor que el obtenido con el DT de profundidad máxima 3, y visualmente la imagen resultante apenas se puede distinguir. Esto constituye otro ejemplo de cómo las métricas pueden ocultar detalles importantes. También muestra que, aunque usar un árbol por salida suele generar peores métricas, puede ser altamente beneficioso dependiendo de la aplicación.

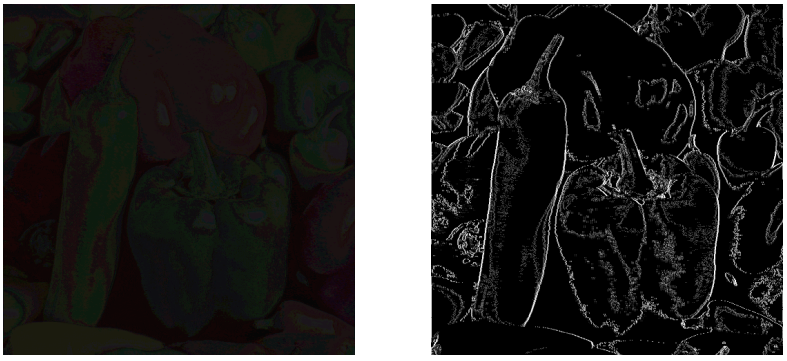
En la Figura 4.8 se comparan tiempos de ejecución para obtener los resultados mostrados en las tablas anteriores. Se observa que el método `decision_tree` es el que tuvo ejecuciones más largas. Cabe notar que para `decision_tree` se utilizó un set de datos ajustado para que la simulación durase 20 s, en vez de los 2 s de simulación a los que se ajustaron los otros métodos. Esto se hizo porque el método `decision_tree` solo realiza 2 simulaciones por ejecución, a diferencia de los otros métodos que realizan una por

(a) Aplicación de filtros con sumador exacto.



SSIM=0.18 SSIM=0.17

(b) Profundidad máxima=3; Un árbol por salida=sí; MRED=27.3 %; MED=149.2; Área=59.73 %



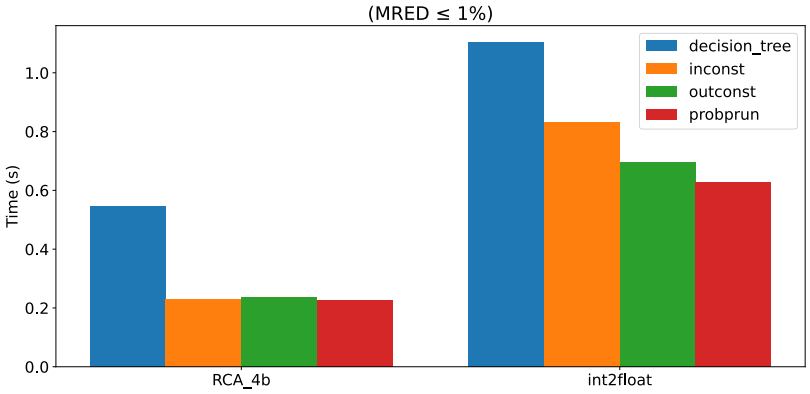
SSIM=0.05 SSIM=0.17

(c) Profundidad máxima=4; Un árbol por salida=no; MRED=24.2 %; MED=161.4; Área=49.61 %

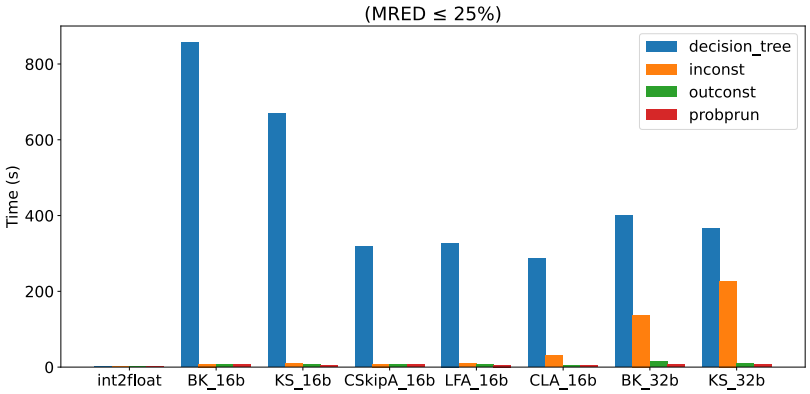
Figura 4.7: Uso de sumadores aproximados con método de DT para la aplicación del filtro de desenfoque gaussiano (izquierda) y el filtro de detección de bordes Sobel (derecha).

iteración y realizan múltiples iteraciones. De igual forma, la diferencia en tiempos para circuitos más grandes supera los 40 s que el método `decision_tree` tomaría en simular, sumado a esto que el método `decision_tree` fue el más lento en circuitos pequeños como RCA_4b e int2float, donde todos los métodos realizan una simulación con el set de entradas exhaustivas que pueden tener los circuitos, se puede concluir que la diferencia en tiempos de ejecución no se debe a los tiempos de simulación.

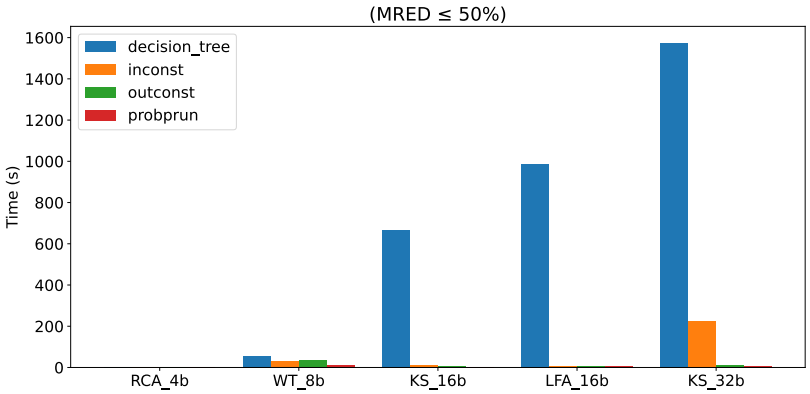
Dado esto, parece indicar que el método `decision_tree` dura más tiempo que los otros debido a los tiempos de sintetización. Durante las ejecuciones se observó que la etapa de sintetización del circuito generado por el DT contribuía de forma importante al tiempo total de ejecución, que es notablemente mayor que en los demás métodos (como se muestra en la Figura 4.8). Se teoriza que esto se debe a que el DT genera circuitos donde a cada bit de salida se le asigna una SOP compleja directamente, lo cual resulta en una descripción del circuito inusual y difícil de optimizar, sin variables intermedias reutilizables.



(a) Tiempos de ejecución para resultados de Tabla 4.3.



(b) Tiempos de ejecución para resultados de Tabla 4.4.



(c) Tiempos de ejecución para resultados de Tabla 4.6.

Figura 4.8: Resultados de aplicación de resíntesis con método `decision_tree`.

5 Conclusiones y recomendaciones

Se lograron los objetivos planteados en el proyecto. Se evaluaron técnicas de ALS basadas en ML del estado del arte, considerando su viabilidad práctica y relevancia en el estado del arte. Basado en esto se escogió la técnica DT por su rapidez de entrenamiento, fácil mapeo a circuitos y presencia destacada en la literatura. Se implementó una clase que facilita el entrenamiento de un DT y su conversión a un circuito en forma de un módulo de Verilog. Esta clase también sirve como un ejemplo que sienta las bases de cómo implementar métodos de aprendizaje supervisado dentro de AxLS a futuro. Además, se creó una API simplificada y una CLI para ejecutar diferentes métodos, lo que permite una experimentación mucho más ágil y una curva de aprendizaje menor para utilizar AxLS de manera básica.

El método basado en DT fue evaluado frente a tres técnicas de poda disponibles en AxLS. Los resultados muestran que es competitivo, y en algunos casos logra los mejores resultados en términos de reducción de área bajo cierto umbral de error.

Se exploraron variantes del método DT. La resíntesis aportó pequeñas mejoras en algunos casos, pero no fue decisiva. Usar un único árbol multi-salida genera circuitos más pequeños y con menor error que utilizar un árbol diferente para cada bit de salida de un circuito, pero no de manera contundente. Se observó que utilizar múltiples árboles puede generar circuitos que a pesar de ser más grandes, o hasta tener peores métricas de error, capturan mejor los comportamientos complejos de un circuito.

Se encontró que en varios circuitos el método DT no logró resultados aceptables bajo las métricas de producir un circuito con área menor al 100 % de la original con error MRED menor al 50 %. Esto se vio con tres circuitos de aplicación muy diferente, al ser los circuitos más grandes y complejos. La limitación más importante identificada en el método de DT es la dificultad para controlar el error introducido en el circuito generado. Sin embargo, una ventaja es que es posible ajustar parcialmente el rendimiento del circuito sobre ciertos rangos de datos, controlando el set de entrenamiento del DT. Se recomienda que futuros estudios indaguen más sobre esta propiedad, explorando la capacidad de manipular los datos de entrenamiento para adaptar el rendimiento del circuito generado a las necesidades específicas de su aplicación final. Esta misma estrategia puede ser relevante para otros métodos de aprendizaje supervisado.

Durante la experimentación, también se identificó que la generación de archivos SAIF para el método `probprun` es un cuello de botella. Se logró reducir el tiempo de creación de los archivos SAIF aproximadamente a un 5 % del tiempo original con una reimplementación en Rust de la utilidad que los crea, aunque aún existe margen para optimización. Si este proceso se vuelve suficientemente rápido, comparable a ejecutar una simulación, podría mejorar la efectividad del método `probprun` al permitirle reevaluar información de temporización durante la poda.

Se recomienda investigar otras técnicas de ML que ofrezcan mejor control sobre el error introducido. Para esto, se podría explorar métodos con múltiples hiperparámetros ajustables, como los MLP. También sería valioso explorar modelos cuya estructura se asemeje más a la de un circuito, como los DG o la CGP. Los experimentos realizados indican que el método DT no se adapta bien a circuitos grandes, posiblemente porque su estructura de árbol no refleja la de estos circuitos.

También se sugiere utilizar sets de datos más representativos y balanceados, o ejemplos prácticos con conjuntos de datos adecuados para cada caso. En general, es preferible utilizar ejemplos prácticos para evaluar los métodos, más allá de métricas abstractas.

Finalmente, sería útil ampliar el set de circuitos de prueba, incluyendo más circuitos de diferentes tipos y de menor tamaño, para facilitar pruebas rápidas y exploración.

Bibliografía

[1] Tecnológico de Costa Rica. «Reseña del TEC,» visitado 4 de abr. de 2025. dirección: https://www.tec.ac.cr/resena-tec.

[2] J. Castro-Godinez, H. Barrantes-Garcia, M. Shafique y J. Henkel, «AxLS: A framework for approximate logic synthesis based on netlist transformations,» *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 68, n.º 8, págs. 2845-2849, ago. de 2021, ISSN: 1549-7747, 1558-3791. DOI: 10.1109/TCSII.2021.3068757. visitado 7 de feb. de 2025. dirección: https://ieeexplore.ieee.org/document/9386244/.

[3] G. Pasandi, S. Nazarian y M. Pedram, «Approximate logic synthesis: A reinforcement learning-based technology mapping approach,» en *20th International Symposium on Quality Electronic Design (ISQED)*, Santa Clara, CA, USA: IEEE, mar. de 2019, págs. 26-32, ISBN: 978-1-7281-0392-1. DOI: 10.1109/ISQED.2019.8697679. visitado 7 de feb. de 2025. dirección: https://ieeexplore.ieee.org/document/8697679/.

[4] G. Pasandi, M. Peterson, M. Herrera, S. Nazarian y M. Pedram, «Deep-PowerX: A deep learning-based framework for low-power approximate logic synthesis,» en *Proceedings of the ACM/IEEE International Symposium on Low Power Electronics and Design*, Boston Massachusetts: ACM, 10 de ago. de 2020, págs. 73-78, ISBN: 978-1-4503-7053-0. DOI: 10.1145/3370748.3406555. visitado 16 de dic. de 2024. dirección: https://dl.acm.org/doi/10.1145/3370748.3406555.

[5] Y. Ye, T. Chen, Y. Gao, H. Yan, B. Yu y L. Shi, «Timing-Driven Technology Mapping Approximation Based on Reinforcement Learning,» *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 43, n.º 9, págs. 2755-2768, sep. de 2024, ISSN: 0278-0070, 1937-4151. DOI: 10.1109/TCAD.2024.3379016. visitado 21 de feb. de 2025. dirección: https://ieeexplore.ieee.org/document/10477245/.

[6] M. A. Rajput, S. Alyami, Q. A. Ahmed, H. Alshahrani, Y. Asiri y A. Shaikh, «Improved Learning-Based Design Space Exploration for Approximate Instance Generation,» *IEEE Access*, vol. 11, págs. 18 291-18 299, 2023, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2023.3247303. visitado 21 de feb. de 2025. dirección: https://ieeexplore.ieee.org/document/10049435/.

[7] M. Awais, H. G. Mohammadi y M. Platzner, «DeepApprox: Rapid Deep Learning based Design Space Exploration of Approximate Circuits via Check-pointing,» en *2024 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, Knoxville, TN, USA: IEEE, 1 de jul. de 2024, págs. 88-93, ISBN: 979-8-3503-5411-9. DOI:

10.1109/ISVLSI61997.2024.00027. visitado 21 de feb. de 2025. dirección: https://ieeexplore.ieee.org/document/10682776/.

[8] S. Boroumand, C.-S. Bouganis y G. A. Constantinides, «Learning boolean circuits from examples for approximate logic synthesis,» en *Proceedings of the 26th Asia and South Pacific Design Automation Conference*, Tokyo Japan: ACM, 18 de ene. de 2021, págs. 524-529, ISBN: 978-1-4503-7999-1. DOI: 10.1145/3394885.3431559. visitado 7 de feb. de 2025. dirección: https://dl.acm.org/doi/10.1145/3394885.3431559.

[9] B. A. De Abreu et al., «Fast Logic Optimization Using Decision Trees,» en *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*, Daegu, Korea: IEEE, mayo de 2021, págs. 1-5, ISBN: 978-1-7281-9201-7. DOI: 10.1109/ISCAS51556.2021.9401664. visitado 14 de mar. de 2025. dirección: https://ieeexplore.ieee.org/document/9401664/.

[10] Y. Miyasaka, X. Zhang, M. Yu, Q. Yi y M. Fujita, «Logic Synthesis for Generalization and Learning Addition,» en *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Grenoble, France: IEEE, 1 de feb. de 2021, págs. 1032-1037, ISBN: 978-3-9819263-5-4. DOI: 10.23919/DATE51398.2021.9474169. visitado 25 de mar. de 2025. dirección: https://ieeexplore.ieee.org/document/9474169/.

[11] S. Rai et al., «Logic synthesis meets machine learning: Trading exactness for generalization,» en *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Grenoble, France: IEEE, 1 de feb. de 2021, págs. 1026-1031, ISBN: 978-3-9819263-5-4. DOI: 10.23919/DATE51398.2021.9473972. visitado 7 de feb. de 2025. dirección: https://ieeexplore.ieee.org/document/9473972/.

[12] W. Zeng, A. Davoodi y R. O. Topaloglu, «Sampling-Based Approximate Logic Synthesis: An Explainable Machine Learning Approach,» en *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, Munich, Germany: IEEE, 1 de nov. de 2021, págs. 1-9, ISBN: 978-1-6654-4507-8. DOI: 10.1109/ICCAD51958.2021.9643484. visitado 20 de feb. de 2025. dirección: https://ieeexplore.ieee.org/document/9643484/.

[13] Y.-S. Huang y J.-H. R. Jiang, «Circuit Learning: From Decision Trees to Decision Graphs,» *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 42, n.º 11, págs. 3985-3996, nov. de 2023, ISSN: 0278-0070, 1937-4151. DOI: 10.1109/TCAD.2023.3258747. visitado 14 de mar. de 2025. dirección: https://ieeexplore.ieee.org/document/10075520/.

[14] H. Hu y S. Cai, *OPTDTALS: Approximate Logic Synthesis via Optimal Decision Trees Approach*, Version Number: 1, 2024. DOI: 10.48550/ARXIV.2408.12304. visitado 14 de mar. de 2025. dirección: https://arxiv.org/abs/2408.12304.

[15] A. A. S. Berndt et al., «CGP-based Logic Flow: Optimizing Accuracy and Size of Approximate Circuits,» *Journal of Integrated Circuits and Systems*, vol. 17, n.º 1, págs. 1-12, 30 de abr. de 2022, ISSN: 1872-0234, 1807-1953. DOI: 10.29292/jics.v17i1.546. visitado 14 de mar. de 2025. dirección: https://jics.org.br/ojs/index.php/JICS/article/view/546.

[16] J. C. Prats Ramos, N. Sachetti, A. Berndt, J. T. Carvalho y C. Meinhardt, «Impact on Delay, Power and Area of Machine Learning-based Approximate Logic Synthesis,» en *2024 37th SBC/SBMicro/IEEE Symposium on Integrated Circuits and Systems Design (SBCCI)*, Joao Pessoa, Brazil: IEEE, 2 de sep. de 2024, págs. 1-5, ISBN: 979-8-3503-9169-5. DOI: 10.1109/SBCCI62366.2024.10703989. visitado 14 de mar. de 2025. dirección: https://ieeexplore.ieee.org/document/10703989/.

[17] J. Han y M. Orshansky, «Approximate computing: An emerging paradigm for energy-efficient design,» en *2013 18TH IEEE EUROPEAN TEST SYMPOSIUM (ETS)*, Avignon, France: IEEE, mayo de 2013, págs. 1-6, ISBN: 978-1-4673-6377-8 978-1-4673-6376-1. DOI: 10.1109/ETS.2013.6569370. visitado 20 de abr. de 2025. dirección: http://ieeexplore.ieee.org/document/6569370/.

