Aplicación Multiplataforma para el Aprendizaje y Depuración de Código HDL sin Requerimiento de Hardware Físico

Trabajo Terminal No. TT 2026-B059

Alumno: *Ulrich Tamayo Daniel**

Directores: Pastrana Fernández Carlos Jesús, Cifuentes Alvarez Alejandro Sigrido

E-mail: dulricht1700@alumno.ipn.mx

Resumen - El presente trabajo terminal desarrollará una aplicación multiplataforma de escritorio, ejecutada en un entorno local, orientada a la depuración de código en HDL (Hardware Description Language). Su objetivo es facilitar el análisis de programas mediante la simulación y la visualización interactiva de señales digitales, ciclos de reloj y estados de puertos. La propuesta busca apoyar el aprendizaje académico de diseño digital en FPGAs (Field Programmable Gate Arrays), ofreciendo una herramienta gratuita y open-source que fortalezca las prácticas educativas y permita comprender de manera práctica la lógica de los sistemas digitales sin requerir hardware físico.

Palabras clave - Academia de Ingeniería de Software, Departamento de Sistemas Electrónicos, Educación en Lenguaje descriptivo de Hardware, Simulación Digital

1. Introducción

La enseñanza del diseño de sistemas digitales y su implementación en dispositivos lógicos programables, como las Field Programmable Gate Arrays (FPGAs), enfrenta barreras significativas que limitan la formación práctica de los estudiantes. Aunque el dominio de lenguajes de descripción de hardware (HDLs) como VHDL (VHSIC Hardware Description Language) o Verilog es esencial, la comprensión profunda de la lógica subyacente exige una experiencia de depuración más allá de la simple escritura de código.

Hemos observado que las herramientas educativas actuales presentan limitaciones en dos dimensiones principales: económicas y pedagógicas. Por un lado, la dependencia del hardware físico constituye un obstáculo financiero relevante para estudiantes e instituciones. Una tarjeta FPGA Altera Cyclone IV puede costar alrededor de $1,866.00 MXN, mientras que la popular Basys 3 alcanza $165.00 USD, sin contar accesorios [1], [2]. Estos costos restringen la práctica accesible. A ello se suma el precio de cursos especializados, que en algunos casos supera los $5,000 USD [3]. Por otro lado, existe un vacío pedagógico en la oferta de plataformas virtuales. Aunque algunos fabricantes, como Intel y AMD/Xilinx, ofrecen entornos de diseño (Quartus Prime, Vivado), sus versiones gratuitas poseen limitaciones técnicas importantes [4], [5].

En cuanto a herramientas de código abierto, GHDL es un simulador potente, pero su uso está limitado a la línea de comandos y depende de la integración con GTKWave para visualizar señales, lo que genera un flujo

fragmentado y poco intuitivo para principiantes [6], [7]. En contraste, soluciones comerciales como ModelSim o QuestaSim ofrecen entornos robustos, pero sus licencias educativas restringidas impiden un uso completo en entornos académicos [8].

En la revisión de Trabajos Terminales (TTs) en la ESCOM, no hemos encontrado propuestas previas dedicadas a la creación de un depurador interactivo de HDL. Sin embargo, existen antecedentes de proyectos relacionados con simulación, control lógico y sistemas educativos interactivos [9], lo que evidencia una línea institucional que puede ser ampliada con la presente propuesta.

La solución planteada busca desarrollar una plataforma académica multiplataforma, ejecutada como servicio web local multiplataforma, gratuita y open-source, enfocada en la depuración lógica y funcional de HDL. A diferencia de los simuladores tradicionales, se propone una experiencia integrada con interfaz gráfica intuitiva, depuración en tiempo real y sin necesidad de hardware físico, lo que la convierte en una herramienta estratégica para fortalecer la enseñanza de diseño digital en la ESCOM.

