Instituto Tecnológico de Costa Rica

Escuela de Ingeniería Electrónica

EL-5617 Trabajo Final de Graduación

Diseño de un ambiente de verificación funcional para un bus digital empleando los estándares UVM y PSS

Anteproyecto

Mario Alberto Montero Marenco, 2019197658

Cartago, Costa Rica

20 de agosto de 2024

Índice

1. Entorno del proyecto

Los circuitos integrados digitales son componentes esenciales en la mayoría de los dispositivos electrónicos modernos. Con el avance tecnológico, su funcionalidad ha evolucionado hacia tareas más sofisticadas, aumentando su nivel de complejidad. De acuerdo con el estudio de [1], para el 2022 cerca de un 36 % de los circuitos integrados de aplicación específica se componían por más de 10 millones de compuertas lógicas, esto sin incluir la etapa de memoria. Esta evolución se traduce en elementos extremadamente robustos, a los cuales se necesita asegurar su correcto funcionamiento antes de poder integrarlos en sistemas comerciales. Lo que destaca la importancia de la verificación funcional en el flujo de diseño VLSI (Very Large-Scale Integration).

1.1. El proceso de verificación

El flujo de diseño VLSI se divide en etapas largas y complejas, empezando por los requerimientos del cliente hasta la materialización y comercialización del chip, como se puede apreciar en la Figura 1. Dentro de las etapas más críticas se encuentra la verificación funcional, ya que es aquí donde los ingenieros se aseguran que el dispositivo bajo prueba (DUT por sus siglas en inglés) se comporta de acuerdo a las especificaciones del cliente.

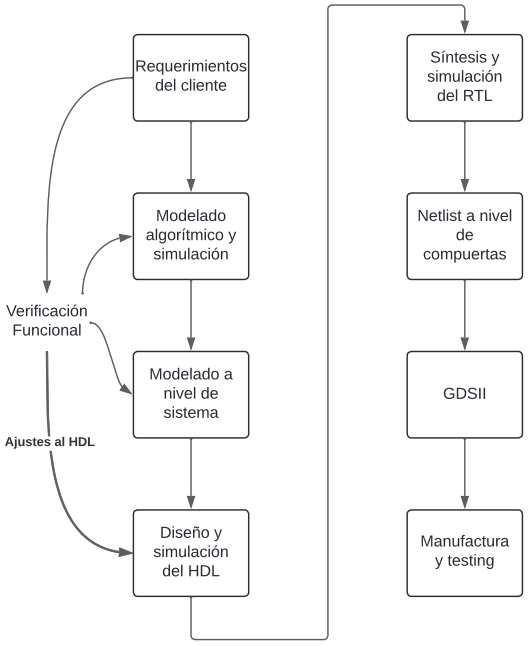

Figura 1: Etapas del proceso de diseño de un circuito integrado.

En el proceso de fabricación de un chip siempre se busca encontrar y remover los *bugs* en las etapas tempranas de la verificación, ya que esto le puede ahorrar a la empresa mucho dinero. Normalmente, el diseñador solo debe cambiar la descripción del hardware (por lo general escrita en Verilog o VHDL) para que posteriormente el equipo de verificación se asegure que el error fue debidamente corregido. Si, por el contrario, estas fallas en el diseño fuesen encontradas en las etapas finales, los costos de rehacer de un chip serían muy elevados. O inclusive podrían llegar a arruinar de manera permanente la reputación de una empresa en el caso de que los fallos sean detectados hasta que está en las manos del cliente [2].

1.2. Plan de verificación funcional

En el área de verificación se maneja lo que se llama un plan de pruebas (*test-plan*). Este concepto se vincula con la especificación del hardware y contiene una descripción de las características que deben ser probadas y las técnicas a utilizar. Estos pasos pueden incluir pruebas dirigidas o aleatorias, aserciones, emulación, cobertura esperada, pruebas formales e incluso el uso de módulos prediseñados específicos para verificación [3].

1.3. Metodologías de verificación

El propósito de las metodologías de verificación es ofrecer un manual con un conjunto común de estándares para establecer un consenso sobre el uso adecuado de un lenguaje de verificación. Se puede considerar como un plan para el éxito en la verificación y a menudo incluye una biblioteca de clases base. La metodología debe proporcionar habilidades prácticas portátiles que permitan a diferentes dominios compartir conocimientos y experiencia. Un problema con las primeras metodologías fue que estaban específicamente diseñadas para ciertos proveedores de herramientas [4].