[18] G. Ammes, P. F. Butzen, A. I. Reis y R. Ribas, «Two-Level and Multilevel Approximate Logic Synthesis,» *Journal of Integrated Circuits and Systems*, vol. 17, n.º 3, págs. 1-14, 31 de dic. de 2022, ISSN: 1872-0234, 1807-1953. DOI: 10.29292/jics.v17i3.661. visitado 7 de feb. de 2025. dirección: https://jics.org.br/ojs/index.php/JICS/article/view/661.

[19] R. M. Barr, «An investigation of the symmetric properties of logical functions,» Tesis doct., Monterey, California: US Naval Postgraduate School, 1960.

[20] A. A. S. Berndt, M. Fogaça y C. Meinhardt, «Review of Machine Learning in Logic Synthesis,» *Journal of Integrated Circuits and Systems*, vol. 17, n.º 3, págs. 1-12, 31 de dic. de 2022, ISSN: 1872-0234, 1807-1953. DOI: 10.29292/jics.v17i3.649. visitado 21 de feb. de 2025. dirección: https://jics.org.br/ojs/index.php/JICS/article/view/649.

[21] S. J. Russell y P. Norvig, *Artificial intelligence: a modern approach*. pearson, 2016, págs. 651, 653, 655, 657, 666, 696-697, 751-752.

[22] R. S. Sutton y A. Barto, «Reinforcement learning: an introduction,» 2018, Publisher: The MIT Press.

[23] C. M. Bishop, *Pattern recognition and machine learning* (Information science and statistics). New York: Springer, 2006, pág. 2, 738 págs., ISBN: 978-0-387-31073-2.

[24] S. Schlesinger, «Terminology for model credibility,» *SIMULATION*, vol. 32, n.º 3, págs. 103-104, mar. de 1979, Publisher: SAGE Publications, ISSN: 0037-5497, 1741-3133. DOI: 10.1177/003754977903200304. visitado 10 de abr. de 2025. dirección: https://journals.sagepub.com/doi/10.1177/003754977903200304.

[25] G. Boole, *An Investigation of the Laws of Thought on which are Founded the Mathematical Theories of Logic and Probabilities*. Walton y Maberly, 1854, pág. 72, 451 págs.

[26] W. S. McCulloch y W. Pitts, «A logical calculus of the ideas immanent in nervous activity,» *The bulletin of mathematical biophysics*, vol. 5, págs. 115-133, 1943, Publisher: Springer.

[27] M.-C. Popescu, V. E. Balas, L. Perescu-Popescu y N. Mastorakis, «Multilayer perceptron and neural networks,» *WSEAS Transactions on Circuits and Systems*, vol. 8, n.º 7, págs. 579-588, 2009.

[28] J. F. Miller et al., «An empirical study of the efficiency of learning boolean functions using a cartesian genetic programming approach,» en *Proceedings of the genetic and evolutionary computation conference*, vol. 2, 1999, págs. 1135-1142.

[29] R. Morales-Monge, J. Castro-Godínez y G. Paim, «Improving Netlist Transformation-Based Approximate Logic Synthesis Through Resynthesis,» *IEEE Embedded Systems Letters*, vol. 16, n.º 3, págs. 279-282, sep. de 2024, ISSN: 1943-0663, 1943-0671. DOI: 10.1109/LES.2024.3391220. visitado 28 de abr. de 2025. dirección: https://ieeexplore.ieee.org/document/10504781/.

[30] I. Scarabottolo, G. Ansaloni y L. Pozzi, «Circuit carving: A methodology for the design of approximate hardware,» en *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Dresden, Germany: IEEE, mar. de 2018, págs. 545-550, ISBN: 978-3-9819263-0-9. DOI: 10.23919/DATE.2018.8342067. visitado 28 de abr. de 2025. dirección: http://ieeexplore.ieee.org/document/8342067/.

[31] J. Schlachter, V. Camus, K. V. Palem y C. Enz, «Design and Applications of Approximate Circuits by Gate-Level Pruning,» *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, n.º 5, págs. 1694-1702, mayo de 2017, ISSN: 1063-8210, 1557-9999. DOI: 10.1109/TVLSI.2017.2657799. visitado 28 de abr. de 2025. dirección: https://ieeexplore.ieee.org/document/7850945/.

[32] AMD, *Vivado Design Suite User Guide: Power Analysis and Optimization*. AMD, nov. de 2024, UG907, Version 2024.2. visitado 26 de mayo de 2025. dirección: https://docs.amd.com/r/en-US/ug907-vivado-power-analysis-optimization.

[33] *IEEE Standard Verilog Hardware Description Language*, ISBN: 9780738128276. DOI: 10.1109/IEEESTD.2001.93352. visitado 26 de mayo de 2025. dirección: https://ieeexplore.ieee.org/document/954909/.

[34] W. Bugden y A. Alahmar, «Rust: The programming language for safety and performance,» *arXiv preprint arXiv:2206.05503*, 2022.

6 Apéndices y anexos

6.1. Implicaciones ambientales y de desarrollo sostenible

Este apéndice presenta un análisis integral de las implicaciones sociales, económicas, legales, ambientales, de seguridad y salud relacionadas con el desarrollo del presente proyecto. La reflexión considera tanto el impacto actual como el potencial futuro del trabajo realizado, y se relaciona con los Objetivos de Desarrollo Sostenible propuestos por las Naciones Unidas, en el contexto del ejercicio profesional del ingeniero en computadores.

6.1.1. Aspecto social

Este proyecto se construyó sobre AxLS, una herramienta ya de fuente abierta, lo cual facilita su acceso, reutilización y expansión por parte de la comunidad. Su naturaleza abierta permite que cualquier persona interesada en ALS pueda acceder, experimentar y contribuir. Esto democratiza el acceso a herramientas avanzadas, normalmente reservadas para instituciones con mayores recursos. Además, contribuye al ecosistema académico y de investigación, al brindar puntos de referencia adicionales y plataformas reproducibles que fortalecen la colaboración global.

Relación con los ODS Este proyecto se alinea directamente con el Objetivo 4 (Educación de calidad) y el Objetivo 9 (Industria, innovación e infraestructura), al fortalecer la infraestructura digital y apoyar la formación en tecnologías emergentes.

Proyecciones futuras Si el proyecto AxLS continúa evolucionando, podría consolidarse como un referente en ALS, habilitando redes colaborativas de investigación, incentivando la innovación en arquitectura digital y fortaleciendo el aprendizaje autodidacta en ingeniería.

6.1.2. Aspecto económico

A pesar de ser un proyecto académico, su impacto económico potencial es relevante, especialmente por cómo puede facilitar la investigación y adopción de computación aproximada. Al facilitar el acceso y la experimentación con computación aproximada, se acelera el desarrollo de conocimiento y prototipos en esta área. Esto podría reducir los tiempos de adopción industrial de ALS, impactando sectores donde el costo por eficiencia energética o computacional es crítico.

Relación con los ODS Este proyecto se relaciona con el Objetivo 8 (Trabajo decente y crecimiento económico), al fomentar el desarrollo de nuevas tecnologías que potencialmente dinamizan industrias tecnológicas emergentes.

Proyecciones futuras Al mantenerse como software libre, AxLS podría convertirse en una herramienta esencial para investigación y desarrollo de bajo costo, disminuyendo las barreras de entrada para startups, instituciones educativas y laboratorios de países con menos recursos tecnológicos.

6.1.3. Aspecto legal

Actualmente, AxLS no cuenta con una licencia explícita, lo que deja espacio para múltiples escenarios. La ausencia de una licencia clara limita su adopción industrial y académica, ya que su estatus legal es ambiguo. La elección futura de una licencia, ya sea permisiva (como MIT o BSD) o restrictiva (como GPL o licencias no comerciales), influirá directamente en cómo se usa y distribuye el proyecto.

Relación con los ODS Este aspecto se relaciona con el Objetivo 9 (Industria, innovación e infraestructura), al fomentar entornos de desarrollo responsables y sostenibles mediante el uso adecuado de licencias de software.

Proyecciones futuras La elección de licencia influirá directamente en el tipo de adopción y contribuciones futuras. Si se opta por una licencia permisiva, como MIT o BSD, AxLS podría ser integrado sin restricciones incluso en productos comerciales, incentivando contribuciones por parte de empresas. En cambio, una licencia más restrictiva como GPL asegura que las derivaciones también se mantengan abiertas, protegiendo la filosofía del software libre frente a intentos de apropiación propietaria.

6.1.4. Aspecto ambiental

La computación aproximada tiene como uno de sus principales atractivos la posibilidad de reducir el consumo energético en cómputo intensivo. Al permitir cierto margen de error en los cálculos, se logra disminuir el uso de recursos de hardware, como transistores o energía. Por lo tanto, promover esta técnica puede ayudar a disminuir la huella ambiental de sistemas embebidos, centros de datos y otros contextos tecnológicos.

Relación con los ODS Este trabajo impacta el Objetivo 13 (Acción por el clima) y el Objetivo 12 (Producción y consumo responsables), al buscar soluciones computacionales más eficientes energéticamente.

Proyecciones futuras Existe el riesgo de que se manifieste la Paradoja de Jevons, donde al volverse más eficiente una tecnología, su uso se dispara, generando mayor consumo total. Si ALS se masifica, podría incrementar indirectamente el empleo de dispositivos que, aunque eficientes individualmente, consumen más recursos en conjunto.

6.1.5. Aspecto de seguridad

Uno de los retos de la computación aproximada es que su comportamiento puede ser no determinístico o difícil de predecir en escenarios no controlados. Por ello, no es recomendable utilizarla en aplicaciones críticas para la seguridad, como sistemas de control de aeronaves, automóviles o infraestructura sensible. Aunque el proyecto explora su potencial, es necesario advertir que el uso debe estar restringido a contextos donde el error esté acotado y sea aceptable.

Relación con los ODS Este aspecto se conecta con el Objetivo 9 (Industria, innovación e infraestructura) y el Objetivo 11 (Ciudades y comunidades sostenibles), al considerar la seguridad en tecnologías que puedan integrarse en sistemas reales.

Proyecciones futuras Con mejor entendimiento y herramientas de verificación, podría ser posible usar ALS en contextos más amplios, siempre y cuando existan garantías matemáticas y formales sobre sus límites de error.

6.1.6. Aspecto de salud

Similar al aspecto de seguridad, el uso de computación aproximada en áreas críticas como sistemas médicos debe evitarse a menos que su comportamiento esté estrictamente validado. Un error pequeño puede representar un riesgo importante para el bienestar de las personas. Por otro lado, si ALS contribuye a reducir el consumo energético en centros de datos, podría disminuir emisiones contaminantes, beneficiando indirectamente la salud pública.

Relación con los ODS Este punto se vincula con el Objetivo 3 (Salud y bienestar) y el Objetivo 7 (Energía asequible y no contaminante), al contribuir a una infraestructura tecnológica más limpia y menos intensiva en recursos.

Proyecciones futuras Si el desarrollo de ALS avanza con responsabilidad, podría permitir sistemas médicos auxiliares de bajo consumo energético, siempre que su precisión y confiabilidad estén garantizadas.

6.2. Tablas complementarias a resultados

Tabla 6.1: Tiempo en simular un dato de entrada para cada circuito proveído por AxLS.

Circuito	Tiempo por iteración de simulación (ms)
div_64b	34.5005
square_64b	17.5193
fwrdk2j	12.3813
Mul_32b	8.2466
sin_24b	7.9651
voter	7.3150
RForest	3.8796
max_128b	1.0192
Mul_16b	0.9842
barshift_128b	0.7421
adder_128b	0.3548
sobel	0.3350
dec	0.1928
DTree	0.1448
KS_32b	0.0912
WT_8b	0.0884
BK_32b	0.0731
KS_16b	0.0521
CLA_16b	0.0439
CSkipA_16b	0.0413
BK_16b	0.0408
LFA_16b	0.0399
int2float	0.0367
RCA_4b	0.0335
fir	0.0246

Tabla 6.2: Tiempos de ejecución de `vcd2saif_rs` para generar archivos SAIF basados en una simulación de 2 s.

Circuito	Tiempo de ejecución de `vcd2saif_rs` (s)
voter	549.53
RForest	343.29
DTree	80.87
max_128b	273.32
adder_128b	109.61
barshift_128b	203.19
div_64b	1201.76
sobel	88.06
square_64b	1027.07
BK_32b	30.86
fwrdk2j	487.46
KS_32b	36.82
Mul_32b	507.08
CLA_16b	13.65
Mul_16b	221.48
BK_16b	13.29
CSkipA_16b	13.28
KS_16b	17.00
LFA_16b	13.67
sin_24b	137.59
WT_8b	321.15
int2float	1.31
RCA_4b	0.04
dec	0.61

Tabla 6.3: Resultados recolectados para el método `decision_tree`.

Circuito	Resíntesis	Profundidad máxima	Un árbol por salida	MRED	MRED validación	HD	HD validación	WCE	WCE validación	Tiempo de ejecución (s)	Área (% del área original)
DTree	Sí	3,00	Sí	0,767	0,759	0,829	0,830	16,0	16,0	11,2	$8{,}55 \times 10^{-3}$
DTree	Sí	3,00	No	0,767	0,759	0,830	0,831	16,0	16,0	5,68	$8{,}55 \times 10^{-3}$
DTree	No	3,00	Sí	0,767	0,759	0,829	0,830	16,0	16,0	11,1	$8{,}55 \times 10^{-3}$
DTree	No	3,00	No	0,767	0,759	0,830	0,831	16,0	16,0	5,23	$8{,}55 \times 10^{-3}$
DTree	Sí	4,00	Sí	1,01	1,01	0,754	0,756	23,0	23,0	13,2	0,0387
DTree	Sí	4,00	No	0,722	0,731	0,769	0,771	16,0	16,0	6,60	0,0248
DTree	No	4,00	Sí	1,01	1,01	0,754	0,756	23,0	23,0	13,5	0,0375
DTree	No	4,00	No	0,722	0,731	0,769	0,771	16,0	16,0	6,60	0,0301
DTree	Sí	5,00	Sí	0,718	0,726	0,679	0,683	24,0	24,0	15,1	0,0525
DTree	Sí	5,00	No	0,626	0,627	0,699	0,704	16,0	16,0	7,42	0,0505
DTree	No	5,00	Sí	0,718	0,726	0,679	0,683	24,0	24,0	15,5	0,0586
DTree	No	5,00	No	0,626	0,627	0,699	0,704	16,0	16,0	7,09	0,0533
DTree	Sí	6,00	Sí	0,835	0,827	0,613	0,618	27,0	26,0	17,6	0,114
DTree	Sí	6,00	No	0,647	0,659	0,637	0,640	16,0	16,0	8,45	0,109
DTree	No	6,00	Sí	0,835	0,827	0,613	0,618	27,0	26,0	17,2	0,116
DTree	No	6,00	No	0,647	0,659	0,637	0,640	16,0	16,0	7,43	0,111
DTree	Sí	8,00	Sí	0,650	0,651	0,467	0,474	28,0	27,0	21,5	0,389
DTree	Sí	8,00	No	0,548	0,549	0,511	0,517	16,0	16,0	11,1	0,352
DTree	No	8,00	Sí	0,650	0,651	0,467	0,474	28,0	27,0	21,6	0,383
DTree	No	8,00	No	0,548	0,549	0,511	0,517	16,0	16,0	11,5	0,367
DTree	Sí	10,0	Sí	0,451	0,460	0,344	0,360	27,0	27,0	33,8	1,20
DTree	Sí	10,0	No	0,442	0,456	0,394	0,407	16,0	16,0	20,6	1,13
DTree	No	10,0	Sí	0,451	0,459	0,344	0,359	27,0	27,0	33,8	1,21
DTree	No	10,0	No	0,442	0,456	0,394	0,407	16,0	16,0	20,0	1,14
barshift_128b	Sí	3,00	Sí	3,48	3,90	60,8	63,3	$3{,}39 \times 10^{38}$	$3{,}33 \times 10^{38}$	18,4	0,214
barshift_128b	Sí	3,00	No	3,32	2,97	62,5	63,3	$3{,}40 \times 10^{38}$	$3{,}40 \times 10^{38}$	87,3	0,0445
barshift_128b	No	3,00	Sí	3,48	3,90	60,8	63,3	$3{,}39 \times 10^{38}$	$3{,}33 \times 10^{38}$	17,8	0,219
barshift_128b	No	3,00	No	3,32	2,97	62,5	63,3	$3{,}40 \times 10^{38}$	$3{,}40 \times 10^{38}$	90,5	0,0451
barshift_128b	Sí	4,00	Sí	4,10	4,54	59,6	63,2	$3{,}39 \times 10^{38}$	$3{,}36 \times 10^{38}$	29,8	0,652
barshift_128b	Sí	4,00	No	3,70	3,61	62,0	63,1	$3{,}40 \times 10^{38}$	$3{,}40 \times 10^{38}$	7,59	0,166
barshift_128b	No	4,00	Sí	4,10	4,54	59,6	63,2	$3{,}39 \times 10^{38}$	$3{,}36 \times 10^{38}$	27,8	0,637
barshift_128b	No	4,00	No	3,70	3,61	62,0	63,1	$3{,}40 \times 10^{38}$	$3{,}40 \times 10^{38}$	7,06	0,171
barshift_128b	Sí	5,00	Sí	3,20	3,43	58,0	63,1	$3{,}34 \times 10^{38}$	$3{,}37 \times 10^{38}$	44,4	1,24
barshift_128b	Sí	5,00	No	3,31	3,40	61,3	63,0	$3{,}38 \times 10^{38}$	$3{,}40 \times 10^{38}$	12,1	0,388
barshift_128b	No	5,00	Sí	3,20	3,43	58,0	63,1	$3{,}34 \times 10^{38}$	$3{,}37 \times 10^{38}$	41,0	1,24
barshift_128b	No	5,00	No	3,31	3,40	61,3	63,0	$3{,}38 \times 10^{38}$	$3{,}40 \times 10^{38}$	11,4	0,405
barshift_128b	Sí	6,00	Sí	3,47	3,98	55,7	63,1	$3{,}35 \times 10^{38}$	$3{,}31 \times 10^{38}$	96,6	2,57
barshift_128b	Sí	6,00	No	3,23	3,33	60,3	63,0	$3{,}39 \times 10^{38}$	$3{,}35 \times 10^{38}$	23,3	0,907
barshift_128b	No	6,00	Sí	3,46	4,08	55,7	63,1	$3{,}35 \times 10^{38}$	$3{,}31 \times 10^{38}$	85,9	2,57
barshift_128b	No	6,00	No	3,23	3,33	60,3	63,0	$3{,}39 \times 10^{38}$	$3{,}35 \times 10^{38}$	21,4	0,930
barshift_128b	Sí	8,00	Sí	2,60	4,16	48,1	63,3	$3{,}32 \times 10^{38}$	$3{,}38 \times 10^{38}$	439	10,1
barshift_128b	Sí	8,00	No	2,98	3,50	56,9	62,6	$3{,}39 \times 10^{38}$	$3{,}37 \times 10^{38}$	114	3,92
barshift_128b	No	8,00	Sí	2,60	4,18	48,1	63,3	$3{,}32 \times 10^{38}$	$3{,}38 \times 10^{38}$	376	10,2
barshift_128b	No	8,00	No	2,98	3,50	56,9	62,6	$3{,}39 \times 10^{38}$	$3{,}37 \times 10^{38}$	101	3,99
barshift_128b	Sí	10,0	Sí	1,70	4,50	33,5	63,5	$3{,}30 \times 10^{38}$	$3{,}38 \times 10^{38}$	$3{,}40 \times 10^{3}$	35,9
barshift_128b	Sí	10,0	No	2,92	3,31	50,4	62,1	$3{,}35 \times 10^{38}$	$3{,}36 \times 10^{38}$	804	15,7
barshift_128b	No	10,0	Sí	1,73	4,53	33,5	63,6	$3{,}30 \times 10^{38}$	$3{,}38 \times 10^{38}$	$2{,}72 \times 10^{3}$	36,2
barshift_128b	No	10,0	No	2,92	3,31	50,4	62,1	$3{,}35 \times 10^{38}$	$3{,}36 \times 10^{38}$	665	15,8
sobel	Sí	3,00	Sí	0,510	0,510	1,96	1,98	253	252	2,38	$7{,}27 \times 10^{-3}$
sobel	Sí	3,00	No	0,510	0,510	1,96	1,99	253	253	1,58	$5{,}87 \times 10^{-3}$
sobel	No	3,00	Sí	0,510	0,510	1,96	1,98	253	252	2,18	$8{,}38 \times 10^{-3}$