Software / Proyecto	Características	Precio en el mercado
GHDL + GTKWave	Simulador de VHDL por línea de comandos y visualizador de ondas post-mortem; flujo de trabajo fragmentado y no interactivo.	Gratuito, Open-source
Verilator	Compilador de Verilog/SystemVerilog a C++/SystemC; simulación de alta velocidad; limitado a diseños sintetizables.	Gratuito, Open-source
ModelSim	Simulador comercial robusto de VHDL/Verilog; depuración avanzada, cobertura de código, UVM; interfaz gráfica profesional.	Licencia de pago (Starter gratuita con limitaciones)
Quartus Prime Lite	Entorno de Intel para diseño y simulación de FPGAs; versión gratuita con funciones recortadas, sin análisis avanzado.	Gratuito (limitado) / Licencia de pago
SPICE (Berkeley)	Primer simulador de circuitos analógicos; análisis de punto de operación, transitorios y señal pequeña; base de la familia SPICE.	Gratuito, Open-source
LTspice	Implementación freeware de SPICE por Analog Devices; interfaz gráfica, librerías de modelos de fabricantes.	Gratuito

TT0004: ARACNOTRÓN	Robot experimental en forma de araña; control electrónico y remoto; implicó uso implícito de lógica digital simulada.	Proyecto académico (ESCOM)
TT0007: Control de brazo artificial	Prototipo de mano artificial controlada por PC; lógica digital y control interactivo de hardware.	Proyecto académico (ESCOM)
TT0012: Control de brazo pintor	Proyecto de control preciso de actuadores; requería validación previa de lógica digital.	Proyecto académico (ESCOM)
TT0006: Sistema de electrocardiografía	Adquisición y análisis de señales ECG; diseño de front-end analógico con validación SPICE.	Proyecto académico (ESCOM)
TT0022: Fonocardiógrafo digital	Captura y análisis de sonidos cardíacos; uso implícito de simulación analógica y digital.	Proyecto académico (ESCOM)
TT0030: HM1 (Electroencefalógrafo digital)	Adquisición y transmisión de señales EEG; requería simulación mixta analógica (SPICE) y digital (HDL).	Proyecto académico (ESCOM)
TT0024: Especificación de protocolos con PROMELA	Validación formal de protocolos de red con simulador SPIN; paralelo conceptual a la simulación de hardware.	Proyecto académico (ESCOM)
Solución Propuesta	Aplicación local, multiplataforma, orientada a educación; depuración interactiva de HDL en tiempo real con visualización de señales, ciclos de reloj y puertos en una interfaz unificada.	Gratuito, Open-source

Tabla 1. Resumen de productos similares.

2. Objetivo

2.1 Objetivo General

Desarrollar una aplicación multiplataforma, gratuita y de código abierto, que permita la simulación y depuración interactiva de código en lenguajes de descripción de hardware (HDL) en un entorno local, facilitando el aprendizaje de diseño digital sin requerir hardware físico.

2.2 Objetivos Específicos

1. Implementar un analizador sintáctico (Parser) capaz de procesar código en lenguaje de descripción de hardware y generar una representación estructurada (AST) que incluya entidades, arquitecturas, señales y procesos.

2. Desarrollar un intérprete de HDL que ejecute la lógica descrita en el código, evaluando expresiones, gestionando estados de señales y simulando el comportamiento de circuitos digitales.

3. Diseñar un módulo de control temporal (TimeMachine) que administre el tiempo de simulación discreto, permitiendo controlar ciclos de reloj y el flujo de ejecución.

4. Construir una interfaz gráfica unificada que integre un editor de código, visualización de señales y controles de simulación, accesible desde múltiples plataformas (Windows, macOS, Linux, Android, iOS).

5. Proveer estimación de recursos de hardware (flip-flops, LUTs, pines I/O, dominios de reloj) para validar diseños contra especificaciones de dispositivos FPGA.

6. Definir los requerimientos técnicos del sistema, incluyendo versiones mínimas de sistemas operativos compatibles y especificaciones de hardware, con base en las capacidades del framework de desarrollo seleccionado (.NET MAUI para aplicaciones nativas y Blazor para la versión web).

3. Justificación

El presente trabajo se fundamenta en la necesidad de contar con herramientas académicas y profesionales que faciliten el aprendizaje, diseño y simulación de sistemas digitales mediante lenguajes de descripción de hardware y arquitecturas reconfigurables. Aunque existen alternativas comerciales y de código abierto, muchas presentan limitaciones en compatibilidad multiplataforma, accesibilidad y facilidad de uso, lo cual dificulta su incorporación tanto en entornos de enseñanza como de investigación.