1.4. Metodología de Verificación Universal (UVM)

Para llevar a cabo la verificación funcional de una manera eficiente se utiliza una metodología estandarizada para crear bancos de prueba (*testbench*), la cual se denomina Metodología de Verificación Universal (UVM). Dicha metodología fue creada por Accellera en cooperación con proveedores de EDA (Electronic Design Automation) como Synopsys, Mentor y Cadence. Su propósito principal es el de mejorar el desarrollo de ambientes de verificación al incrementar la compatibilidad entre sistemas y simplificar la reutilización de componentes de verificación [4].

Cada clase y componente en un *testbench* desarrollado con UVM tiene un propósito específico y una interfaz clara con las demás partes, lo que mejora la eficiencia y pro-

mueve la reutilización de código. Al combinar estos elementos, se crea un ambiente de verificación modular y reutilizable. Esto permite que los ingenieros se centren en la funcionalidad que debe ser verificada a nivel de transacción, así como en la generación de los estímulos que se le insertarán al DUT [5]. Esto aumenta el nivel de productividad, ya que una parte importante de la infraestructura del banco de pruebas se encuentra construida.

1.5. DCILab

El proyecto será desarrollado en el Laboratorio de Diseño de Circuitos Integrados (DCILab) de la Escuela de Ingeniería Electrónica del Instituto Tecnológico de Costa Rica. Este laboratorio se especializa en la microelectrónica para estudiar así el flujo de diseño a gran escala. Entre los principales proyectos desarrollados por el laboratorio se pueden mencionar desde la implementación de sistemas para la detección de disparos en bosques debido a caza furtiva hasta el desarrollo de circuitos de radiofrecuencia RFID, el Siwa, el cual es el primer microcontrolador de 32 bits diseñado en Costa Rica y basado en tecnología CMOS de 180 nm para aplicaciones médicas, además se han diseñado circuitos de muy bajo consumo capaces de procesar información acústica para detectar actividades ilegales en zonas protegidas.

1.6. Microcontrolador Siwa

Siwa se trata de un *system-on-chip* (SoC) cuyo propósito es servir como unidad central de procesamiento para dispositivos médicos implantables. Este dispositivo fue desarrollado como una cooperación entre el ITCR y la Universidad Católica del Uruguay. La Figura 2 muestra un diagrama de alto nivel de la arquitectura del microcontrolador. Se puede observar que el mismo cuenta con diversas interfaces para comunicarse con dispositivos externos, así como un núcleo encargado de realizar el procesamiento de los datos. Este último incluye el decodificador de instrucciones, la unidad de control, un banco de registros basado en latches, la Unidad Aritmética Lógica (ALU), entre otros [6].

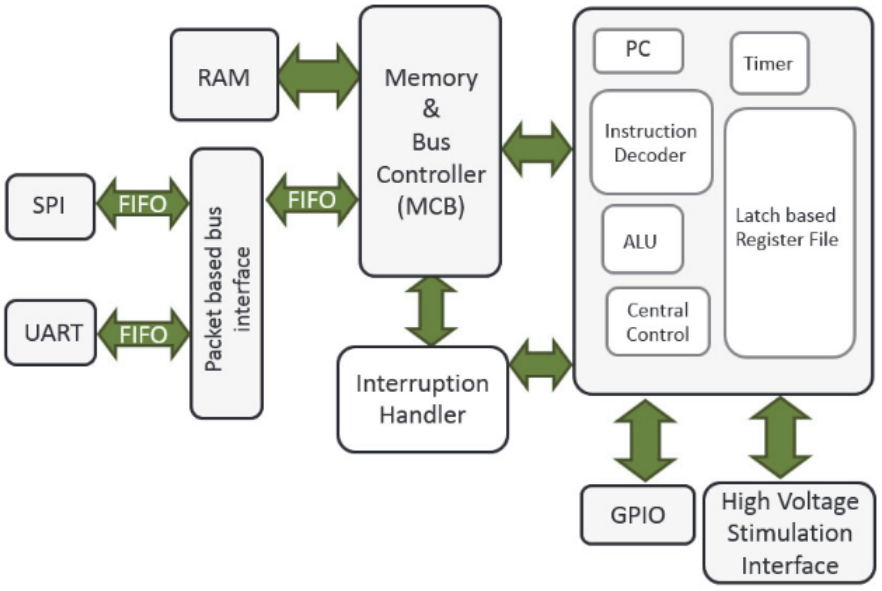

Figura 2: Descripción de alto nivel del microcontrolador Siwa [6].