Circuito	Resíntesis	Profundidad máxima	Un árbol por salida	MRED	MRED validación	HD	HD validación	WCE	WCE validación	Tiempo de ejecución (s)	Área (% del área original)
sobel	No	3,00	No	0,510	0,510	1,96	1,99	253	253	1,35	$5{,}87 \times 10^{-3}$
sobel	Sí	4,00	Sí	0,510	0,510	1,96	1,98	253	252	2,81	0,0165
sobel	Sí	4,00	No	0,510	0,510	1,96	1,98	253	253	1,71	0,0179
sobel	No	4,00	Sí	0,510	0,510	1,96	1,98	253	252	2,39	0,0165
sobel	No	4,00	No	0,510	0,510	1,96	1,98	253	253	1,50	0,0176
sobel	Sí	5,00	Sí	0,510	0,510	1,95	1,98	253	252	3,20	0,0369
sobel	Sí	5,00	No	0,510	0,510	1,95	1,99	253	253	2,00	0,0419
sobel	No	5,00	Sí	0,510	0,510	1,95	1,98	253	252	2,97	0,0369
sobel	No	5,00	No	0,510	0,510	1,95	1,99	253	253	1,74	0,0419
sobel	Sí	6,00	Sí	0,510	0,510	1,94	1,98	253	252	3,72	0,0690
sobel	Sí	6,00	No	0,510	0,510	1,94	1,97	253	253	2,32	0,0752
sobel	No	6,00	Sí	0,510	0,510	1,94	1,98	253	252	3,46	0,0724
sobel	No	6,00	No	0,510	0,510	1,94	1,97	253	253	2,02	0,0738
sobel	Sí	8,00	Sí	0,509	0,510	1,90	1,98	253	252	6,36	0,293
sobel	Sí	8,00	No	0,509	0,510	1,90	1,98	253	252	4,41	0,290
sobel	No	8,00	Sí	0,509	0,510	1,90	1,98	253	252	5,93	0,300
sobel	No	8,00	No	0,509	0,510	1,90	1,98	253	252	3,96	0,296
sobel	Sí	10,0	Sí	0,481	0,505	1,79	2,19	253	252	33,0	2,70
sobel	Sí	10,0	No	0,505	0,509	1,83	2,02	253	253	15,3	1,39
sobel	No	10,0	Sí	0,481	0,505	1,79	2,19	253	252	30,1	2,70
sobel	No	10,0	No	0,505	0,509	1,83	2,02	253	253	13,8	1,40
BK_32b	Sí	3,00	Sí	0,477	0,476	15,9	16,1	$5{,}27 \times 10^{9}$	$5{,}31 \times 10^{9}$	32,9	0,662
BK_32b	Sí	3,00	No	0,218	0,218	15,8	15,9	$2{,}72 \times 10^{9}$	$2{,}72 \times 10^{9}$	23,0	0,232
BK_32b	No	3,00	Sí	0,477	0,476	15,9	16,1	$5{,}27 \times 10^{9}$	$5{,}31 \times 10^{9}$	32,5	0,665
BK_32b	No	3,00	No	0,218	0,218	15,8	15,9	$2{,}72 \times 10^{9}$	$2{,}72 \times 10^{9}$	21,9	0,242
BK_32b	Sí	4,00	Sí	0,471	0,466	15,9	16,1	$5{,}24 \times 10^{9}$	$5{,}30 \times 10^{9}$	50,9	1,71
BK_32b	Sí	4,00	No	0,169	0,169	15,8	15,9	$2{,}13 \times 10^{9}$	$2{,}12 \times 10^{9}$	14,7	0,500
BK_32b	No	4,00	Sí	0,471	0,466	15,9	16,1	$5{,}24 \times 10^{9}$	$5{,}30 \times 10^{9}$	51,1	1,74
BK_32b	No	4,00	No	0,169	0,169	15,8	15,9	$2{,}13 \times 10^{9}$	$2{,}12 \times 10^{9}$	14,5	0,501
BK_32b	Sí	5,00	Sí	0,451	0,453	15,6	15,9	$4{,}80 \times 10^{9}$	$4{,}72 \times 10^{9}$	74,7	3,43
BK_32b	Sí	5,00	No	0,125	0,125	15,3	15,4	$1{,}60 \times 10^{9}$	$1{,}57 \times 10^{9}$	24,0	1,26
BK_32b	No	5,00	Sí	0,451	0,453	15,6	15,9	$4{,}80 \times 10^{9}$	$4{,}72 \times 10^{9}$	74,4	3,37
BK_32b	No	5,00	No	0,125	0,125	15,3	15,4	$1{,}60 \times 10^{9}$	$1{,}57 \times 10^{9}$	26,0	1,31
BK_32b	Sí	6,00	Sí	0,451	0,456	15,1	15,6	$4{,}80 \times 10^{9}$	$4{,}75 \times 10^{9}$	122	6,73
BK_32b	Sí	6,00	No	0,0820	0,0820	15,2	15,4	$1{,}02 \times 10^{9}$	$1{,}01 \times 10^{9}$	36,5	2,26
BK_32b	No	6,00	Sí	0,451	0,456	15,1	15,6	$4{,}80 \times 10^{9}$	$4{,}75 \times 10^{9}$	120	6,72
BK_32b	No	6,00	No	0,0820	0,0820	15,2	15,4	$1{,}02 \times 10^{9}$	$1{,}01 \times 10^{9}$	37,6	2,38
BK_32b	Sí	8,00	Sí	0,430	0,445	13,8	14,8	$4{,}49 \times 10^{9}$	$4{,}51 \times 10^{9}$	465	25,0
BK_32b	Sí	8,00	No	0,0470	0,0480	14,6	15,0	$2{,}21 \times 10^{9}$	$2{,}22 \times 10^{9}$	90,4	7,56
BK_32b	No	8,00	Sí	0,430	0,445	13,8	14,8	$4{,}49 \times 10^{9}$	$4{,}51 \times 10^{9}$	503	25,0
BK_32b	No	8,00	No	0,0470	0,0480	14,6	15,0	$2{,}21 \times 10^{9}$	$2{,}22 \times 10^{9}$	98,6	7,80
BK_32b	Sí	10,0	Sí	0,394	0,435	12,1	14,0	$4{,}40 \times 10^{9}$	$4{,}40 \times 10^{9}$	$1{,}83 \times 10^{3}$	84,6
BK_32b	Sí	10,0	No	0,0260	0,0260	13,7	14,5	$4{,}40 \times 10^{9}$	$4{,}39 \times 10^{9}$	391	25,4
BK_32b	No	10,0	Sí	0,394	0,435	12,1	14,0	$4{,}40 \times 10^{9}$	$4{,}40 \times 10^{9}$	$1{,}79 \times 10^{3}$	84,6
BK_32b	No	10,0	No	0,0260	0,0260	13,7	14,5	$4{,}40 \times 10^{9}$	$4{,}39 \times 10^{9}$	401	25,8
KS_32b	Sí	3,00	Sí	0,491	0,493	15,8	16,0	$5{,}29 \times 10^{9}$	$5{,}28 \times 10^{9}$	26,4	0,540
KS_32b	Sí	3,00	No	0,276	0,276	15,8	15,9	$3{,}07 \times 10^{9}$	$3{,}07 \times 10^{9}$	16,4	0,177
KS_32b	No	3,00	Sí	0,491	0,493	15,8	16,0	$5{,}29 \times 10^{9}$	$5{,}28 \times 10^{9}$	26,3	0,552
KS_32b	No	3,00	No	0,276	0,276	15,8	15,9	$3{,}07 \times 10^{9}$	$3{,}07 \times 10^{9}$	13,5	0,182
KS_32b	Sí	4,00	Sí	0,479	0,478	15,6	15,9	$5{,}29 \times 10^{9}$	$5{,}25 \times 10^{9}$	39,9	1,33
KS_32b	Sí	4,00	No	0,189	0,189	15,8	15,8	$2{,}00 \times 10^{9}$	$2{,}00 \times 10^{9}$	11,6	0,430
KS_32b	No	4,00	Sí	0,479	0,478	15,6	15,9	$5{,}29 \times 10^{9}$	$5{,}25 \times 10^{9}$	39,0	1,33
KS_32b	No	4,00	No	0,189	0,189	15,8	15,8	$2{,}00 \times 10^{9}$	$2{,}00 \times 10^{9}$	11,6	0,451
KS_32b	Sí	5,00	Sí	0,469	0,474	15,2	15,6	$4{,}78 \times 10^{9}$	$4{,}80 \times 10^{9}$	59,4	2,69
KS_32b	Sí	5,00	No	0,130	0,130	15,3	15,4	$1{,}58 \times 10^{9}$	$1{,}53 \times 10^{9}$	19,9	1,02
KS_32b	No	5,00	Sí	0,469	0,474	15,2	15,6	$4{,}78 \times 10^{9}$	$4{,}80 \times 10^{9}$	58,1	2,71
KS_32b	No	5,00	No	0,130	0,130	15,3	15,4	$1{,}58 \times 10^{9}$	$1{,}53 \times 10^{9}$	19,5	1,04
KS_32b	Sí	6,00	Sí	0,460	0,466	14,7	15,3	$4{,}79 \times 10^{9}$	$4{,}74 \times 10^{9}$	96,0	5,30
KS_32b	Sí	6,00	No	0,0930	0,0940	15,2	15,4	$1{,}06 \times 10^{9}$	$1{,}04 \times 10^{9}$	29,9	1,86
KS_32b	No	6,00	Sí	0,460	0,466	14,7	15,3	$4{,}79 \times 10^{9}$	$4{,}74 \times 10^{9}$	93,9	5,27
KS_32b	No	6,00	No	0,0930	0,0940	15,2	15,4	$1{,}06 \times 10^{9}$	$1{,}04 \times 10^{9}$	30,1	1,90
KS_32b	Sí	8,00	Sí	0,429	0,456	13,6	14,7	$4{,}52 \times 10^{9}$	$4{,}50 \times 10^{9}$	392	19,8
KS_32b	Sí	8,00	No	0,0490	0,0500	14,5	15,0	$2{,}28 \times 10^{9}$	$2{,}27 \times 10^{9}$	75,5	6,06
KS_32b	No	8,00	Sí	0,429	0,456	13,6	14,7	$4{,}52 \times 10^{9}$	$4{,}50 \times 10^{9}$	380	19,8
KS_32b	No	8,00	No	0,0490	0,0500	14,5	15,0	$2{,}28 \times 10^{9}$	$2{,}27 \times 10^{9}$	85,7	6,39
KS_32b	Sí	10,0	Sí	0,393	0,440	11,9	14,0	$4{,}37 \times 10^{9}$	$4{,}37 \times 10^{9}$	$1{,}54 \times 10^{3}$	67,0
KS_32b	Sí	10,0	No	0,0240	0,0250	13,6	14,5	$4{,}34 \times 10^{9}$	$4{,}33 \times 10^{9}$	353	20,5
KS_32b	No	10,0	Sí	0,393	0,440	11,9	14,0	$4{,}37 \times 10^{9}$	$4{,}37 \times 10^{9}$	$1{,}58 \times 10^{3}$	67,7
KS_32b	No	10,0	No	0,0240	0,0250	13,6	14,5	$4{,}34 \times 10^{9}$	$4{,}33 \times 10^{9}$	366	20,9
CLA_16b	Sí	3,00	Sí	0,454	0,459	7,94	8,01	$8{,}00 \times 10^{4}$	$8{,}00 \times 10^{4}$	21,4	0,665
CLA_16b	Sí	3,00	No	0,234	0,241	7,85	7,89	$3{,}57 \times 10^{4}$	$3{,}57 \times 10^{4}$	10,8	0,300
CLA_16b	No	3,00	Sí	0,454	0,459	7,94	8,01	$8{,}00 \times 10^{4}$	$8{,}00 \times 10^{4}$	20,4	0,660
CLA_16b	No	3,00	No	0,234	0,241	7,85	7,89	$3{,}57 \times 10^{4}$	$3{,}57 \times 10^{4}$	11,0	0,296