Originalidad del trabajo. La propuesta se distingue por el desarrollo de una aplicación multiplataforma construida en .NET 9, con soporte para Windows, macOS, Android, iOS e iPadOS a través de MAUI. Este enfoque asegura que la solución funcione como un software instalable y autónomo, sin requerir un servicio web ni conexión a internet. De esta manera, se elimina la dependencia de arquitecturas cliente-servidor y se garantiza que los usuarios puedan ejecutar la aplicación en cualquier entorno local

Vinculación con los usuarios potenciales. Los principales beneficiarios serán estudiantes y docentes de asignaturas como Arquitectura de Computadoras, Fundamentos de Diseño Digital, Diseño de Sistemas Digitales y Sistemas en Chip. Investigadores y desarrolladores también podrán aprovecharla como herramienta de prototipado y validación rápida.

Mejora a lo ya existente. A diferencia de las soluciones actuales, que suelen fragmentar el flujo de trabajo o limitarse a un sistema operativo, este proyecto integra en una sola aplicación un entorno unificado, multiplataforma y accesible, entre sus aportaciones más destacadas se encuentran:

- Visualización dinámica de variables: Permite observar en cualquier momento los estados de señales, puertos y variables internas, facilitando la comprensión del comportamiento digital.
- Reducción de problemas de sincronización: Minimiza errores comunes al trabajar con arquitecturas como Cyclone IV o Spartan, evitando conflictos con múltiples relojes o diferencias entre plataformas.
- Apoyo académico directo: Ofrece a estudiantes y docentes una herramienta práctica para reforzar la enseñanza de HDL y diseño digital.
- Función de “pre-laboratorio”: Permite realizar prácticas de preparación antes de un laboratorio físico, optimizando el tiempo limitado en talleres y mitigando la falta de material disponible.
- Accesibilidad desde distintos dispositivos: Al poder ejecutarse en computadoras y dispositivos móviles, ofrece flexibilidad de estudio y práctica en cualquier lugar, sin depender de un laboratorio o de internet.
- Facilitador de prototipado: Un diseño que en un entorno tradicional podría tardar semanas en validarse, puede ser verificado en cuestión de días dentro de la aplicación, reduciendo tiempos de iteración y aumentando la productividad en proyectos académicos y de investigación.

Complejidad y viabilidad. El proyecto presenta un nivel de complejidad elevado al combinar simulación digital, entornos gráficos interactivos y desarrollo multiplataforma. Sin embargo, es viable gracias al uso de recursos open-source y librerías oficiales de Microsoft, que aseguran compatibilidad y estabilidad en diferentes sistemas operativos. El trabajo se ajusta a los tiempos y alcances definidos, integrando los conocimientos adquiridos a lo largo de la carrera en una aplicación con impacto académico y profesional real.

4. Productos o Resultados esperados

El desarrollo del Trabajo Terminal dará como resultado un conjunto de productos que permitirán no solo validar la implementación del sistema, sino también asegurar su adecuada utilización, mantenimiento y futura extensión. Dichos entregables contemplan tanto el software desarrollado como la documentación necesaria para garantizar su comprensión y uso por parte de estudiantes, docentes e investigadores.

Los principales productos esperados son:

- Código fuente en GitHub, con control de versiones y acceso público para fomentar la transparencia, colaboración y reutilización académica.
- Documentación técnica, que describa la arquitectura, diseño y funcionamiento interno del sistema, orientada a facilitar su mantenimiento y posibles mejoras futuras.
- Manual de usuario, con instrucciones claras y prácticas para la instalación, configuración y uso de la herramienta en distintos entornos.
- Aplicativo ejecutable para cada sistema operativo soportado (Windows, macOS, Android, iOS, iPadOS y versión web para Linux), asegurando la cobertura multiplataforma definida como objetivo principal del proyecto.
- Pruebas de validación, incluyendo casos de simulación de código HDL, pruebas de rendimiento en distintos dispositivos y evaluaciones de usabilidad para verificar la efectividad del sistema como recurso académico.