1.7. Bus digital

El bus digital de la arquitectura del microcontrolador Siwa, representado en la Figura 3, facilita la comunicación entre los componentes conectados por medio de la utilización de los bloques denominados FIFOs (First In, First Out). Este dispositivo transmite y recibe paquetes con base en los campos de origen y destino, tal y como se muestra en la Figura 4.

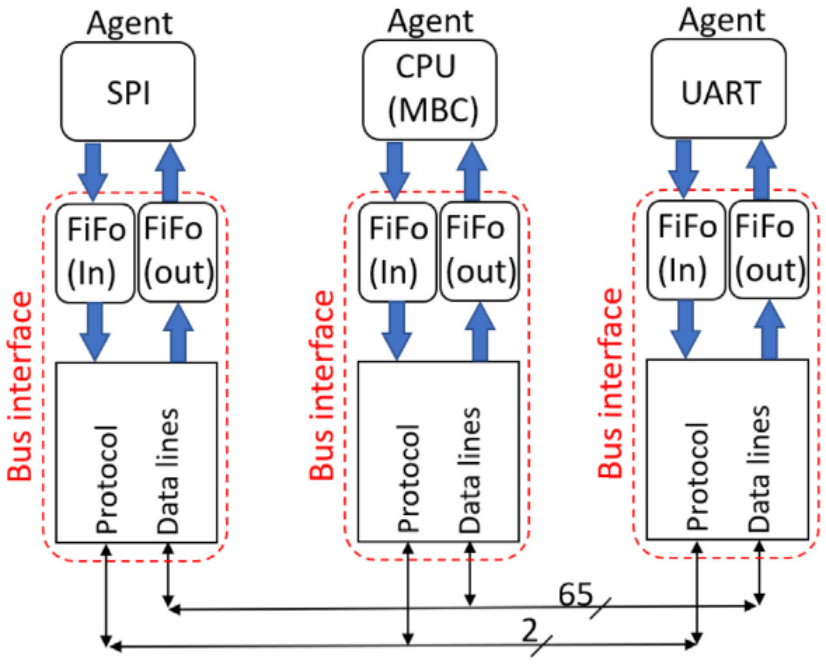

Figura 3: Micro-arquitectura del bus digital del microcontrolador Siwa [7].

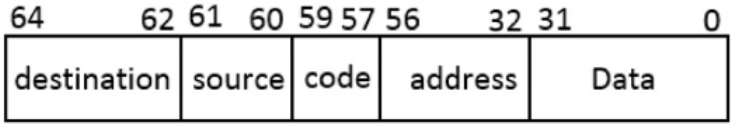

Figura 4: Formato del paquete de información usado por el bus digital [7].

2. Definición del problema

2.1. Generalidades

La verificación funcional en el flujo de diseño VLSI es esencial, dado que un diseño sin capacidad de validación carece de utilidad. Esta etapa, como se mencionó previamente, demanda considerable tiempo y esfuerzo debido a la complejidad en aumento de los diseños y la colaboración de diversas entidades en el desarrollo del circuito integrado. En este contexto, los ingenieros de validación suelen emplear distintas técnicas y lenguajes de programación para crear estímulos y casos de prueba similares, adaptándolos según la plataforma de trabajo, ya sea a nivel de bloque, unidad o sistema, como en un SoC [8].

Cuando se verifica un diseño a nivel de bloque y unidad, se utiliza con frecuencia SystemVerilog aunque también se usan algunos otros como e-language, SystemC y VHDL. A nivel de sistema, el software embebido se utiliza con frecuencia para ejercitar el diseño. Sin embargo, varias dificultades surgen debido a la utilización de diferentes lenguajes y técnicas en la verificación a nivel de bloque y subsistema. Ya que resulta difícil aprovechar los escenarios de prueba a nivel de bloque en el nivel de sistema. Además, el software embebido que genera estímulos en entornos de nivel SoC no proporciona soporte para la generación automatizada de estímulos, de la misma manera que lo hacen los lenguajes como SystemVerilog en entornos de bloque y subsistema [9]. Esta complejidad no solo dificulta la transferencia eficiente de casos de prueba entre niveles, sino también el intercambio óptimo de conocimientos entre equipos de ingenieros.

2.2. Problemática

El proyecto surge de la necesidad de la Escuela de Ingeniería Electrónica, específicamente del DCILab, de seguir adquiriendo conocimientos en el desarrollo de bancos de pruebas utilizando metodologías estandarizadas. Además, se busca profundizar en la construcción de modelos de estímulos y escenarios de prueba portables y reutilizables en los diferentes niveles de abstracción de un circuito integrado (es decir, bloque, subsistema, sistema). Para ello, se planea realizar la validación funcional del bloque del bus digital del microcontrolador Siwa, tomando como referencia los ambientes de prueba diseñados por estudiantes de semestres anteriores e incorporando un modelo que permita generar estímulos de prueba que garanticen el correcto funcionamiento del dispositivo.