Circuito	Resíntesis	Profundidad máxima	Un árbol por salida	MRED	MRED validación	HD	HD validación	WCE	WCE validación	Tiempo de ejecución (s)	Área (% del área original)
CLA_16b	Sí	4,00	Sí	0,421	0,420	7,72	7,81	$8{,}01 \times 10^4$	$7{,}94 \times 10^4$	31,8	1,63
CLA_16b	Sí	4,00	No	0,174	0,180	7,84	7,88	$2{,}97 \times 10^4$	$2{,}97 \times 10^4$	13,7	0,631
CLA_16b	No	4,00	Sí	0,421	0,420	7,72	7,81	$8{,}01 \times 10^4$	$7{,}94 \times 10^4$	31,9	1,64
CLA_16b	No	4,00	No	0,174	0,180	7,84	7,88	$2{,}97 \times 10^4$	$2{,}97 \times 10^4$	13,6	0,623
CLA_16b	Sí	5,00	Sí	0,408	0,416	7,32	7,43	$7{,}29 \times 10^4$	$7{,}31 \times 10^4$	50,1	3,33
CLA_16b	Sí	5,00	No	0,118	0,120	7,38	7,44	$2{,}43 \times 10^4$	$2{,}42 \times 10^4$	19,9	1,42
CLA_16b	No	5,00	Sí	0,408	0,416	7,32	7,43	$7{,}29 \times 10^4$	$7{,}31 \times 10^4$	50,2	3,28
CLA_16b	No	5,00	No	0,118	0,120	7,38	7,44	$2{,}43 \times 10^4$	$2{,}42 \times 10^4$	20,9	1,50
CLA_16b	Sí	6,00	Sí	0,387	0,391	6,84	6,99	$7{,}28 \times 10^4$	$7{,}25 \times 10^4$	78,8	5,90
CLA_16b	Sí	6,00	No	0,0810	0,0840	7,36	7,44	$1{,}61 \times 10^4$	$1{,}59 \times 10^4$	29,6	2,29
CLA_16b	No	6,00	Sí	0,387	0,391	6,84	6,99	$7{,}28 \times 10^4$	$7{,}25 \times 10^4$	78,1	6,07
CLA_16b	No	6,00	No	0,0810	0,0840	7,36	7,44	$1{,}61 \times 10^4$	$1{,}59 \times 10^4$	31,0	2,42
CLA_16b	Sí	8,00	Sí	0,344	0,356	5,70	5,94	$6{,}87 \times 10^4$	$6{,}81 \times 10^4$	226	19,8
CLA_16b	Sí	8,00	No	0,0420	0,0430	6,82	6,97	$9{,}59 \times 10^3$	$9{,}52 \times 10^3$	82,6	7,84
CLA_16b	No	8,00	Sí	0,344	0,356	5,70	5,94	$6{,}87 \times 10^4$	$6{,}81 \times 10^4$	231	20,1
CLA_16b	No	8,00	No	0,0420	0,0430	6,82	6,97	$9{,}59 \times 10^3$	$9{,}52 \times 10^3$	86,6	8,21
CLA_16b	Sí	10,0	Sí	0,290	0,309	4,51	4,91	$6{,}71 \times 10^4$	$6{,}66 \times 10^4$	787	59,0
CLA_16b	Sí	10,0	No	0,0220	0,0230	6,21	6,49	$1{,}82 \times 10^4$	$1{,}82 \times 10^4$	271	25,2
CLA_16b	No	10,0	Sí	0,290	0,309	4,51	4,91	$6{,}71 \times 10^4$	$6{,}66 \times 10^4$	799	60,0
CLA_16b	No	10,0	No	0,0220	0,0230	6,21	6,49	$1{,}82 \times 10^4$	$1{,}82 \times 10^4$	286	25,7
BK_16b	Sí	3,00	Sí	0,433	0,437	7,75	7,80	$8{,}12 \times 10^4$	$8{,}07 \times 10^4$	22,7	0,702
BK_16b	Sí	3,00	No	0,241	0,241	7,85	7,87	$4{,}76 \times 10^4$	$4{,}73 \times 10^4$	13,1	0,268
BK_16b	No	3,00	Sí	0,433	0,437	7,75	7,80	$8{,}12 \times 10^4$	$8{,}07 \times 10^4$	22,1	0,714
BK_16b	No	3,00	No	0,241	0,241	7,85	7,87	$4{,}76 \times 10^4$	$4{,}73 \times 10^4$	12,3	0,274
BK_16b	Sí	4,00	Sí	0,369	0,371	7,25	7,32	$8{,}02 \times 10^4$	$8{,}01 \times 10^4$	33,4	1,65
BK_16b	Sí	4,00	No	0,154	0,155	7,84	7,87	$3{,}12 \times 10^4$	$3{,}10 \times 10^4$	13,3	0,636
BK_16b	No	4,00	Sí	0,369	0,371	7,25	7,32	$8{,}02 \times 10^4$	$8{,}01 \times 10^4$	33,4	1,65
BK_16b	No	4,00	No	0,154	0,155	7,84	7,87	$3{,}12 \times 10^4$	$3{,}10 \times 10^4$	14,0	0,661
BK_16b	Sí	5,00	Sí	0,337	0,339	6,70	6,80	$7{,}28 \times 10^4$	$7{,}18 \times 10^4$	51,5	3,17
BK_16b	Sí	5,00	No	0,136	0,138	7,39	7,44	$2{,}41 \times 10^4$	$2{,}39 \times 10^4$	23,4	1,61
BK_16b	No	5,00	Sí	0,337	0,339	6,70	6,80	$7{,}28 \times 10^4$	$7{,}18 \times 10^4$	50,3	3,19
BK_16b	No	5,00	No	0,136	0,138	7,39	7,44	$2{,}41 \times 10^4$	$2{,}39 \times 10^4$	24,1	1,71
BK_16b	Sí	6,00	Sí	0,300	0,305	6,16	6,28	$7{,}30 \times 10^4$	$7{,}19 \times 10^4$	77,5	5,72
BK_16b	Sí	6,00	No	0,0920	0,0930	7,37	7,44	$1{,}60 \times 10^4$	$1{,}61 \times 10^4$	33,8	2,57
BK_16b	No	6,00	Sí	0,300	0,305	6,16	6,28	$7{,}30 \times 10^4$	$7{,}19 \times 10^4$	77,6	5,76
BK_16b	No	6,00	No	0,0920	0,0930	7,37	7,44	$1{,}60 \times 10^4$	$1{,}61 \times 10^4$	35,0	2,68
BK_16b	Sí	8,00	Sí	0,274	0,281	5,12	5,34	$6{,}86 \times 10^4$	$6{,}86 \times 10^4$	218	18,8
BK_16b	Sí	8,00	No	0,0480	0,0490	6,83	6,97	$3{,}34 \times 10^4$	$3{,}31 \times 10^4$	97,5	8,83
BK_16b	No	8,00	Sí	0,274	0,281	5,12	5,34	$6{,}86 \times 10^4$	$6{,}86 \times 10^4$	214	19,0
BK_16b	No	8,00	No	0,0480	0,0490	6,83	6,97	$3{,}34 \times 10^4$	$3{,}31 \times 10^4$	98,2	9,04
BK_16b	Sí	10,0	Sí	0,237	0,251	4,06	4,42	$6{,}68 \times 10^4$	$6{,}62 \times 10^4$	817	56,3
BK_16b	Sí	10,0	No	0,0240	0,0250	6,22	6,49	$6{,}61 \times 10^4$	$6{,}57 \times 10^4$	296	26,2
BK_16b	No	10,0	Sí	0,236	0,252	4,06	4,42	$6{,}68 \times 10^4$	$6{,}62 \times 10^4$	857	56,7
BK_16b	No	10,0	No	0,0240	0,0250	6,22	6,49	$6{,}61 \times 10^4$	$6{,}57 \times 10^4$	305	27,0
CSkipA_16b	Sí	3,00	Sí	0,456	0,460	7,30	7,36	$7{,}93 \times 10^4$	$7{,}87 \times 10^4$	21,8	0,677
CSkipA_16b	Sí	3,00	No	0,237	0,240	7,84	7,88	$4{,}74 \times 10^4$	$4{,}71 \times 10^4$	11,1	0,296
CSkipA_16b	No	3,00	Sí	0,456	0,460	7,30	7,36	$7{,}93 \times 10^4$	$7{,}87 \times 10^4$	21,2	0,689
CSkipA_16b	No	3,00	No	0,237	0,240	7,84	7,88	$4{,}74 \times 10^4$	$4{,}71 \times 10^4$	11,9	0,333
CSkipA_16b	Sí	4,00	Sí	0,487	0,492	7,00	7,08	$8{,}10 \times 10^4$	$8{,}11 \times 10^4$	33,1	1,59
CSkipA_16b	Sí	4,00	No	0,217	0,221	7,83	7,88	$3{,}21 \times 10^4$	$3{,}19 \times 10^4$	13,9	0,632
CSkipA_16b	No	4,00	Sí	0,487	0,492	7,00	7,08	$8{,}10 \times 10^4$	$8{,}11 \times 10^4$	32,2	1,61
CSkipA_16b	No	4,00	No	0,217	0,221	7,83	7,88	$3{,}21 \times 10^4$	$3{,}19 \times 10^4$	13,7	0,673
CSkipA_16b	Sí	5,00	Sí	0,410	0,411	6,44	6,54	$7{,}30 \times 10^4$	$7{,}24 \times 10^4$	49,8	3,14
CSkipA_16b	Sí	5,00	No	0,117	0,117	7,38	7,43	$2{,}38 \times 10^4$	$2{,}38 \times 10^4$	21,7	1,54
CSkipA_16b	No	5,00	Sí	0,410	0,411	6,44	6,54	$7{,}30 \times 10^4$	$7{,}24 \times 10^4$	48,8	3,10
CSkipA_16b	No	5,00	No	0,117	0,117	7,38	7,43	$2{,}38 \times 10^4$	$2{,}38 \times 10^4$	22,4	1,60
CSkipA_16b	Sí	6,00	Sí	0,411	0,419	6,00	6,13	$7{,}28 \times 10^4$	$7{,}20 \times 10^4$	75,4	5,74
CSkipA_16b	Sí	6,00	No	0,0950	0,0970	7,36	7,44	$1{,}56 \times 10^4$	$1{,}53 \times 10^4$	33,2	2,60
CSkipA_16b	No	6,00	Sí	0,411	0,419	6,00	6,13	$7{,}28 \times 10^4$	$7{,}20 \times 10^4$	75,8	5,74
CSkipA_16b	No	6,00	No	0,0950	0,0970	7,36	7,44	$1{,}56 \times 10^4$	$1{,}53 \times 10^4$	33,6	2,64
CSkipA_16b	Sí	8,00	Sí	0,377	0,389	5,08	5,28	$6{,}86 \times 10^4$	$6{,}85 \times 10^4$	205	18,1
CSkipA_16b	Sí	8,00	No	0,0420	0,0430	6,82	6,96	$9{,}68 \times 10^3$	$9{,}53 \times 10^3$	85,4	8,41
CSkipA_16b	No	8,00	Sí	0,377	0,389	5,08	5,28	$6{,}86 \times 10^4$	$6{,}85 \times 10^4$	213	18,3
CSkipA_16b	No	8,00	No	0,0420	0,0430	6,82	6,96	$9{,}68 \times 10^3$	$9{,}53 \times 10^3$	89,5	8,72
CSkipA_16b	Sí	10,0	Sí	0,333	0,355	4,12	4,46	$6{,}68 \times 10^4$	$6{,}66 \times 10^4$	776	55,5
CSkipA_16b	Sí	10,0	No	0,0230	0,0230	6,22	6,47	$1{,}68 \times 10^4$	$1{,}67 \times 10^4$	309	26,5
CSkipA_16b	No	10,0	Sí	0,333	0,355	4,12	4,46	$6{,}68 \times 10^4$	$6{,}66 \times 10^4$	842	56,0
CSkipA_16b	No	10,0	No	0,0230	0,0230	6,22	6,47	$1{,}68 \times 10^4$	$1{,}67 \times 10^4$	318	27,4
KS_16b	Sí	3,00	Sí	0,484	0,483	7,81	7,86	$8{,}06 \times 10^4$	$7{,}98 \times 10^4$	17,5	0,567
KS_16b	Sí	3,00	No	0,254	0,252	7,84	7,88	$4{,}40 \times 10^4$	$4{,}41 \times 10^4$	10,3	0,207
KS_16b	No	3,00	Sí	0,484	0,483	7,81	7,86	$8{,}06 \times 10^4$	$7{,}98 \times 10^4$	17,2	0,587

Circuito	Resíntesis	Profundidad máxima	Un árbol por salida	MRED	MRED validación	HD	HD validación	WCE	WCE validación	Tiempo de ejecución (s)	Área (% del área original)
KS_16b	No	3,00	No	0,254	0,252	7,84	7,88	$4{,}40 \times 10^4$	$4{,}41 \times 10^4$	10,3	0,212
KS_16b	Sí	4,00	Sí	0,443	0,441	7,35	7,44	$7{,}91 \times 10^4$	$7{,}95 \times 10^4$	25,8	1,27
KS_16b	Sí	4,00	No	0,186	0,184	7,83	7,87	$2{,}76 \times 10^4$	$2{,}77 \times 10^4$	11,4	0,441
KS_16b	No	4,00	Sí	0,443	0,441	7,35	7,44	$7{,}91 \times 10^4$	$7{,}95 \times 10^4$	26,3	1,28
KS_16b	No	4,00	No	0,186	0,184	7,83	7,87	$2{,}76 \times 10^4$	$2{,}77 \times 10^4$	10,9	0,450
KS_16b	Sí	5,00	Sí	0,431	0,429	6,89	6,99	$7{,}26 \times 10^4$	$7{,}29 \times 10^4$	39,8	2,57
KS_16b	Sí	5,00	No	0,126	0,126	7,38	7,43	$2{,}42 \times 10^4$	$2{,}43 \times 10^4$	17,7	1,21
KS_16b	No	5,00	Sí	0,431	0,429	6,89	6,99	$7{,}26 \times 10^4$	$7{,}29 \times 10^4$	40,1	2,46
KS_16b	No	5,00	No	0,126	0,126	7,38	7,43	$2{,}42 \times 10^4$	$2{,}43 \times 10^4$	17,8	1,17
KS_16b	Sí	6,00	Sí	0,375	0,375	6,32	6,46	$7{,}29 \times 10^4$	$7{,}22 \times 10^4$	63,6	4,53
KS_16b	Sí	6,00	No	0,0890	0,0890	7,35	7,44	$1{,}62 \times 10^4$	$1{,}61 \times 10^4$	26,6	2,00
KS_16b	No	6,00	Sí	0,375	0,375	6,32	6,46	$7{,}29 \times 10^4$	$7{,}22 \times 10^4$	61,9	4,64
KS_16b	No	6,00	No	0,0890	0,0890	7,35	7,44	$1{,}62 \times 10^4$	$1{,}61 \times 10^4$	27,4	2,11
KS_16b	Sí	8,00	Sí	0,299	0,305	5,20	5,44	$6{,}89 \times 10^4$	$6{,}81 \times 10^4$	176	14,8
KS_16b	Sí	8,00	No	0,0430	0,0430	6,80	6,97	$7{,}98 \times 10^3$	$8{,}05 \times 10^3$	66,6	6,31
KS_16b	No	8,00	Sí	0,299	0,305	5,20	5,44	$6{,}89 \times 10^4$	$6{,}81 \times 10^4$	184	15,1
KS_16b	No	8,00	No	0,0430	0,0430	6,80	6,97	$7{,}98 \times 10^3$	$8{,}05 \times 10^3$	67,5	6,39
KS_16b	Sí	10,0	Sí	0,242	0,258	4,12	4,52	$6{,}72 \times 10^4$	$6{,}73 \times 10^4$	653	44,2
KS_16b	Sí	10,0	No	0,0230	0,0240	6,18	6,49	$3{,}43 \times 10^4$	$3{,}43 \times 10^4$	244	20,3
KS_16b	No	10,0	Sí	0,242	0,258	4,12	4,52	$6{,}72 \times 10^4$	$6{,}73 \times 10^4$	669	44,6
KS_16b	No	10,0	No	0,0230	0,0240	6,18	6,49	$3{,}43 \times 10^4$	$3{,}43 \times 10^4$	257	20,9
LFA_16b	Sí	3,00	Sí	0,505	0,510	8,02	8,08	$8{,}05 \times 10^4$	$7{,}90 \times 10^4$	23,1	0,691
LFA_16b	Sí	3,00	No	0,275	0,281	7,85	7,88	$4{,}12 \times 10^4$	$4{,}10 \times 10^4$	13,2	0,270
LFA_16b	No	3,00	Sí	0,505	0,510	8,02	8,08	$8{,}05 \times 10^4$	$7{,}90 \times 10^4$	22,6	0,693
LFA_16b	No	3,00	No	0,275	0,281	7,85	7,88	$4{,}12 \times 10^4$	$4{,}10 \times 10^4$	12,1	0,284
LFA_16b	Sí	4,00	Sí	0,488	0,495	7,77	7,85	$8{,}11 \times 10^4$	$7{,}99 \times 10^4$	35,8	1,75
LFA_16b	Sí	4,00	No	0,226	0,231	7,84	7,88	$3{,}20 \times 10^4$	$3{,}15 \times 10^4$	13,4	0,549
LFA_16b	No	4,00	Sí	0,488	0,495	7,77	7,85	$8{,}11 \times 10^4$	$7{,}99 \times 10^4$	35,1	1,76
LFA_16b	No	4,00	No	0,226	0,231	7,84	7,88	$3{,}20 \times 10^4$	$3{,}15 \times 10^4$	13,7	0,586
LFA_16b	Sí	5,00	Sí	0,468	0,474	7,36	7,47	$7{,}24 \times 10^4$	$7{,}17 \times 10^4$	53,7	3,30
LFA_16b	Sí	5,00	No	0,123	0,124	7,38	7,43	$2{,}43 \times 10^4$	$2{,}43 \times 10^4$	22,5	1,46
LFA_16b	No	5,00	Sí	0,468	0,474	7,36	7,47	$7{,}24 \times 10^4$	$7{,}17 \times 10^4$	54,2	3,33
LFA_16b	No	5,00	No	0,123	0,124	7,38	7,43	$2{,}43 \times 10^4$	$2{,}43 \times 10^4$	22,1	1,48
LFA_16b	Sí	6,00	Sí	0,464	0,470	6,92	7,06	$7{,}28 \times 10^4$	$7{,}16 \times 10^4$	87,7	6,34
LFA_16b	Sí	6,00	No	0,0830	0,0850	7,36	7,42	$1{,}59 \times 10^4$	$1{,}57 \times 10^4$	32,9	2,52
LFA_16b	No	6,00	Sí	0,464	0,470	6,92	7,06	$7{,}28 \times 10^4$	$7{,}16 \times 10^4$	86,8	6,56
LFA_16b	No	6,00	No	0,0830	0,0850	7,36	7,42	$1{,}59 \times 10^4$	$1{,}57 \times 10^4$	33,3	2,53
LFA_16b	Sí	8,00	Sí	0,413	0,428	5,94	6,18	$6{,}87 \times 10^4$	$6{,}86 \times 10^4$	265	21,3
LFA_16b	Sí	8,00	No	0,0450	0,0460	6,83	6,96	$8{,}13 \times 10^3$	$8{,}16 \times 10^3$	93,0	8,63
LFA_16b	No	8,00	Sí	0,413	0,428	5,94	6,18	$6{,}87 \times 10^4$	$6{,}86 \times 10^4$	267	21,5
LFA_16b	No	8,00	No	0,0450	0,0460	6,83	6,96	$8{,}13 \times 10^3$	$8{,}16 \times 10^3$	100	8,93
LFA_16b	Sí	10,0	Sí	0,361	0,388	4,84	5,25	$6{,}71 \times 10^4$	$6{,}65 \times 10^4$	973	65,0
LFA_16b	Sí	10,0	No	0,0240	0,0240	6,22	6,49	$3{,}39 \times 10^4$	$3{,}40 \times 10^4$	310	26,5
LFA_16b	No	10,0	Sí	0,362	0,388	4,84	5,25	$6{,}71 \times 10^4$	$6{,}65 \times 10^4$	985	65,7
LFA_16b	No	10,0	No	0,0240	0,0240	6,22	6,49	$3{,}39 \times 10^4$	$3{,}40 \times 10^4$	328	27,3
WT_8b	Sí	3,00	Sí	0,531		5,42		$4{,}86 \times 10^4$		2,69	0,166
WT_8b	Sí	3,00	No	0,986		6,13		$6{,}48 \times 10^4$		1,30	0,0110
WT_8b	No	3,00	Sí	0,484		5,46		$3{,}23 \times 10^4$		2,37	0,142
WT_8b	No	3,00	No	0,986		6,13		$6{,}48 \times 10^4$		1,11	0,0110
WT_8b	Sí	4,00	Sí	0,368		4,84		$2{,}42 \times 10^4$		3,46	0,295
WT_8b	Sí	4,00	No	0,966		5,91		$6{,}45 \times 10^4$		1,58	0,0470
WT_8b	No	4,00	Sí	0,370		4,85		$2{,}42 \times 10^4$		3,29	0,298
WT_8b	No	4,00	No	0,966		5,91		$6{,}45 \times 10^4$		1,39	0,0446
WT_8b	Sí	5,00	Sí	0,302		4,39		$2{,}38 \times 10^4$		4,98	0,535
WT_8b	Sí	5,00	No	0,888		5,74		$6{,}45 \times 10^4$		1,95	0,11
WT_8b	No	5,00	Sí	0,310		4,38		$1{,}96 \times 10^4$		4,71	0,553
WT_8b	No	5,00	No	0,888		5,74		$6{,}45 \times 10^4$		1,72	0,104
WT_8b	Sí	6,00	Sí	0,253		3,87		$1{,}97 \times 10^4$		7,63	0,962
WT_8b	Sí	6,00	No	0,751		5,52		$6{,}35 \times 10^4$		3,09	0,369
WT_8b	No	6,00	Sí	0,263		3,86		$1{,}97 \times 10^4$		6,98	0,984
WT_8b	No	6,00	No	0,751		5,52		$6{,}35 \times 10^4$		2,76	0,374
WT_8b	Sí	8,00	Sí	0,168		3,01		$1{,}34 \times 10^4$		17,9	2,67
WT_8b	Sí	8,00	No	0,498		5,07		$6{,}14 \times 10^4$		9,65	1,48
WT_8b	No	8,00	Sí	0,169		3,01		$1{,}33 \times 10^4$		17,6	2,83
WT_8b	No	8,00	No	0,498		5,07		$6{,}14 \times 10^4$		8,74	1,48
WT_8b	Sí	10,0	Sí	0,105		2,22		$2{,}51 \times 10^4$		60,3	7,69
WT_8b	Sí	10,0	No	0,273		4,47		$5{,}73 \times 10^4$		33,0	5,05
WT_8b	No	10,0	Sí	0,106		2,20		$3{,}38 \times 10^4$		54,4	7,73
WT_8b	No	10,0	No	0,274		4,47		$5{,}73 \times 10^4$		30,9	5,22
int2float	Sí	3,00	Sí	0,144		1,61		140		0,588	0,0812
int2float	Sí	3,00	No	0,179		2,96		84,0		0,567	0,0519