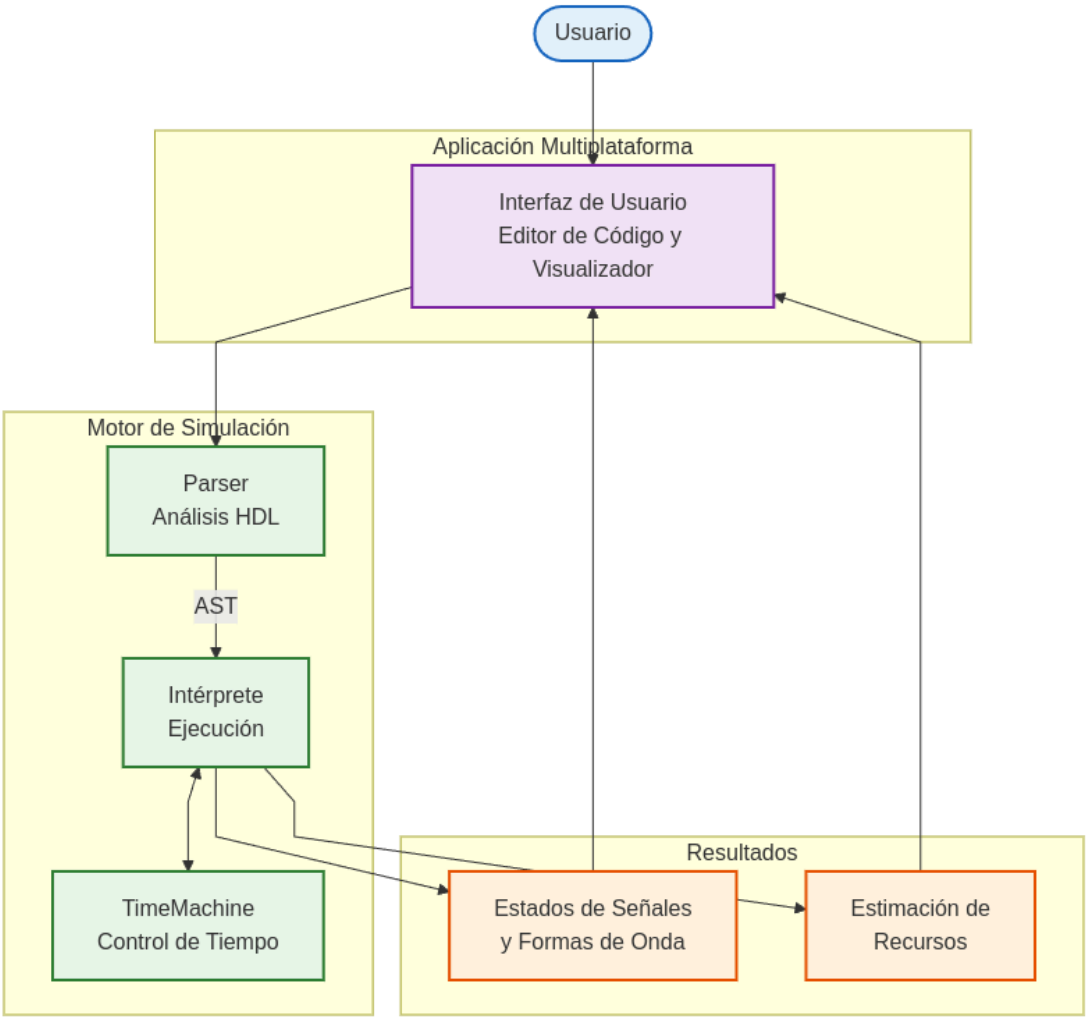

Figura 1. Arquitectura del sistema.

5. Metodología

Para el desarrollo del proyecto se adoptará la metodología Kanban, la cual permitirá organizar y dar seguimiento a las tareas de manera visual y flexible, adecuada al carácter individual del trabajo. Esta metodología facilitará la identificación del estado de cada actividad, promoviendo la priorización y el cumplimiento oportuno de las metas intermedias establecidas.

El proyecto se dividirá en fases principales:

- Análisis y planeación inicial: Definición detallada de los requerimientos, identificación de las plataformas objetivo y selección de librerías y frameworks compatibles.
- Diseño del sistema: Elaboración de la arquitectura del entorno, considerando la implementación del motor de simulación, el editor de código y la interfaz gráfica para visualización de señales.
- Implementación: Desarrollo progresivo de los módulos principales (editor de código, motor de simulación propio, visualización de señales digitales y generación de ejecutables multiplataforma).
- Pruebas y validación: Ejecución de casos de simulación, pruebas de compatibilidad en distintos sistemas operativos y validación con ejemplos académicos.
- Documentación y entrega final: Redacción de la documentación técnica y manual de usuario, así como la publicación del código fuente en un repositorio abierto.