2.3. Síntesis del problema

El DCILab necesita evaluar la capacidad que tienen sus ambientes de verificación funcional para integrarse con las nuevas técnicas y estándares de verificación del mercado.

3. Enfoque de la solución

El diseño del entorno de pruebas para el bus digital del microcontrolador Siwa debe realizarse utilizando la metodología estandarizada UVM. Este marco de trabajo proporciona una estructura con funciones estándar que facilita la instancia, conexión y construcción de los componentes en el entorno de simulación. De esta forma, se puede dedicar la mayor parte del tiempo del proyecto al desarrollo de un modelo de alto nivel capaz de generar los estímulos de prueba que ejerciten todas las capacidades del DUT.

3.1. Ambiente de pruebas con UVM

La Figura 5 muestra la arquitectura típica de un banco de pruebas desarrollado en UVM. En el nivel más alto de la jerarquía se encuentra el UVM *testbench*, en el cual se instancia el DUT, se crean las interfaces y se corre la prueba. Como la clase *UVM Test* se instancia dinámicamente durante el tiempo de corrida, el banco de pruebas puede ser compilado una sola vez y correr varias pruebas [10].

El *test* en UVM usualmente se encarga de ejecutar tres funciones principales: instanciar y configurar el ambiente, conectar las interfaces con el *driver* y el monitor, así como generar los estímulos y aplicarlos al DUT por medio de las secuencias. Por lo general, hay una única prueba base con la instanciación del ambiente y otros elementos comunes. Luego, otras pruebas individuales pueden extender esta prueba base y configurar el entorno de manera diferente o seleccionar secuencias diferentes para ejecutar [10].

El ambiente de UVM es un componente cuya función es la de agrupar otros componentes relacionados entre sí, como lo son el agente, *scoreboard* e incluso componentes encargados de medir la cobertura funcional, conocidos como suscriptores. Adicionalmente, el ambiente también debe conectar los diferentes componentes, por ejemplo, el puerto TLM (Transaction Level Modeling) de un monitor al puerto de un *scoreboard* [4].

El *UVM Scoreboard* se encarga de validar el funcionamiento del DUT. Este componente recibe las transacciones que entran y salen del dispositivo bajo prueba a través de los puertos de análisis del agente. Por medio de un modelo de referencia (también conocido como predictor), genera las transacciones esperadas de un determinado estímulo. Luego, hace una comparación entre la salida esperada y la salida actual, determinando si existen errores o inconsistencias [10].

Un agente en UVM encapsula al secuenciador, *driver* y monitor en una sola entidad, creando instancias y conectando los componentes a través de puertos TLM. Un agente puede ser configurado para ser de tipo pasivo o activo según se requiera. Uno pasivo solo instancia el monitor, lo cual es útil cuando no se tienen que conducir estímulos al DUT; su función se reduce a monitorear señales y medir cobertura funcional. Por otro

lado, un agente activo instancia los tres componentes, lo que permite la inserción de estímulos al DUT.

El secuenciador se encarga de generar transacciones como objetos de clase y las envía al *driver* para su posterior ejecución. En otras palabras, controla el flujo de las transacciones generadas por una o más secuencias. Por su parte, las secuencias de UVM se componen de varios elementos de datos que se pueden juntar de distintas formas para crear estímulos [10].

El *driver* o manejador en UVM es responsable de comunicarse a nivel de transacción con el secuenciador a través de un puerto TLM, y de convertir el objeto de secuencia (*sequence_item*) en una transacción o señal binaria para comunicarse con el DUT a través de una interfaz virtual. Como su nombre lo indica, los manejadores típicamente reciben un objeto de secuencia y utilizan esa información para conducir señales a una interfaz particular del diseño, y en ciertas aplicaciones, también pueden recibir una respuesta a nivel de puerto del DUT y convertirla nuevamente en un objeto de secuencia para que así se complete la transacción [5].

Por último, el monitor se encarga de supervisar y capturar la información proveniente de la interfaz del DUT para que sea enviada a través de los puertos de análisis TLM hacia los otros componentes del banco de pruebas, como lo puede ser el *scoreboard*.