Circuito	Resíntesis	Profundidad máxima	Un árbol por salida	MRED	MRED validación	HD	HD validación	WCE	WCE validación	Tiempo de ejecución (s)	Área (% del área original)
int2float	No	3,00	Sí	0,163		2,34		150		0,380	0,0839
int2float	No	3,00	No	0,179		2,96		84,0		0,365	0,0519
int2float	Sí	4,00	Sí	0,139		1,18		140		0,658	0,145
int2float	Sí	4,00	No	0,178		2,86		120		0,647	0,0866
int2float	No	4,00	Sí	0,142		1,59		120		0,404	0,170
int2float	No	4,00	No	0,178		2,86		120		0,381	0,0919
int2float	Sí	5,00	Sí	0,104		1,02		120		0,685	0,261
int2float	Sí	5,00	No	0,130		1,45		140		0,658	0,186
int2float	No	5,00	Sí	0,110		1,39		120		0,424	0,328
int2float	No	5,00	No	0,130		1,45		140		0,400	0,186
int2float	Sí	6,00	Sí	0,0630		0,457		80,0		0,737	0,358
int2float	Sí	6,00	No	0,0730		0,701		80,0		0,694	0,284
int2float	No	6,00	Sí	0,0640		0,501		80,0		0,453	0,419
int2float	No	6,00	No	0,0730		0,701		80,0		0,426	0,265
int2float	Sí	8,00	Sí	0,0100		0,166		80,0		0,851	0,660
int2float	Sí	8,00	No	0,0150		0,192		110		0,809	0,539
int2float	No	8,00	Sí	0,0100		0,208		80,0		0,512	0,686
int2float	No	8,00	No	0,0160		0,188		140		0,487	0,549
int2float	Sí	10,0	Sí	$3{,}00 \times 10^{-3}$		0,113		40,0		1,11	1,13
int2float	Sí	10,0	No	$3{,}00 \times 10^{-3}$		0,0320		71,0		0,907	0,760
int2float	No	10,0	Sí	$1{,}00 \times 10^{-3}$		0,0140		41,0		0,591	0,916
int2float	No	10,0	No	$3{,}00 \times 10^{-3}$		0,0320		80,0		0,551	0,762
RCA_4b	Sí	3,00	Sí	0,608		2,13		19,0		0,533	0,104
RCA_4b	Sí	3,00	No	0,326		1,88		11,0		0,547	0,261
RCA_4b	No	3,00	Sí	0,588		1,88		18,0		0,346	0,216
RCA_4b	No	3,00	No	0,326		1,88		11,0		0,345	0,269
RCA_4b	Sí	4,00	Sí	0,371		1,38		19,0		0,628	1,05
RCA_4b	Sí	4,00	No	0,569		1,88		19,0		0,575	0,351
RCA_4b	No	4,00	Sí	0,548		1,81		19,0		0,357	0,619
RCA_4b	No	4,00	No	0,569		1,88		19,0		0,347	0,351
RCA_4b	Sí	5,00	Sí	0,339		1,38		17,0		0,621	0,925
RCA_4b	Sí	5,00	No	0,157		1,44		5,00		0,627	1,25
RCA_4b	No	5,00	Sí	0,332		1,25		17,0		0,401	1,96
RCA_4b	No	5,00	No	0,157		1,44		5,00		0,379	1,16
RCA_4b	Sí	6,00	Sí	0,237		0,789		17,0		0,749	2,73
RCA_4b	Sí	6,00	No	0,354		1,44		17,0		0,615	0,709
RCA_4b	No	6,00	Sí	0,254		0,977		17,0		0,449	3,60
RCA_4b	No	6,00	No	0,354		1,44		17,0		0,373	0,701
RCA_4b	Sí	8,00	Sí	0,0990		0,258		16,0		0,917	4,52
RCA_4b	Sí	8,00	No	0,201		0,969		16,0		0,920	5,39
RCA_4b	No	8,00	Sí	0,0770		0,141		16,0		0,483	3,30
RCA_4b	No	8,00	No	0,201		0,969		16,0		0,529	5,51
RCA_4b	Sí	10,0	Sí	0,0		0,0		0		0,834	0,948
RCA_4b	Sí	10,0	No	0,0		0,0		0		0,888	0,970
RCA_4b	No	10,0	Sí	0,0		0,0		0		0,546	1,01
RCA_4b	No	10,0	No	0,0		0,0		0		0,645	0,985
dec	Sí	3,00	Sí	1,00		1,50		$1{,}70 \times 10^{39}$		0,475	0,0
dec	Sí	3,00	No	1,00		1,50		$1{,}70 \times 10^{39}$		0,361	0,0
dec	No	3,00	Sí	1,00		1,50		$1{,}70 \times 10^{39}$		0,386	0,0
dec	No	3,00	No	1,00		1,50		$1{,}70 \times 10^{39}$		0,277	0,0
dec	Sí	4,00	Sí	1,00		1,50		$1{,}70 \times 10^{39}$		0,481	0,0
dec	Sí	4,00	No	1,00		1,50		$1{,}70 \times 10^{39}$		0,366	0,0
dec	No	4,00	Sí	1,00		1,50		$1{,}70 \times 10^{39}$		0,388	0,0
dec	No	4,00	No	1,00		1,50		$1{,}70 \times 10^{39}$		0,276	0,0
dec	Sí	5,00	Sí	1,00		1,50		$1{,}70 \times 10^{39}$		0,477	0,0
dec	Sí	5,00	No	1,00		1,50		$1{,}70 \times 10^{39}$		0,379	0,0
dec	No	5,00	Sí	1,00		1,50		$1{,}70 \times 10^{39}$		0,388	0,0
dec	No	5,00	No	1,00		1,50		$1{,}70 \times 10^{39}$		0,289	0,0
dec	Sí	6,00	Sí	1,00		1,50		$1{,}70 \times 10^{39}$		0,486	0,0
dec	Sí	6,00	No	1,00		1,50		$1{,}70 \times 10^{39}$		0,414	0,0

Circuito	Resíntesis	Profundidad máxima	Un árbol por salida	MRED	MRED validación	HD	HD validación	WCE	WCE validación	Tiempo de ejecución (s)	Área (% del área original)
dec	No	6,00	Sí	1,00		1,50		$1{,}70 \times 10^{39}$		0,391	0,0
dec	No	6,00	No	1,00		1,50		$1{,}70 \times 10^{39}$		0,322	0,0
dec	Sí	8,00	Sí	0,0		0,0		0		2,21	1,93
dec	Sí	8,00	No	0,0		0,0		0		2,05	1,23
dec	No	8,00	Sí	0,0		0,0		0		1,42	1,98
dec	No	8,00	No	0,0		0,0		0		1,47	1,89
dec	Sí	10,0	Sí	0,0		0,0		0		2,22	1,94
dec	Sí	10,0	No	0,0		0,0		0		2,05	1,23
dec	No	10,0	Sí	0,0		0,0		0		1,41	1,96
dec	No	10,0	No	0,0		0,0		0		1,49	1,89

Tabla 6.4: Resultados recolectados para el método `inconst`.

Circuito	Resíntesis	Umbral de error	MRED	MRED validación	HD	HD validación	WCE	WCE validación	Tiempo de ejecución (s)	Área (% del área original)
DTree	Sí	0,0100	$2{,}00 \times 10^{-3}$	0,0380	$2{,}00 \times 10^{-3}$	$7{,}00 \times 10^{-3}$	15,0	15,0	12,6	0,917
DTree	No	0,0100	0,0	0,0	0,0	0,0	0,0	0,0	3,54	0,996
DTree	Sí	0,0500	0,0350	0,0700	0,0150	0,0240	15,0	15,0	44,3	0,795
DTree	No	0,0500	0,0410	0,0450	0,0160	0,0120	15,0	15,0	21,0	0,961
DTree	Sí	0,100	0,0790	0,103	0,0760	0,0900	15,0	15,0	78,9	0,638
DTree	No	0,100	0,0950	0,109	0,0600	0,0520	15,0	15,0	55,3	0,898
DTree	Sí	0,250	0,140	0,136	0,142	0,135	15,0	15,0	31,1	0,557
DTree	No	0,250	0,247	0,262	0,307	0,300	15,0	15,0	20,0	0,772
DTree	Sí	0,500	0,423	0,444	0,347	0,340	15,0	15,0	22,9	0,466
DTree	No	0,500	0,409	0,430	0,385	0,367	16,0	15,0	13,7	0,698
barshift_128b	Sí	0,0100	$7{,}00 \times 10^{-3}$	0,281	0,488	22,0	$1{,}70 \times 10^{38}$	$2{,}34 \times 10^{38}$	8,63	0,840
barshift_128b	No	0,0100	$7{,}00 \times 10^{-3}$	$7{,}00 \times 10^{-3}$	0,488	0,525	$1{,}70 \times 10^{38}$	$1{,}70 \times 10^{38}$	3,85	0,999
barshift_128b	Sí	0,0500	$7{,}00 \times 10^{-3}$	0,281	0,488	22,0	$1{,}70 \times 10^{38}$	$2{,}34 \times 10^{38}$	8,67	0,840
barshift_128b	No	0,0500	0,0120	0,0140	0,995	1,04	$1{,}70 \times 10^{38}$	$1{,}70 \times 10^{38}$	5,29	0,998
barshift_128b	Sí	0,100	$7{,}00 \times 10^{-3}$	0,281	0,488	22,0	$1{,}70 \times 10^{38}$	$2{,}34 \times 10^{38}$	8,61	0,840
barshift_128b	No	0,100	0,0120	0,0140	0,995	1,04	$1{,}70 \times 10^{38}$	$1{,}70 \times 10^{38}$	5,25	0,998
barshift_128b	Sí	0,250	0,0120	0,0140	0,995	1,04	$1{,}70 \times 10^{38}$	$1{,}70 \times 10^{38}$	51,6	0,995
barshift_128b	No	0,250	0,0120	0,0140	0,995	1,04	$1{,}70 \times 10^{38}$	$1{,}70 \times 10^{38}$	23,4	0,998
barshift_128b	Sí	0,500	0,0120	0,0140	0,995	1,04	$1{,}70 \times 10^{38}$	$1{,}70 \times 10^{38}$	51,3	0,995
barshift_128b	No	0,500	0,0120	0,0140	0,995	1,04	$1{,}70 \times 10^{38}$	$1{,}70 \times 10^{38}$	23,5	0,998
sobel	Sí	0,0100	$6{,}00 \times 10^{-3}$	0,0215	0,363	0,425	129	89,0	7,77	1,02
sobel	No	0,0100	$8{,}00 \times 10^{-3}$	0,0230	0,481	0,465	130	122	5,16	0,994
sobel	Sí	0,0500	0,0360	0,0570	1,12	1,19	158	159	51,8	0,808
sobel	No	0,0500	0,0440	0,0470	1,14	1,16	135	192	33,2	0,940
sobel	Sí	0,100	0,0915	0,122	1,54	1,59	203	209	77,7	0,651
sobel	No	0,100	0,0860	0,0950	1,44	1,47	171	146	79,6	0,870
sobel	Sí	0,250	0,195	0,187	1,82	1,79	164	174	29,3	0,361
sobel	No	0,250	0,219	0,238	1,99	2,04	230	200	128	0,670
sobel	Sí	0,500	0,443	0,453	3,68	3,64	165	165	11,0	0,0
sobel	No	0,500	0,476	0,493	2,98	2,94	247	249	797	0,388
BK_32b	Sí	0,0100	$8{,}50 \times 10^{-3}$	0,0115	13,8	14,2	$9{,}99 \times 10^{7}$	$1{,}33 \times 10^{8}$	34,1	0,159
BK_32b	No	0,0100	$8{,}00 \times 10^{-3}$	$8{,}00 \times 10^{-3}$	13,5	13,4	$6{,}67 \times 10^{7}$	$6{,}64 \times 10^{7}$	85,0	0,250
BK_32b	Sí	0,0500	0,0360	0,0505	14,8	15,1	$4{,}00 \times 10^{8}$	$5{,}27 \times 10^{8}$	35,1	0,0956
BK_32b	No	0,0500	0,0440	0,0450	14,9	15,0	$4{,}01 \times 10^{8}$	$3{,}96 \times 10^{8}$	104	0,137
BK_32b	Sí	0,100	0,0600	0,110	15,4	15,5	$5{,}33 \times 10^{8}$	$1{,}05 \times 10^{9}$	34,8	0,0699
BK_32b	No	0,100	0,0850	0,0850	15,2	15,2	$6{,}66 \times 10^{8}$	$6{,}63 \times 10^{8}$	106	0,119
BK_32b	Sí	0,250	0,164	0,163	15,9	15,9	$1{,}60 \times 10^{9}$	$1{,}60 \times 10^{9}$	9,29	0,0428
BK_32b	No	0,250	0,164	0,163	15,9	15,9	$1{,}60 \times 10^{9}$	$1{,}60 \times 10^{9}$	24,9	0,0733
BK_32b	Sí	0,500	0,275	0,271	16,2	16,2	$2{,}13 \times 10^{9}$	$2{,}11 \times 10^{9}$	6,80	0,0143
BK_32b	No	0,500	0,275	0,271	16,2	16,2	$2{,}13 \times 10^{9}$	$2{,}11 \times 10^{9}$	23,6	0,0295
KS_32b	Sí	0,0100	$8{,}50 \times 10^{-3}$	0,0125	13,9	14,2	$9{,}99 \times 10^{7}$	$1{,}32 \times 10^{8}$	32,3	0,133
KS_32b	No	0,0100	$3{,}00 \times 10^{-3}$	$3{,}00 \times 10^{-3}$	13,0	13,0	$2{,}51 \times 10^{7}$	$2{,}49 \times 10^{7}$	141	0,280
KS_32b	Sí	0,0500	0,0345	0,0540	14,9	15,2	$4{,}00 \times 10^{8}$	$5{,}31 \times 10^{8}$	33,2	0,0768
KS_32b	No	0,0500	$9{,}00 \times 10^{-3}$	0,0100	14,0	14,0	$8{,}36 \times 10^{7}$	$8{,}32 \times 10^{7}$	161	0,215
KS_32b	Sí	0,100	0,0590	0,120	15,5	15,6	$5{,}34 \times 10^{8}$	$1{,}06 \times 10^{9}$	33,1	0,0561
KS_32b	No	0,100	0,0490	0,0500	15,0	15,0	$4{,}00 \times 10^{8}$	$3{,}96 \times 10^{8}$	191	0,147
KS_32b	Sí	0,250	0,138	0,148	15,9	15,9	$1{,}20 \times 10^{9}$	$1{,}20 \times 10^{9}$	12,0	0,0344
KS_32b	No	0,250	0,125	0,134	15,9	15,9	$1{,}06 \times 10^{9}$	$1{,}07 \times 10^{9}$	44,8	0,0642
KS_32b	Sí	0,500	0,285	0,301	16,2	16,3	$2{,}23 \times 10^{9}$	$2{,}20 \times 10^{9}$	7,86	0,0115
KS_32b	No	0,500	0,308	0,314	16,1	16,1	$3{,}21 \times 10^{9}$	$3{,}16 \times 10^{9}$	43,6	0,0405
CLA_16b	Sí	0,0100	$8{,}00 \times 10^{-3}$	0,0260	5,14	5,49	$2{,}19 \times 10^{4}$	$2{,}21 \times 10^{4}$	17,8	0,485
CLA_16b	No	0,0100	$6{,}00 \times 10^{-3}$	$6{,}00 \times 10^{-3}$	5,48	5,51	763	764	31,5	0,515
CLA_16b	Sí	0,0500	0,0410	0,0790	6,51	6,81	$2{,}26 \times 10^{4}$	$1{,}01 \times 10^{4}$	23,3	0,244
CLA_16b	No	0,0500	0,0450	0,0440	6,97	7,01	$6{,}12 \times 10^{3}$	$6{,}08 \times 10^{3}$	43,0	0,285
CLA_16b	Sí	0,100	0,0600	0,154	7,43	7,67	$8{,}28 \times 10^{3}$	$2{,}41 \times 10^{4}$	26,0	0,123
CLA_16b	No	0,100	0,0740	0,0680	7,42	7,42	$1{,}23 \times 10^{4}$	$1{,}22 \times 10^{4}$	45,0	0,243
CLA_16b	Sí	0,250	0,163	0,162	7,86	7,93	$2{,}43 \times 10^{4}$	$2{,}44 \times 10^{4}$	7,56	0,0827
CLA_16b	No	0,250	0,124	0,121	7,87	7,86	$1{,}63 \times 10^{4}$	$1{,}62 \times 10^{4}$	11,6	0,134
CLA_16b	Sí	0,500	0,362	0,359	8,25	8,29	$4{,}89 \times 10^{4}$	$4{,}88 \times 10^{4}$	8,15	0,0386
CLA_16b	No	0,500	0,267	0,257	8,24	8,23	$3{,}25 \times 10^{4}$	$3{,}24 \times 10^{4}$	12,1	0,0754
BK_16b	Sí	0,0100	$5{,}00 \times 10^{-3}$	$5{,}00 \times 10^{-3}$	5,48	5,51	679	676	6,88	0,424
BK_16b	No	0,0100	$8{,}00 \times 10^{-3}$	$9{,}00 \times 10^{-3}$	5,99	5,97	$1{,}53 \times 10^{3}$	$1{,}53 \times 10^{3}$	8,27	0,438
BK_16b	Sí	0,0500	0,0430	0,0450	6,95	6,97	$6{,}82 \times 10^{3}$	$6{,}76 \times 10^{3}$	6,02	0,235
BK_16b	No	0,0500	0,0450	0,0450	6,97	6,95	$6{,}12 \times 10^{3}$	$6{,}10 \times 10^{3}$	10,4	0,278
BK_16b	Sí	0,100	0,0840	0,0870	7,41	7,45	$1{,}09 \times 10^{4}$	$1{,}09 \times 10^{4}$	7,94	0,156
BK_16b	No	0,100	0,0840	0,0840	6,97	6,95	$8{,}16 \times 10^{3}$	$8{,}15 \times 10^{3}$	10,1	0,261
BK_16b	Sí	0,250	0,200	0,210	7,85	7,90	$2{,}70 \times 10^{4}$	$2{,}72 \times 10^{4}$	6,36	0,105
BK_16b	No	0,250	0,166	0,168	7,89	7,87	$2{,}45 \times 10^{4}$	$2{,}44 \times 10^{4}$	10,9	0,148
BK_16b	Sí	0,500	0,398	0,413	8,23	8,27	$4{,}34 \times 10^{4}$	$4{,}32 \times 10^{4}$	7,42	0,0292
BK_16b	No	0,500	0,419	0,420	8,26	8,23	$6{,}55 \times 10^{4}$	$6{,}54 \times 10^{4}$	10,4	0,119