Las técnicas y herramientas a emplear incluyen:

- Gestión de tareas: Tablero Kanban digital para organización y control de avances.
- Lenguaje y frameworks: .NET (C#) con MAUI y ASP.NET Blazor para garantizar la compatibilidad multiplataforma.
- Interfaz de usuario: Monaco Editor para la edición de código y librerías gráficas de Microsoft para la visualización de señales.
- Control de versiones: GitHub, para asegurar trazabilidad y acceso público al desarrollo.

6. Cronograma

Alumno: Ulrich Tamayo Daniel

Título del TT: Aplicación Multiplataforma para el Aprendizaje y Depuración de Código HDL sin Requerimiento de Hardware Físico

Actividad	**FEB**	**MAR**	**ABR**	**MAY**	**JUN**	**JUL**	**AGO**	**SEP**	**OCT**	**NOV**	**DIC**	**ENE**
Investigación del estado del arte												
Definición de requerimientos												
Análisis del alcance y planeación de módulos												
Definición de requerimientos técnicos y versiones compatibles												
Diseño de la arquitectura del sistema (MAUI/.NET, ASP.NET Blazor)												
Diseño de la interfaz y editor de código (Monaco)												
Desarrollo inicial del motor de simulación HDL (self-code)												
Documentación preliminar y entrega para **Evaluación TT1**												

Desarrollo del motor de simulación avanzado												
Implementación de la visualización de señales y ciclos de reloj												
Generación de ejecutables multiplataforma compatibles (Windows, macOS, Android, iOS, iPadOS, versión web Linux)												
Pruebas de validación (compatibilidad, rendimiento, usabilidad)												
Elaboración del manual de usuario y documentación técnica final												
Publicación del código fuente en GitHub y entrega para **Evaluación TT2**												

7. Referencias

[1] Geek Factory, *FPGA y CPLD - Tarjetas de desarrollo*, 2025. [En línea]. Disponible en: https://www.geekfactory.mx/categoria-de-producto/tarjetas/fpga-y-cpld/

[2] Digilent, *FPGA Development Boards*, 2025. [En línea]. Disponible en: https://digilent.com/shop/products/fpga-boards/

[3] Johns Hopkins Engineering for Professionals, *FPGA Design Using VHDL - 525.642*, 2025. [En línea]. Disponible en: https://ep.jhu.edu/courses/525642-fpga-design-using-vhdl/

[4] Intel, *Quartus Prime - Overview*, 2025. [En línea]. Disponible en: https://www.intel.com/content/www/us/en/software/programmable/quartus-prime/overview.html

[5] AMD Xilinx, *Vivado Design Suite - Downloads*, 2025. [En línea]. Disponible en: https://www.xilinx.com/support/download.html

[6] GHDL Project, *GHDL Main/Home Page*, 2025. [En línea]. Disponible en: http://ghdl.free.fr/

[7] GTKWave Documentation, *GTKWave User Guide*, 2025. [En línea]. Disponible en: https://gtkwave.github.io/gtkwave/

[8] Siemens EDA, *ModelSim - Product Overview*, 2025. [En línea]. Disponible en: https://eda.sw.siemens.com/en-US/ic/modelsim/

[9] F. Galindo Soria, *Trabajos Terminales de la ESCOM del IPN*, 2025. [En línea]. Disponible en: http://www.fgalindosoria.com/ligas/tt/index.htm

8. Alumnos y Directores

Alumno:

Ulrich Tamayo Daniel - Alumno de la carrera de Ing. en Sistemas Computacionales en ESCOM, Especialidad Sistemas, Boleta: 2021630722, Tel. 5561232707, Email: danielulrichtamayo@gmail.com

Firma: _____

Directores:

M en C. Pastrana Fernández Carlos Jesús. Maestría en Tecnología de Cómputo por el CIDETEC I.P.N. Maestría en Administración General U.N.A.L. Ingeniería en Sistemas Electrónicos por Tecnológico de Monterrey. Tel. 57296000, ext 52032. Email: cpastrana@ipn.mx.

Firma: _____

Dr. Cifuentes Alvarez Alejandro Sigrido. Ing. Electrónico en Comunicaciones. Maestría en Ingeniería en Sistemas. Dr. en Educación Superior. Area de interés: Programación de móviles inteligentes. Tel. 55 6610 9357. Email: avionica1@yahoo.com.mx.

Firma: _____