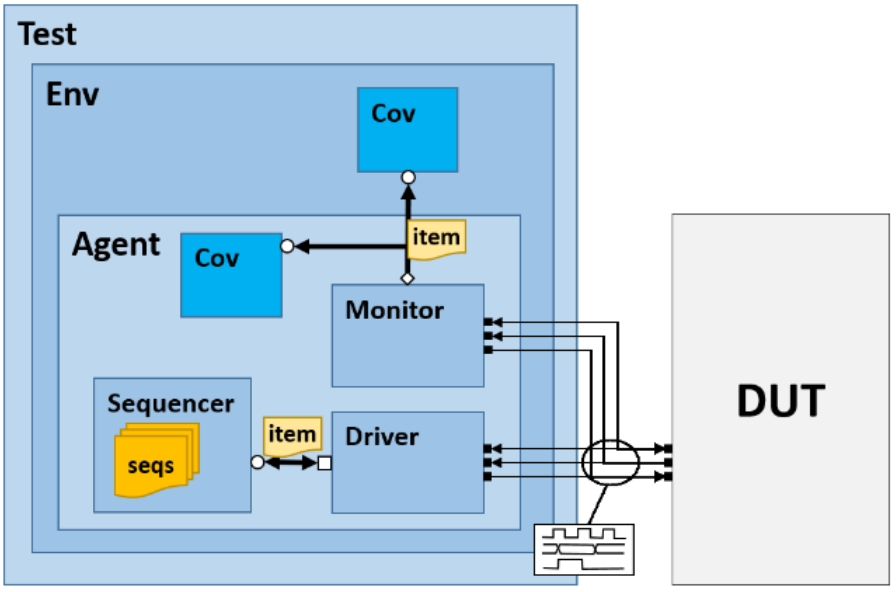

Figura 5: Arquitectura típica de un banco de pruebas en UVM [5].

3.2. Modelo de abstracción PSS

La alternativa que se presenta en este documento para generar el modelo de escenarios de prueba para el bus digital es la adopción del nuevo estándar PSS (Portable Test and Stimulus Standard) de Accellera Systems Initiative [11]. Este estándar busca generar estímulos portables a través de diversas plataformas (e.g., simulación del RTL, prototipado en FPGA o validación post-silicio) y niveles de verificación (e.g., bloque, subsistema o sistema).

Accellera creó el estándar PSS para facilitar la automatización de la generación de estímulos y la reutilización de la misma especificación en múltiples lenguajes de verificación [9]. Básicamente, PSS define una especificación para crear una representación única de escenarios de prueba, utilizable por una variedad de usuarios en diferentes niveles de integración y bajo diferentes configuraciones. Mediante la cual los usuarios pueden especificar un conjunto de comportamientos una vez, de donde se pueden derivar múltiples implementaciones [11].

El estándar toma sus conceptos fundamentales de lenguajes de programación orientados a objetos, lenguajes de verificación de hardware y lenguajes de modelado de comportamiento. PSS cuenta con herramientas integradas para representar conceptos fundamentales del sistema, como control y flujo de datos, concurrencia y sincronización, requisitos de recursos, y estados y transiciones [11].

Es importante mencionar que PSS no es un lenguaje o técnica para reemplazar ningún método de verificación como SystemVerilog y UVM. No crea un banco de pruebas para la verificación, simplemente define un lenguaje específico de dominio (DSL) para la generación del modelo de abstracción de estímulos de prueba. La infraestructura como un *testbench* de UVM debe existir para ejecutar las pruebas generadas utilizando el modelo. El estándar trabaja a un nivel de abstracción más alto que es independiente del tipo de plataforma objetivo. Asimismo, el ambiente basado en UVM es utilizado y mejorado por la implementación de PSS, lo que permite aprovechar los beneficios de esta metodología, como la reutilización de componentes y un enfoque estándar predefinido [8].

3.3. Diagrama de la solución

Dado que la solución implica la adopción del nuevo estándar PSS para generar los estímulos y escenarios de prueba en el ambiente de verificación funcional desarrollado con UVM, la Figura 6 presenta un diagrama de la propuesta de solución, el cual está basado con lo que se menciona en [12]. A continuación, se ofrece una breve explicación de la funcionalidad de cada bloque:

- Modelo de abstracción PSS: Se definen los escenarios y los casos de prueba a un nivel de mayor abstracción usando un DSL.
- Compilador DSL: Toma el modelo abstracto PSS y lo traduce a un modelo de escenario que puede ser utilizado en la generación de pruebas.
- Modelo del escenario: Contiene los escenarios detallados que se han traducido desde el modelo de abstracción PSS.
- Solucionador de restricciones: Aquí se toma el modelo del escenario y se resuelve cualquier restricción definida, generando así un modelo resuelto.
- Modelo resuelto: Este es el resultado del solucionador de restricciones, un modelo que cumple con todas las restricciones y condiciones especificadas en el modelo PSS.
- Generador de pruebas: Utiliza el modelo resuelto para mapear los escenarios de prueba en el lenguaje objetivo, en este caso se trata de SystemVerilog.
- Banco de pruebas UVM: Contiene los componentes diseñados previamente para realizar la verificación del bus digital. Son de particular relevancia las secuencias, ya que contienen los estímulos generados a partir del modelo resuelto.