Circuito	Resíntesis	Umbral de error	MRED	MRED validación	HD	HD validación	WCE	WCE validación	Tiempo de ejecución (s)	Área (% del área original)
CSkipA_16b	Sí	0,0100	$5{,}00 \times 10^{-3}$	$5{,}00 \times 10^{-3}$	5,48	5,53	682	672	7,08	0,424
CSkipA_16b	No	0,0100	$8{,}00 \times 10^{-3}$	$9{,}00 \times 10^{-3}$	5,98	6,02	$1{,}54 \times 10^{3}$	$1{,}52 \times 10^{3}$	8,48	0,438
CSkipA_16b	Sí	0,0500	0,0430	0,0420	6,95	6,99	$6{,}79 \times 10^{3}$	$6{,}77 \times 10^{3}$	6,09	0,235
CSkipA_16b	No	0,0500	0,0450	0,0450	6,95	6,98	$6{,}12 \times 10^{3}$	$6{,}09 \times 10^{3}$	10,6	0,278
CSkipA_16b	Sí	0,100	0,0840	0,0830	7,43	7,47	$1{,}09 \times 10^{4}$	$1{,}09 \times 10^{4}$	7,48	0,156
CSkipA_16b	No	0,100	0,0830	0,0840	6,95	6,97	$8{,}17 \times 10^{3}$	$8{,}14 \times 10^{3}$	10,2	0,261
CSkipA_16b	Sí	0,250	0,201	0,201	7,86	7,91	$2{,}72 \times 10^{4}$	$2{,}70 \times 10^{4}$	6,22	0,105
CSkipA_16b	No	0,250	0,165	0,165	7,87	7,87	$2{,}44 \times 10^{4}$	$2{,}43 \times 10^{4}$	10,4	0,148
CSkipA_16b	Sí	0,500	0,399	0,396	8,22	8,27	$4{,}35 \times 10^{4}$	$4{,}34 \times 10^{4}$	7,21	0,0292
CSkipA_16b	No	0,500	0,421	0,421	8,26	8,26	$6{,}54 \times 10^{4}$	$6{,}53 \times 10^{4}$	10,4	0,119
KS_16b	Sí	0,0100	$4{,}00 \times 10^{-3}$	$4{,}00 \times 10^{-3}$	5,48	5,48	506	507	8,04	0,327
KS_16b	No	0,0100	$6{,}00 \times 10^{-3}$	$6{,}00 \times 10^{-3}$	5,48	5,49	762	763	9,87	0,513
KS_16b	Sí	0,0500	0,0350	0,0360	6,95	6,99	$6{,}13 \times 10^{3}$	$6{,}13 \times 10^{3}$	7,12	0,179
KS_16b	No	0,0500	$7{,}00 \times 10^{-3}$	$7{,}00 \times 10^{-3}$	5,97	5,99	$1{,}01 \times 10^{3}$	$1{,}02 \times 10^{3}$	11,2	0,422
KS_16b	Sí	0,100	0,0850	0,0890	7,21	7,25	$1{,}43 \times 10^{4}$	$1{,}42 \times 10^{4}$	7,86	0,132
KS_16b	No	0,100	0,0890	0,0910	7,42	7,46	$8{,}17 \times 10^{3}$	$8{,}18 \times 10^{3}$	15,9	0,253
KS_16b	Sí	0,250	0,190	0,195	7,87	7,91	$2{,}66 \times 10^{4}$	$2{,}64 \times 10^{4}$	6,81	0,0800
KS_16b	No	0,250	0,193	0,198	7,42	7,48	$4{,}09 \times 10^{4}$	$4{,}08 \times 10^{4}$	15,9	0,190
KS_16b	Sí	0,500	0,289	0,295	8,25	8,26	$3{,}44 \times 10^{4}$	$3{,}45 \times 10^{4}$	7,48	0,0222
KS_16b	No	0,500	0,357	0,355	7,86	7,90	$8{,}17 \times 10^{4}$	$8{,}11 \times 10^{4}$	17,6	0,113
LFA_16b	Sí	0,0100	$5{,}00 \times 10^{-3}$	$5{,}00 \times 10^{-3}$	5,25	5,24	849	842	7,39	0,434
LFA_16b	No	0,0100	$8{,}00 \times 10^{-3}$	$8{,}00 \times 10^{-3}$	5,98	5,98	$1{,}53 \times 10^{3}$	$1{,}53 \times 10^{3}$	8,54	0,438
LFA_16b	Sí	0,0500	0,0300	0,0300	6,97	6,95	$4{,}24 \times 10^{3}$	$4{,}25 \times 10^{3}$	8,67	0,218
LFA_16b	No	0,0500	0,0450	0,0440	6,95	6,99	$6{,}11 \times 10^{3}$	$6{,}05 \times 10^{3}$	10,7	0,278
LFA_16b	Sí	0,100	0,0610	0,0610	7,45	7,42	$8{,}30 \times 10^{3}$	$8{,}31 \times 10^{3}$	7,57	0,156
LFA_16b	No	0,100	0,0830	0,0830	6,95	6,98	$8{,}15 \times 10^{3}$	$8{,}10 \times 10^{3}$	10,5	0,261
LFA_16b	Sí	0,250	0,127	0,128	7,89	7,85	$1{,}64 \times 10^{4}$	$1{,}65 \times 10^{4}$	8,17	0,0875
LFA_16b	No	0,250	0,164	0,164	7,85	7,88	$2{,}44 \times 10^{4}$	$2{,}44 \times 10^{4}$	11,0	0,148
LFA_16b	Sí	0,500	0,272	0,272	8,27	8,24	$3{,}27 \times 10^{4}$	$3{,}28 \times 10^{4}$	7,37	0,0292
LFA_16b	No	0,500	0,417	0,417	8,23	8,26	$6{,}53 \times 10^{4}$	$6{,}49 \times 10^{4}$	10,3	0,119
WT_8b	Sí	0,0100	$9{,}00 \times 10^{-3}$		3,11		40,0		26,8	0,900
WT_8b	No	0,0100	$3{,}00 \times 10^{-3}$		2,50		8,00		29,0	0,983
WT_8b	Sí	0,0500	0,0460		4,76		155		36,5	0,780
WT_8b	No	0,0500	0,0440		4,30		96,0		61,4	0,907
WT_8b	Sí	0,100	0,0460		4,76		155		36,7	0,780
WT_8b	No	0,100	0,0830		4,81		176		46,0	0,898
WT_8b	Sí	0,250	0,227		5,34		488		49,7	0,651
WT_8b	No	0,250	0,177		4,95		384		98,4	0,821
WT_8b	Sí	0,500	0,429		5,85		872		38,7	0,637
WT_8b	No	0,500	0,312		5,31		640		95,2	0,819
int2float	Sí	0,0100	$8{,}00 \times 10^{-3}$		0,0810		69,0		2,91	0,804
int2float	No	0,0100	$7{,}00 \times 10^{-3}$		0,170		41,0		0,830	0,945
int2float	Sí	0,0500	0,0370		0,686		90,0		3,86	0,506
int2float	No	0,0500	0,0310		0,510		70,0		0,872	0,899
int2float	Sí	0,100	0,0940		0,963		135		3,67	0,341
int2float	No	0,100	0,0680		1,10		112		1,02	0,876
int2float	Sí	0,250	0,246		3,71		127		3,14	0,0
int2float	No	0,250	0,245		3,01		147		2,47	0,525
int2float	Sí	0,500	0,246		3,71		127		3,13	0,0
int2float	No	0,500	0,499		3,65		100		6,31	0,101
RCA_4b	Sí	0,0100	0,0		0,0		0,0		1,65	0,985
RCA_4b	No	0,0100	0,0		0,0		0,0		0,228	1,00
RCA_4b	Sí	0,0500	0,0210		0,234		2,00		1,39	0,918
RCA_4b	No	0,0500	0,0210		0,234		2,00		0,230	0,955
RCA_4b	Sí	0,100	0,0850		1,44		2,00		0,909	0,604
RCA_4b	No	0,100	0,0850		1,44		2,00		0,180	0,881
RCA_4b	Sí	0,250	0,142		1,88		5,00		1,27	0,336
RCA_4b	No	0,250	0,136		1,88		4,00		0,265	0,545
RCA_4b	Sí	0,500	0,456		2,50		27,0		0,893	0,0672
RCA_4b	No	0,500	0,288		2,25		8,00		0,267	0,239
dec	Sí	0,0100	0,0		0,0		0,0		3,63	1,00
dec	No	0,0100	0,0		0,0		0,0		0,578	1,00
dec	Sí	0,0500	0,0		0,0		0,0		3,11	1,00
dec	No	0,0500	0,0		0,0		0,0		0,688	1,00
dec	Sí	0,100	0,0		0,0		0,0		3,04	1,00
dec	No	0,100	0,0		0,0		0,0		0,540	1,00
dec	Sí	0,250	0,0		0,0		0,0		2,90	1,00
dec	No	0,250	0,0		0,0		0,0		0,523	1,00
dec	Sí	0,500	0,438		0,656		$8{,}51 \times 10^{38}$		1,87	0,606
dec	No	0,500	0,438		0,656		$8{,}51 \times 10^{38}$		0,441	0,995

Tabla 6.5: Resultados recolectados para el método `outconst`.