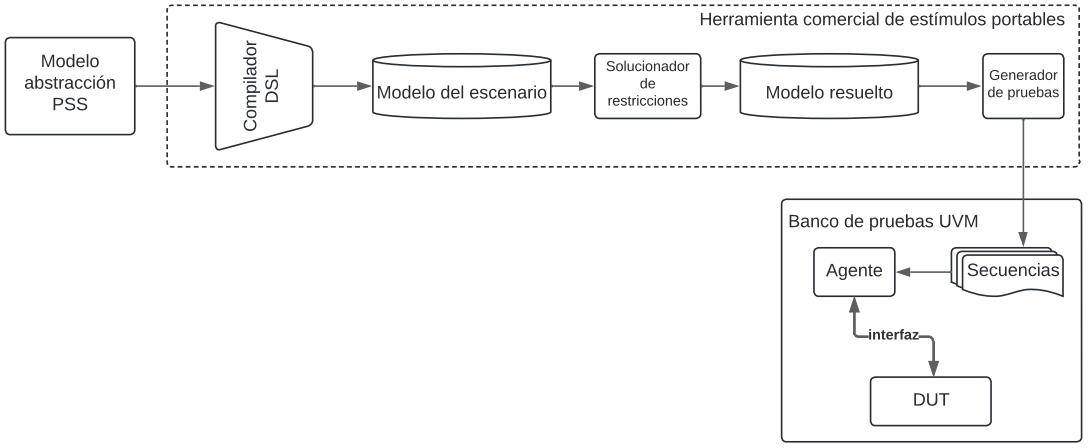

Figura 6: Diagrama de la solución.

4. Meta

Crear un modelo de estímulos portable y reutilizable en los procesos de validación presilicio del microcontrolador Siwa de 32 bits, ya sea a nivel de bloque, subsistema o sistema, desarrollados en el Laboratorio de Diseño de Circuitos Integrados del Instituto Tecnológico de Costa Rica.

5. Objetivos

A continuación se presentan los objetivos generales y específicos que se deben alcanzar al finalizar el proyecto.

5.1. Objetivo general

Diseñar un ambiente de verificación funcional para el bus digital del microcontrolador Siwa utilizando la metodología UVM y el estándar PSS, con el fin de evaluar la capacidad de generación de estímulos y escenarios de prueba de esta nueva técnica de verificación mediante el uso de simuladores comerciales.

Indicador: Se entregará un ambiente de verificación con múltiples secuencias utilizando los estándares de UVM y PSS, y un reporte con los detalles técnicos de esta implementación.

5.2. Objetivos específicos

1. Diseñar los componentes básicos de un ambiente de verificación en UVM para el bus digital del microcontrolador Siwa, utilizando SystemVerilog y VCS.
   Indicador: Compilación sin errores en la herramienta VCS de Synopsys del ambiente de verificación con todos los componentes de la Figura 5 y del estándar [10].

2. Crear un modelo de estímulos de prueba para verificar el funcionamiento del bus digital del microcontrolador Siwa empleando el lenguaje específico de dominio del estándar PSS.
   Indicador: Generación de un escenario de prueba mediante la herramienta VCPS de Synopsys y que el código cumpla a su vez con la especificación del estándar PSS de [11].

3. Ejecutar las pruebas generadas por el modelo de estímulos portables en el entorno UVM del bus digital utilizando el simulador VCS, para evaluar el rendimiento de la nueva técnica de verificación funcional.
   Indicador: Se deben evaluar las siguientes métricas: el tiempo de elaboración de las pruebas generadas por el modelo PSS, el tiempo de elaboración del ambiente y los porcentajes de cobertura comparados con los obtenidos por los métodos de verificación previos de [13] y [14].

6. Procedimiento para la ejecución del proyecto

Para llevar a cabo el proyecto se deben cumplir un conjunto de tareas para comprender tanto el ambiente de verificación actual de bus digital del microcontrolador, así como el integrar el modelo de estímulos portables de la especificación [11] por medio de una herramienta capaz de mapear esas definiciones en un conjunto de escenarios de prueba escritos en SystemVerilog. Como cada una de estas actividades involucrará una serie de retos que deben ser afrontados de manera gradual, se propone tanto el proceso metodológico de la Tabla 1 como la dependencia de sus actividades y su duración semanal en la Tabla 2.

Tabla 1: Lista de actividades.

Indicador	**Actividad**	**Objetivo**
A	Investigación y estudio sobre la arquitectura del bus digital del microcontrolador Siwa de 32 bits	1, 3
B	Investigación y estudio sobre la metodología UVM	1
C	Investigación sobre el estándar PSS	2
D	Entrenar en materia de generación de estímulos portables para el bus digital	2, 3
E	Investigación y estudio de la herramienta capaz de mapear el modelo PSS a una especificación de SystemVerilog	2, 3
F	Depurar los posibles errores del ambiente de pruebas actual del bus digital	1
G	Ejecutar la simulación del bus digital con el entorno básico UVM	1
H	Diseñar los componentes del modelo de comportamiento de estímulos portables PSS adecuados para el bus digital	2
I	Adaptación del ambiente UVM del bus digital para generar las pruebas desde el modelo de abstracción PSS	2
J	Ejecución del modelo de estímulos portables en la herramienta comercial de estímulos portables así generar los escenarios de prueba en SystemVerilog UVM.	2, 3
K	Ejecutar la simulación del ambiente de verificación UVM en la herramienta VCS junto con las nuevas pruebas generadas por el modelo de estímulos portables	2, 3
L	Validar el funcionamiento del bus digital con las nuevas funcionalidades añadidas por el modelo de abstracción de estímulos portables	3
M	Documentación del proyecto	1, 2, 3

Tabla 2: Dependencias y tiempo de duración del proyecto.

Actividad	**Dependencia**	**Duración** (semanas)
A	-	1
B	-	1
C	-	1
D	C	1
E	D	1
F	A, B	1
G	F	1
H	D	2
I	E, G, H	1
J	I	2
K	J	2
L	A, K	2
M	A-L	17

7. Cronograma

En la Figura 7 se encuentra el diagrama de Gantt, el cual muestra la planificación de las actividades mencionadas en las Tablas 1 y 2. El tiempo de cada actividad se mide en semanas, empezando desde el 22 de julio del 2024 hasta el 28 de noviembre del mismo año, lo que se traduce en un periodo de 18 semanas. Posteriormente, a partir del diagrama de Gantt se generó el diagrama Pert, el cual se puede observar en la Figura 8.

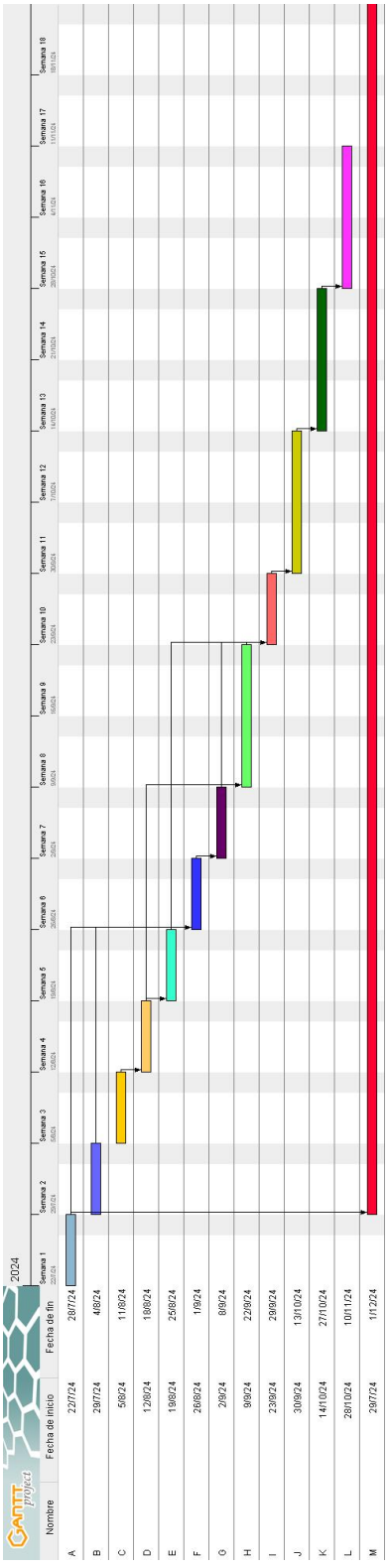

Figura 7: Diagrama de Gantt.