Circuito	Resíntesis	Umbral de error	MRED	MRED validación	HD	HD validación	WCE	WCE validación	Tiempo de ejecución (s)	Área (% del área original)
DTree	Sí	0,0100	0,145	0,295	0,145	0,295	0,667	1,33	14,4	0,813
DTree	No	0,0100	0,0	0,0	0,0	0,0	0,0	0,0	19,9	1,00
DTree	Sí	0,0500	0,145	0,295	0,145	0,295	0,667	1,33	14,3	0,813
DTree	No	0,0500	0,0	0,0	0,0	0,0	0,0	0,0	20,0	1,00
DTree	Sí	0,100	0,145	0,295	0,145	0,295	0,667	1,33	14,5	0,813
DTree	No	0,100	0,0	0,0	0,0	0,0	0,0	0,0	19,8	1,00
DTree	Sí	0,250	0,218	0,209	0,218	0,209	1,00	1,00	5,56	0,910
DTree	No	0,250	0,218	0,209	0,218	0,209	1,00	1,00	3,83	0,947
DTree	Sí	0,500	0,446	0,443	0,446	0,443	2,00	2,00	36,1	0,751
DTree	No	0,500	0,446	0,443	0,446	0,443	2,00	2,00	20,1	0,803
barshift_128b	Sí	0,0100	$7{,}00 \times 10^{-3}$	0,0160	59,4	60,4	$6{,}65 \times 10^{35}$	$1{,}33 \times 10^{36}$	405	0,485
barshift_128b	Sí	0,0500	0,0430	0,0940	60,8	61,9	$5{,}32 \times 10^{36}$	$1{,}06 \times 10^{37}$	410	0,342
barshift_128b	Sí	0,100	0,0800	0,149	61,4	62,4	$1{,}06 \times 10^{37}$	$2{,}12 \times 10^{37}$	413	0,282
barshift_128b	Sí	0,250	0,237	0,243	62,3	62,8	$4{,}25 \times 10^{37}$	$4{,}25 \times 10^{37}$	125	0,190
barshift_128b	No	0,250	0,237	0,243	62,3	62,8	$4{,}25 \times 10^{37}$	$4{,}25 \times 10^{37}$	946	0,348
barshift_128b	Sí	0,500	0,400	0,409	62,9	63,3	$8{,}50 \times 10^{37}$	$8{,}48 \times 10^{37}$	120	0,126
barshift_128b	No	0,500	0,400	0,409	62,9	63,3	$8{,}50 \times 10^{37}$	$8{,}48 \times 10^{37}$	$1{,}07 \times 10^{3}$	0,287
sobel	Sí	0,0100	$7{,}33 \times 10^{-3}$	0,0180	1,00	1,49	2,33	5,00	8,72	0,869
sobel	No	0,0100	$2{,}00 \times 10^{-3}$	$2{,}00 \times 10^{-3}$	0,499	0,495	1,00	1,00	7,02	0,999
sobel	Sí	0,0500	0,0260	0,0473	2,01	2,49	7,00	12,3	11,2	0,945
sobel	No	0,0500	0,0260	0,0260	2,01	1,99	7,00	7,00	9,91	0,984
sobel	Sí	0,100	0,0570	0,100	2,77	3,25	15,0	25,7	13,4	0,944
sobel	No	0,100	0,0570	0,0580	2,77	2,75	15,0	15,0	11,9	0,973
sobel	Sí	0,250	0,246	0,247	4,31	4,30	63,0	63,0	9,07	0,828
sobel	No	0,250	0,246	0,247	4,31	4,30	63,0	63,0	5,61	0,959
sobel	Sí	0,500	0,497	0,495	5,12	5,10	127	127	12,2	0,836
sobel	No	0,500	0,497	0,495	5,12	5,10	127	127	30,1	0,950
BK_32b	Sí	0,0100	$5{,}00 \times 10^{-3}$	$9{,}00 \times 10^{-3}$	12,5	12,8	$3{,}36 \times 10^{7}$	$5{,}59 \times 10^{7}$	26,9	0,505
BK_32b	No	0,0100	$5{,}00 \times 10^{-3}$	$5{,}00 \times 10^{-3}$	12,5	12,5	$3{,}36 \times 10^{7}$	$3{,}36 \times 10^{7}$	14,0	0,592
BK_32b	Sí	0,0500	0,0430	0,0730	14,0	14,3	$2{,}68 \times 10^{8}$	$4{,}47 \times 10^{8}$	28,6	0,460
BK_32b	No	0,0500	0,0430	0,0430	14,0	14,0	$2{,}68 \times 10^{8}$	$2{,}68 \times 10^{8}$	13,5	0,541
BK_32b	Sí	0,100	0,0860	0,142	14,5	14,8	$5{,}37 \times 10^{8}$	$8{,}95 \times 10^{8}$	29,7	0,443
BK_32b	No	0,100	0,0860	0,0880	14,5	14,5	$5{,}37 \times 10^{8}$	$5{,}37 \times 10^{8}$	15,2	0,525
BK_32b	Sí	0,250	0,169	0,169	15,0	15,0	$1{,}07 \times 10^{9}$	$1{,}07 \times 10^{9}$	5,94	0,421
BK_32b	No	0,250	0,169	0,169	15,0	15,0	$1{,}07 \times 10^{9}$	$1{,}07 \times 10^{9}$	4,70	0,508
BK_32b	Sí	0,500	0,327	0,324	15,5	15,5	$2{,}15 \times 10^{9}$	$2{,}15 \times 10^{9}$	11,4	0,417
BK_32b	No	0,500	0,327	0,324	15,5	15,5	$2{,}15 \times 10^{9}$	$2{,}15 \times 10^{9}$	4,25	0,500
KS_32b	Sí	0,0100	$5{,}00 \times 10^{-3}$	$9{,}00 \times 10^{-3}$	12,5	12,8	$3{,}36 \times 10^{7}$	$5{,}59 \times 10^{7}$	24,1	0,409
KS_32b	No	0,0100	$5{,}00 \times 10^{-3}$	$5{,}00 \times 10^{-3}$	12,5	12,5	$3{,}36 \times 10^{7}$	$3{,}36 \times 10^{7}$	10,0	0,813
KS_32b	Sí	0,0500	0,0440	0,0720	14,0	14,4	$2{,}68 \times 10^{8}$	$4{,}47 \times 10^{8}$	25,6	0,368
KS_32b	No	0,0500	0,0440	0,0440	14,0	14,0	$2{,}68 \times 10^{8}$	$2{,}68 \times 10^{8}$	8,18	0,792
KS_32b	Sí	0,100	0,0860	0,140	14,5	14,9	$5{,}37 \times 10^{8}$	$8{,}95 \times 10^{8}$	25,6	0,363
KS_32b	No	0,100	0,0860	0,0860	14,5	14,5	$5{,}37 \times 10^{8}$	$5{,}37 \times 10^{8}$	7,57	0,785
KS_32b	Sí	0,250	0,169	0,167	15,0	15,0	$1{,}07 \times 10^{9}$	$1{,}07 \times 10^{9}$	11,2	0,531
KS_32b	No	0,250	0,169	0,167	15,0	15,0	$1{,}07 \times 10^{9}$	$1{,}07 \times 10^{9}$	11,5	0,778
KS_32b	Sí	0,500	0,327	0,325	15,5	15,5	$2{,}15 \times 10^{9}$	$2{,}15 \times 10^{9}$	9,47	0,521
KS_32b	No	0,500	0,327	0,325	15,5	15,5	$2{,}15 \times 10^{9}$	$2{,}15 \times 10^{9}$	10,9	0,772
CLA_16b	Sí	0,0100	$5{,}00 \times 10^{-3}$	$9{,}00 \times 10^{-3}$	4,49	4,85	511	852	9,01	0,643
CLA_16b	No	0,0100	$5{,}00 \times 10^{-3}$	$5{,}00 \times 10^{-3}$	4,49	4,51	511	511	4,86	0,816
CLA_16b	Sí	0,0500	0,0420	0,0700	5,99	6,34	$4{,}10 \times 10^{3}$	$6{,}83 \times 10^{3}$	10,6	0,506
CLA_16b	No	0,0500	0,0420	0,0420	5,99	6,01	$4{,}10 \times 10^{3}$	$4{,}10 \times 10^{3}$	6,44	0,704
CLA_16b	Sí	0,100	0,0840	0,137	6,49	6,84	$8{,}19 \times 10^{3}$	$1{,}37 \times 10^{4}$	12,3	0,466
CLA_16b	No	0,100	0,0840	0,0840	6,49	6,51	$8{,}19 \times 10^{3}$	$8{,}19 \times 10^{3}$	5,09	0,671
CLA_16b	Sí	0,250	0,167	0,163	6,99	7,00	$1{,}64 \times 10^{4}$	$1{,}64 \times 10^{4}$	6,45	0,439
CLA_16b	No	0,250	0,167	0,163	6,99	7,00	$1{,}64 \times 10^{4}$	$1{,}64 \times 10^{4}$	6,67	0,640
CLA_16b	Sí	0,500	0,323	0,319	7,49	7,50	$3{,}28 \times 10^{4}$	$3{,}28 \times 10^{4}$	7,93	0,449
CLA_16b	No	0,500	0,323	0,319	7,49	7,50	$3{,}28 \times 10^{4}$	$3{,}28 \times 10^{4}$	6,12	0,623
BK_16b	Sí	0,0100	$5{,}00 \times 10^{-3}$	$6{,}00 \times 10^{-3}$	4,50	4,49	511	511	10,1	0,679
BK_16b	No	0,0100	$5{,}00 \times 10^{-3}$	$6{,}00 \times 10^{-3}$	4,50	4,49	511	511	6,77	0,710
BK_16b	Sí	0,0500	0,0430	0,0440	6,01	6,00	$4{,}10 \times 10^{3}$	$4{,}09 \times 10^{3}$	8,75	0,556
BK_16b	No	0,0500	0,0430	0,0440	6,01	6,00	$4{,}10 \times 10^{3}$	$4{,}09 \times 10^{3}$	5,93	0,607
BK_16b	Sí	0,100	0,0860	0,0870	6,51	6,50	$8{,}19 \times 10^{3}$	$8{,}19 \times 10^{3}$	10,9	0,494
BK_16b	No	0,100	0,0860	0,0870	6,51	6,50	$8{,}19 \times 10^{3}$	$8{,}19 \times 10^{3}$	7,46	0,574
BK_16b	Sí	0,250	0,168	0,170	7,00	7,00	$1{,}64 \times 10^{4}$	$1{,}64 \times 10^{4}$	8,29	0,502
BK_16b	No	0,250	0,168	0,170	7,00	7,00	$1{,}64 \times 10^{4}$	$1{,}64 \times 10^{4}$	6,19	0,539
BK_16b	Sí	0,500	0,326	0,331	7,50	7,50	$3{,}28 \times 10^{4}$	$3{,}28 \times 10^{4}$	9,90	0,442
BK_16b	No	0,500	0,326	0,331	7,50	7,50	$3{,}28 \times 10^{4}$	$3{,}28 \times 10^{4}$	7,51	0,506
CSkipA_16b	Sí	0,0100	$5{,}00 \times 10^{-3}$	$5{,}00 \times 10^{-3}$	4,50	4,51	511	511	9,97	0,663
CSkipA_16b	No	0,0100	$5{,}00 \times 10^{-3}$	$5{,}00 \times 10^{-3}$	4,50	4,51	511	511	6,79	0,710
CSkipA_16b	Sí	0,0500	0,0430	0,0430	6,00	6,01	$4{,}10 \times 10^{3}$	$4{,}10 \times 10^{3}$	8,72	0,535

Circuito	Resíntesis	Umbral de error	MRED	MRED validación	HD	HD validación	WCE	WCE validación	Tiempo de ejecución (s)	Área (% del área original)
CSkipA_16b	No	0,0500	0,0430	0,0430	6,00	6,01	$4{,}10 \times 10^{3}$	$4{,}10 \times 10^{3}$	5,95	0,607
CSkipA_16b	Sí	0,100	0,0860	0,0850	6,50	6,51	$8{,}19 \times 10^{3}$	$8{,}19 \times 10^{3}$	10,9	0,527
CSkipA_16b	No	0,100	0,0860	0,0850	6,50	6,51	$8{,}19 \times 10^{3}$	$8{,}19 \times 10^{3}$	7,51	0,574
CSkipA_16b	Sí	0,250	0,170	0,168	7,00	7,01	$1{,}64 \times 10^{4}$	$1{,}64 \times 10^{4}$	9,18	0,492
CSkipA_16b	No	0,250	0,170	0,168	7,00	7,01	$1{,}64 \times 10^{4}$	$1{,}64 \times 10^{4}$	6,28	0,539
CSkipA_16b	Sí	0,500	0,326	0,326	7,50	7,51	$3{,}28 \times 10^{4}$	$3{,}28 \times 10^{4}$	$1{,}0 \times 10^{1}$	0,430
CSkipA_16b	No	0,500	0,326	0,326	7,50	7,51	$3{,}28 \times 10^{4}$	$3{,}28 \times 10^{4}$	7,66	0,506
KS_16b	Sí	0,0100	$5{,}00 \times 10^{-3}$	$6{,}00 \times 10^{-3}$	4,50	4,50	511	511	5,73	0,504
KS_16b	No	0,0100	$5{,}00 \times 10^{-3}$	$6{,}00 \times 10^{-3}$	4,50	4,50	511	511	6,70	0,855
KS_16b	Sí	0,0500	0,0430	0,0440	5,99	6,01	$4{,}10 \times 10^{3}$	$4{,}10 \times 10^{3}$	9,11	0,434
KS_16b	No	0,0500	0,0430	0,0440	5,99	6,01	$4{,}10 \times 10^{3}$	$4{,}10 \times 10^{3}$	6,80	0,797
KS_16b	Sí	0,100	0,0860	0,0870	6,49	6,52	$8{,}19 \times 10^{3}$	$8{,}19 \times 10^{3}$	6,90	0,391
KS_16b	No	0,100	0,0860	0,0870	6,49	6,52	$8{,}19 \times 10^{3}$	$8{,}19 \times 10^{3}$	5,52	0,778
KS_16b	Sí	0,250	0,168	0,172	6,99	7,02	$1{,}64 \times 10^{4}$	$1{,}64 \times 10^{4}$	9,26	0,354
KS_16b	No	0,250	0,168	0,172	6,99	7,02	$1{,}64 \times 10^{4}$	$1{,}64 \times 10^{4}$	7,60	0,759
KS_16b	Sí	0,500	0,328	0,329	7,49	7,51	$3{,}28 \times 10^{4}$	$3{,}28 \times 10^{4}$	7,31	0,308
KS_16b	No	0,500	0,328	0,329	7,49	7,51	$3{,}28 \times 10^{4}$	$3{,}28 \times 10^{4}$	6,45	0,739
LFA_16b	Sí	0,0100	$5{,}00 \times 10^{-3}$	$5{,}00 \times 10^{-3}$	4,49	4,51	511	511	10,3	0,671
LFA_16b	No	0,0100	$5{,}00 \times 10^{-3}$	$5{,}00 \times 10^{-3}$	4,49	4,51	511	511	6,85	0,710
LFA_16b	Sí	0,0500	0,0430	0,0420	5,99	6,01	$4{,}10 \times 10^{3}$	$4{,}10 \times 10^{3}$	8,94	0,566
LFA_16b	No	0,0500	0,0430	0,0420	5,99	6,01	$4{,}10 \times 10^{3}$	$4{,}10 \times 10^{3}$	5,91	0,607
LFA_16b	Sí	0,100	0,0860	0,0850	6,49	6,51	$8{,}19 \times 10^{3}$	$8{,}19 \times 10^{3}$	11,1	0,500
LFA_16b	No	0,100	0,0860	0,0850	6,49	6,51	$8{,}19 \times 10^{3}$	$8{,}19 \times 10^{3}$	7,46	0,574
LFA_16b	Sí	0,250	0,168	0,166	6,99	7,01	$1{,}64 \times 10^{4}$	$1{,}64 \times 10^{4}$	9,45	0,484
LFA_16b	No	0,250	0,168	0,166	6,99	7,01	$1{,}64 \times 10^{4}$	$1{,}64 \times 10^{4}$	6,36	0,539
LFA_16b	Sí	0,500	0,328	0,325	7,50	7,51	$3{,}28 \times 10^{4}$	$3{,}28 \times 10^{4}$	10,1	0,414
LFA_16b	No	0,500	0,328	0,325	7,50	7,51	$3{,}28 \times 10^{4}$	$3{,}28 \times 10^{4}$	7,73	0,506
WT_8b	Sí	0,0100	$7{,}00 \times 10^{-3}$		1,77		30,0		50,0	0,953
WT_8b	No	0,0100	$8{,}00 \times 10^{-3}$		2,02		31,0		38,5	0,968
WT_8b	Sí	0,0500	0,0280		3,00		127		25,2	0,936
WT_8b	No	0,0500	0,0280		3,00		127		42,2	0,953
WT_8b	Sí	0,100	0,0850		3,99		511		37,3	0,929
WT_8b	No	0,100	0,0850		3,99		511		51,1	0,939
WT_8b	Sí	0,250	0,221		4,95		$2{,}05 \times 10^{3}$		21,6	0,913
WT_8b	No	0,250	0,221		4,95		$2{,}05 \times 10^{3}$		19,5	0,930
WT_8b	Sí	0,500	0,495		5,83		$8{,}19 \times 10^{3}$		50,4	0,914
WT_8b	No	0,500	0,495		5,83		$8{,}19 \times 10^{3}$		116	0,917
int2float	Sí	0,0100	0,0		0,0		0,0		3,04	0,985
int2float	No	0,0100	0,0		0,0		0,0		0,695	1,00
int2float	Sí	0,0500	0,0500		2,79		11,0		3,72	0,656
int2float	No	0,0500	0,0500		2,79		11,0		1,63	0,730
int2float	Sí	0,100	0,0500		2,79		11,0		3,32	0,656
int2float	No	0,100	0,0500		2,79		11,0		1,55	0,716
int2float	Sí	0,250	0,146		3,86		33,0		3,32	0,361
int2float	No	0,250	0,146		3,86		33,0		2,68	0,415
int2float	Sí	0,500	0,331		3,82		77,0		2,32	0,0652
int2float	No	0,500	0,331		3,82		77,0		5,29	0,0652
RCA_4b	Sí	0,0100	0,0		0,0		0,0		1,65	0,985
RCA_4b	No	0,0100	0,0		0,0		0,0		0,236	1,00
RCA_4b	Sí	0,0500	0,0470		0,500		1,00		0,902	0,888
RCA_4b	No	0,0500	0,0470		0,500		1,00		0,177	0,873
RCA_4b	Sí	0,100	0,0470		0,500		1,00		0,904	0,888
RCA_4b	No	0,100	0,0470		0,500		1,00		0,195	0,873
RCA_4b	Sí	0,250	0,136		1,00		3,00		1,77	0,672
RCA_4b	No	0,250	0,136		1,00		3,00		0,251	0,739
RCA_4b	Sí	0,500	0,302		1,50		7,00		1,06	0,478
RCA_4b	No	0,500	0,302		1,50		7,00		0,188	0,612
dec	Sí	0,0100	$8{,}00 \times 10^{-3}$		0,0120		10,0		2,36	0,994
dec	No	0,0100	$8{,}00 \times 10^{-3}$		0,0120		10,0		0,422	0,994
dec	Sí	0,0500	0,0470		0,0700		320		3,46	0,962
dec	No	0,0500	0,0470		0,0700		320		0,758	0,962
dec	Sí	0,100	0,0980		0,148		$4{,}10 \times 10^{4}$		6,31	0,924
dec	No	0,100	0,0980		0,148		$4{,}10 \times 10^{4}$		1,70	0,921
dec	Sí	0,250	0,246		0,371		$2{,}15 \times 10^{10}$		10,2	0,790
dec	No	0,250	0,246		0,371		$2{,}15 \times 10^{10}$		5,47	0,788
dec	Sí	0,500	0,496		0,746		$9{,}22 \times 10^{19}$		15,8	0,571
dec	No	0,500	0,496		0,746		$9{,}22 \times 10^{19}$		19,0	0,564

Tabla 6.6: Resultados recolectados para el método `probprun`.

Circuito	Resíntesis	Umbral de error	MRED	MRED validación	HD	HD validación	WCE	WCE validación	Tiempo de ejecución (s)	Área (% del área original)
DTree	Sí	0,0100	0,0100	0,0110	0,0100	0,0110	16,0	16,0	114	0,839
DTree	No	0,0100	0,0100	0,0110	0,0100	0,0110	16,0	16,0	7,11	0,987
DTree	Sí	0,0500	0,0100	0,0110	0,0100	0,0110	16,0	16,0	113	0,839
DTree	No	0,0500	0,0470	0,0550	0,0450	0,0480	16,0	16,0	9,88	0,936
DTree	Sí	0,100	0,0100	0,0110	0,0100	0,0110	16,0	16,0	112	0,839
DTree	No	0,100	0,0880	0,0950	0,0870	0,0890	16,0	16,0	9,81	0,888
DTree	Sí	0,250	0,0450	0,0540	0,0430	0,0460	16,0	16,0	23,8	0,654
DTree	No	0,250	0,244	0,237	0,244	0,231	16,0	16,0	8,23	0,825
DTree	Sí	0,500	0,0450	0,0540	0,0430	0,0460	16,0	16,0	23,8	0,654
DTree	No	0,500	0,436	0,414	0,468	0,447	16,0	16,0	7,88	0,689
barshift_128b	Sí	0,0100	$6{,}00 \times 10^{-3}$	$5{,}00 \times 10^{-3}$	0,215	0,226	$4{,}25 \times 10^{37}$	$4{,}25 \times 10^{37}$	16,0	0,999
barshift_128b	No	0,0100	$6{,}00 \times 10^{-3}$	$5{,}00 \times 10^{-3}$	0,215	0,226	$4{,}25 \times 10^{37}$	$4{,}25 \times 10^{37}$	6,36	0,999
barshift_128b	Sí	0,0500	0,0250	0,0310	0,538	0,592	$9{,}57 \times 10^{37}$	$9{,}57 \times 10^{37}$	897	0,997
barshift_128b	No	0,0500	0,0470	0,0620	2,52	2,83	$9{,}57 \times 10^{37}$	$9{,}57 \times 10^{37}$	9,79	0,991
barshift_128b	Sí	0,100	0,0250	0,0310	0,538	0,592	$9{,}57 \times 10^{37}$	$9{,}57 \times 10^{37}$	903	0,997
barshift_128b	No	0,100	0,0980	0,120	3,51	3,92	$2{,}23 \times 10^{38}$	$2{,}23 \times 10^{38}$	15,6	0,988
barshift_128b	Sí	0,250	0,0500	0,0590	2,74	3,05	$2{,}03 \times 10^{38}$	$2{,}02 \times 10^{38}$	205	0,985
barshift_128b	No	0,250	0,248	0,283	17,7	19,5	$3{,}33 \times 10^{38}$	$3{,}30 \times 10^{38}$	30,5	0,936
barshift_128b	Sí	0,500	0,0500	0,0590	2,74	3,05	$2{,}03 \times 10^{38}$	$2{,}02 \times 10^{38}$	204	0,985
barshift_128b	No	0,500	0,487	0,532	37,6	40,0	$3{,}39 \times 10^{38}$	$3{,}33 \times 10^{38}$	86,5	0,854
sobel	Sí	0,0100	$3{,}00 \times 10^{-3}$	$5{,}00 \times 10^{-3}$	0,0170	0,0300	128	128	223	0,849
sobel	No	0,0100	$7{,}00 \times 10^{-3}$	$7{,}00 \times 10^{-3}$	0,205	0,227	32,0	32,0	8,27	0,981
sobel	Sí	0,0500	$3{,}00 \times 10^{-3}$	$5{,}00 \times 10^{-3}$	0,0170	0,0300	128	128	221	0,849
sobel	No	0,0500	0,0490	0,0660	0,440	0,495	141	135	7,03	0,980
sobel	Sí	0,100	$3{,}00 \times 10^{-3}$	$5{,}00 \times 10^{-3}$	0,0170	0,0300	128	128	222	0,849
sobel	No	0,100	0,0770	0,0860	0,767	0,791	182	135	9,71	0,972
sobel	Sí	0,250	0,0780	0,0870	0,829	0,859	182	139	13,6	0,943
sobel	No	0,250	0,0780	0,0870	0,829	0,859	182	139	7,39	0,967
sobel	Sí	0,500	0,291	0,287	1,13	1,12	196	194	46,5	0,913
sobel	No	0,500	0,475	0,488	2,54	2,54	232	232	24,4	0,0363
BK_32b	Sí	0,0100	0,0	0,0	0,0	0,0	0	0	8,99	0,978
BK_32b	No	0,0100	0,0	0,0	0,0	0,0	0	0	6,62	1,00
BK_32b	Sí	0,0500	0,0	0,0	0,0	0,0	0	0	8,77	0,978
BK_32b	No	0,0500	0,0	0,0	0,0	0,0	0	0	6,31	1,00
BK_32b	Sí	0,100	0,0940	0,0920	0,985	1,00	$1{,}34 \times 10^{9}$	$1{,}34 \times 10^{9}$	41,4	0,954
BK_32b	No	0,100	0,0960	0,0960	8,09	8,10	$1{,}38 \times 10^{9}$	$1{,}38 \times 10^{9}$	9,74	0,850
BK_32b	Sí	0,250	0,101	0,100	5,79	5,82	$1{,}65 \times 10^{9}$	$1{,}65 \times 10^{9}$	11,8	0,809
BK_32b	No	0,250	0,244	0,251	9,37	9,43	$5{,}93 \times 10^{9}$	$5{,}92 \times 10^{9}$	12,6	0,827
BK_32b	Sí	0,500	0,101	0,100	5,79	5,82	$1{,}65 \times 10^{9}$	$1{,}65 \times 10^{9}$	11,4	0,809
BK_32b	No	0,500	0,444	0,436	14,9	14,8	$5{,}73 \times 10^{9}$	$5{,}72 \times 10^{9}$	15,9	0,663
KS_32b	Sí	0,0100	0,0	0,0	0,0	0,0	0	0	8,77	0,959
KS_32b	No	0,0100	0,0	0,0	0,0	0,0	0	0	6,15	1,00
KS_32b	Sí	0,0500	0,0200	0,0190	0,833	0,828	$2{,}89 \times 10^{9}$	$2{,}89 \times 10^{9}$	52,4	0,91
KS_32b	No	0,0500	0,0200	0,0190	1,33	1,32	$2{,}89 \times 10^{9}$	$2{,}89 \times 10^{9}$	4,27	0,976
KS_32b	Sí	0,100	0,0200	0,0190	0,833	0,828	$2{,}89 \times 10^{9}$	$2{,}89 \times 10^{9}$	52,6	0,91
KS_32b	No	0,100	0,0860	0,0760	1,63	1,60	$2{,}89 \times 10^{9}$	$2{,}89 \times 10^{9}$	6,99	0,966
KS_32b	Sí	0,250	0,141	0,137	5,46	5,43	$3{,}15 \times 10^{9}$	$3{,}13 \times 10^{9}$	11,7	0,814
KS_32b	No	0,250	0,205	0,205	15,8	15,8	$4{,}11 \times 10^{9}$	$4{,}09 \times 10^{9}$	9,88	0,525
KS_32b	Sí	0,500	0,141	0,137	5,46	5,43	$3{,}15 \times 10^{9}$	$3{,}13 \times 10^{9}$	11,6	0,814
KS_32b	No	0,500	0,412	0,424	16,1	16,1	$8{,}40 \times 10^{9}$	$8{,}38 \times 10^{9}$	8,41	0,507
CLA_16b	Sí	0,0100	$1{,}00 \times 10^{-3}$	$1{,}00 \times 10^{-3}$	0,245	0,251	256	256	6,26	0,985
CLA_16b	No	0,0100	$1{,}00 \times 10^{-3}$	$1{,}00 \times 10^{-3}$	0,245	0,251	256	256	4,83	0,993
CLA_16b	Sí	0,0500	$1{,}00 \times 10^{-3}$	$1{,}00 \times 10^{-3}$	0,245	0,251	256	256	6,63	0,985
CLA_16b	No	0,0500	$1{,}00 \times 10^{-3}$	$1{,}00 \times 10^{-3}$	0,245	0,251	256	256	4,91	0,993
CLA_16b	Sí	0,100	$1{,}00 \times 10^{-3}$	$1{,}00 \times 10^{-3}$	0,245	0,251	256	256	6,31	0,985
CLA_16b	No	0,100	$1{,}00 \times 10^{-3}$	$1{,}00 \times 10^{-3}$	0,245	0,251	256	256	4,90	0,993
CLA_16b	Sí	0,250	0,200	0,193	0,906	0,919	$6{,}96 \times 10^{4}$	$6{,}96 \times 10^{4}$	18,3	0,961
CLA_16b	No	0,250	0,208	0,205	7,07	7,12	$5{,}57 \times 10^{4}$	$5{,}66 \times 10^{4}$	7,46	0,735
CLA_16b	Sí	0,500	0,200	0,193	0,906	0,919	$6{,}96 \times 10^{4}$	$6{,}96 \times 10^{4}$	18,1	0,961
CLA_16b	No	0,500	0,392	0,394	8,34	8,43	$1{,}21 \times 10^{5}$	$1{,}22 \times 10^{5}$	6,30	0,717
BK_16b	Sí	0,0100	0,0	0,0	0,0	0,0	0	0	7,71	1,29
BK_16b	No	0,0100	0,0	0,0	0,0	0,0	0	0	5,77	1,00
BK_16b	Sí	0,0500	0,0	0,0	0,0	0,0	0	0	7,64	1,29
BK_16b	No	0,0500	0,0	0,0	0,0	0,0	0	0	5,56	1,00
BK_16b	Sí	0,100	0,0870	0,0870	0,543	0,539	$1{,}64 \times 10^{4}$	$1{,}64 \times 10^{4}$	3,95	0,951
BK_16b	No	0,100	0,0870	0,0870	0,543	0,539	$1{,}64 \times 10^{4}$	$1{,}64 \times 10^{4}$	3,12	0,973
BK_16b	Sí	0,250	0,141	0,147	1,22	1,23	$2{,}46 \times 10^{4}$	$2{,}46 \times 10^{4}$	16,7	0,911
BK_16b	No	0,250	0,216	0,218	4,01	4,08	$8{,}23 \times 10^{4}$	$8{,}22 \times 10^{4}$	6,16	0,844
BK_16b	Sí	0,500	0,141	0,147	1,22	1,23	$2{,}46 \times 10^{4}$	$2{,}46 \times 10^{4}$	16,6	0,911
BK_16b	No	0,500	0,466	0,480	4,16	4,24	$9{,}14 \times 10^{4}$	$9{,}14 \times 10^{4}$	5,34	0,837
CSkipA_16b	Sí	0,0100	0,0	0,0	0,0	0,0	0	0	6,91	1,00

Circuito	Resíntesis	Umbral de error	MRED	MRED validación	HD	HD validación	WCE	WCE validación	Tiempo de ejecución (s)	Área (% del área original)
CSkipA_16b	No	0,0100	0,0	0,0	0,0	0,0	0	0	5,71	1,00
CSkipA_16b	Sí	0,0500	0,0	0,0	0,0	0,0	0	0	7,39	1,00
CSkipA_16b	No	0,0500	0,0	0,0	0,0	0,0	0	0	5,66	1,00
CSkipA_16b	Sí	0,100	0,0930	0,0930	0,123	0,123	$6{,}55 \times 10^4$	$6{,}55 \times 10^4$	6,41	1,00
CSkipA_16b	No	0,100	0,0930	0,0930	0,123	0,123	$6{,}55 \times 10^4$	$6{,}55 \times 10^4$	5,01	0,992
CSkipA_16b	Sí	0,250	0,107	0,108	1,16	1,16	$6{,}99 \times 10^4$	$6{,}99 \times 10^4$	16,6	0,936
CSkipA_16b	No	0,250	0,227	0,230	3,19	3,25	$1{,}03 \times 10^5$	$1{,}03 \times 10^5$	4,69	0,879
CSkipA_16b	Sí	0,500	0,107	0,108	1,16	1,16	$6{,}99 \times 10^4$	$6{,}99 \times 10^4$	16,6	0,936
CSkipA_16b	No	0,500	0,497	0,499	7,98	7,96	$1{,}12 \times 10^5$	$1{,}11 \times 10^5$	7,45	0,442
KS_16b	Sí	0,0100	$7{,}00 \times 10^{-3}$	$7{,}00 \times 10^{-3}$	0,169	0,173	$4{,}40 \times 10^4$	$4{,}40 \times 10^4$	5,75	0,824
KS_16b	No	0,0100	$7{,}00 \times 10^{-3}$	$7{,}00 \times 10^{-3}$	0,169	0,173	$4{,}40 \times 10^4$	$4{,}40 \times 10^4$	4,44	0,991
KS_16b	Sí	0,0500	0,0480	0,0470	0,730	0,719	$5{,}02 \times 10^4$	$5{,}02 \times 10^4$	20,8	0,772
KS_16b	No	0,0500	0,0480	0,0480	0,981	0,967	$5{,}17 \times 10^4$	$5{,}17 \times 10^4$	4,39	0,939
KS_16b	Sí	0,100	0,0480	0,0470	0,730	0,719	$5{,}02 \times 10^4$	$5{,}02 \times 10^4$	20,4	0,772
KS_16b	No	0,100	0,0480	0,0480	0,981	0,967	$5{,}17 \times 10^4$	$5{,}17 \times 10^4$	4,28	0,939
KS_16b	Sí	0,250	0,0480	0,0470	0,730	0,719	$5{,}02 \times 10^4$	$5{,}02 \times 10^4$	20,8	0,772
KS_16b	No	0,250	0,242	0,239	7,62	7,57	$6{,}68 \times 10^4$	$6{,}66 \times 10^4$	8,38	0,581
KS_16b	Sí	0,500	0,0480	0,0470	0,730	0,719	$5{,}02 \times 10^4$	$5{,}02 \times 10^4$	20,7	0,772
KS_16b	No	0,500	0,420	0,415	8,22	8,18	$1{,}30 \times 10^5$	$1{,}28 \times 10^5$	8,66	0,536
LFA_16b	Sí	0,0100	0,0	0,0	0,0	0,0	0	0	7,79	1,00
LFA_16b	No	0,0100	0,0	0,0	0,0	0,0	0	0	5,52	1,00
LFA_16b	Sí	0,0500	0,0	0,0	0,0	0,0	0	0	7,73	1,00
LFA_16b	No	0,0500	0,0	0,0	0,0	0,0	0	0	5,70	1,00
LFA_16b	Sí	0,100	0,0950	0,0950	0,126	0,125	$6{,}55 \times 10^4$	$6{,}55 \times 10^4$	6,89	0,988
LFA_16b	No	0,100	0,0950	0,0950	0,126	0,125	$6{,}55 \times 10^4$	$6{,}55 \times 10^4$	4,94	0,992
LFA_16b	Sí	0,250	0,0950	0,0950	0,126	0,125	$6{,}55 \times 10^4$	$6{,}55 \times 10^4$	6,89	0,988
LFA_16b	No	0,250	0,0950	0,0950	0,126	0,125	$6{,}55 \times 10^4$	$6{,}55 \times 10^4$	4,96	0,992
LFA_16b	Sí	0,500	0,494	0,487	1,12	1,13	$8{,}19 \times 10^4$	$8{,}19 \times 10^4$	18,7	0,918
LFA_16b	No	0,500	0,499	0,495	4,65	4,70	$8{,}83 \times 10^4$	$8{,}83 \times 10^4$	5,35	0,829
WT_8b	Sí	0,0100	$7{,}00 \times 10^{-3}$		0,0700		$1{,}84 \times 10^4$		13,4	0,953
WT_8b	No	0,0100	$7{,}00 \times 10^{-3}$		0,0700		$1{,}84 \times 10^4$		12,0	0,992
WT_8b	Sí	0,0500	0,0210		0,0860		$3{,}53 \times 10^4$		356	0,944
WT_8b	No	0,0500	0,0300		0,325		$4{,}22 \times 10^4$		25,3	0,968
WT_8b	Sí	0,100	0,0210		0,0860		$3{,}53 \times 10^4$		355	0,944
WT_8b	No	0,100	0,0880		0,997		$4{,}56 \times 10^4$		43,0	0,917
WT_8b	Sí	0,250	0,0940		0,881		$3{,}96 \times 10^4$		70,5	0,818
WT_8b	No	0,250	0,212		3,53		$5{,}19 \times 10^4$		45,0	0,701
WT_8b	Sí	0,500	0,0940		0,881		$3{,}96 \times 10^4$		70,2	0,818
WT_8b	No	0,500	0,477		5,09		$4{,}90 \times 10^4$		54,6	0,522
int2float	Sí	0,0100	0,0		0,0		0		2,93	0,985
int2float	No	0,0100	0,0		0,0		0		0,627	1,00
int2float	Sí	0,0500	0,0410		0,0260		80,0		10,8	0,839
int2float	No	0,0500	0,0490		0,229		84,0		0,943	0,835
int2float	Sí	0,100	0,0410		0,0260		80,0		10,8	0,839
int2float	No	0,100	0,0940		0,942		84,0		1,27	0,684
int2float	Sí	0,250	0,0410		0,0260		80,0		10,7	0,839
int2float	No	0,250	0,219		3,53		117		1,70	0,209
int2float	Sí	0,500	0,0410		0,0260		80,0		10,7	0,839
int2float	No	0,500	0,394		3,77		157		1,82	0,160
RCA_4b	Sí	0,0100	0,0		0,0		0		1,58	0,985
RCA_4b	No	0,0100	0,0		0,0		0		0,226	1,00
RCA_4b	Sí	0,0500	0,0200		0,484		1,00		1,09	0,843
RCA_4b	No	0,0500	0,0200		0,484		1,00		0,194	0,925
RCA_4b	Sí	0,100	0,0650		0,758		5,00		0,851	0,828
RCA_4b	No	0,100	0,0650		0,758		5,00		0,173	0,896
RCA_4b	Sí	0,250	0,179		0,910		21,0		0,595	0,784
RCA_4b	No	0,250	0,179		0,910		21,0		0,158	0,866
RCA_4b	Sí	0,500	0,291		1,23		29,0		1,06	0,761
RCA_4b	No	0,500	0,370		1,97		29,0		0,187	0,701
dec	Sí	0,0100	$8{,}00 \times 10^{-3}$		$8{,}00 \times 10^{-3}$		$2{,}08 \times 10^{34}$		2,31	0,994
dec	No	0,0100	$8{,}00 \times 10^{-3}$		$8{,}00 \times 10^{-3}$		$2{,}08 \times 10^{34}$		0,401	0,994
dec	Sí	0,0500	0,0200		0,0200		$1{,}66 \times 10^{35}$		26,5	0,986
dec	No	0,0500	0,0470		0,0470		$8{,}51 \times 10^{37}$		0,541	0,962
dec	Sí	0,100	0,0200		0,0200		$1{,}66 \times 10^{35}$		26,5	0,986
dec	No	0,100	0,0980		0,145		$1{,}70 \times 10^{38}$		0,942	0,921
dec	Sí	0,250	0,0200		0,0200		$1{,}66 \times 10^{35}$		26,4	0,986
dec	No	0,250	0,246		0,441		$1{,}70 \times 10^{38}$		1,42	0,802
dec	Sí	0,500	0,0200		0,0200		$1{,}66 \times 10^{35}$		26,4	0,986
dec	No	0,500	0,496		0,914		$1{,}70 \times 10^{39}$		3,36	0,600