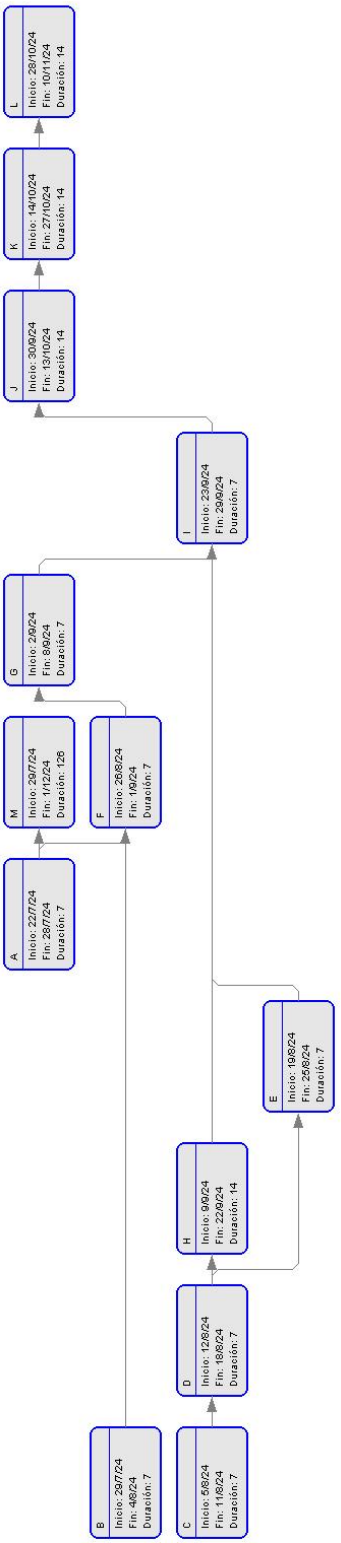

Figura 8: Diagrama Pert.

8. Uso de recursos

En esta sección se presenta una lista de los recursos necesarios para realizar el proyecto, la cual se puede apreciar en la Tabla 3.

Tabla 3: Uso de los recursos.

Recurso	**Tipo**	**Estado**
Computadora de escritorio	Equipo tecnológico	Disponible
Computadora portátil	Equipo tecnológico	Disponible
Licencia de Synopsys VCS	Licencia	Disponible
Licencia de Synopsys VCPS	Licencia	Disponible
Servidor	Equipo tecnológico	Disponible
Escritorio	Equipo de trabajo	Disponible
Internet	Servicios no personales	Disponible
Electricidad	Servicios no personales	Disponible
Agua	Servicios no personales	Disponible

9. Presupuesto

En las Tablas 4 y 5 se presenta el resumen presupuestario de los recursos necesarios para desarrollar el proyecto.

Tabla 4: Costo del equipo.

Equipo	**Costo** (colones costarricenses)
Computadora de escritorio (personalizada)	750 000
Computadora portátil	350 000
Viáticos	No disponible
Total	1 100 000

Tabla 5: Costo de los servicios públicos.

Servicio	**Costo por mes** (colones costarricenses)
Electricidad	30 000
Agua	15 000
Internet	34 000
Total	79 000
Total a 4 meses (16 semanas)	316 000

Referencias

[1] Wilson Research Group and Mentor. (2022) The 2022 Wilson Research Group Functional Verification Study. [Online]. Available: https://blogs.sw.siemens.com/verificationhorizons/2022/10/10/prologue-the-2022-wilson-research-group-functional-verification-study/

[2] B. Wile, J. C. Goss, and W. Roesner, *Comprehensive Functional Verification: The Complete Industry Cycle*, 1st ed. Morgan Kaufmann, 2005.

[3] C. Spear, *SystemVerilog for verification: A Guide to Learning the Testbench Language Features*, 2nd ed. Springer Science & Business Media, 2008.

[4] N. Karlsson, "Development of a new verification environment for a GPU hardware block using the Universal Verification Methodology (UVM)," Tésis de Maestría, Department of Electrical and Information Technology, Faculty of Engineering, LTH, Lund University, SE-221 00 Lund, Sweden, 2020.

[5] Verification Methodology Team, "Universal Verification Methodology UVM Cookbook," *Siemens Digital Industries Software*, 2018. [Online]. Available: https://verificationacademy.com/cookbook/

[6] R. Garcia-Ramirez, A. Chacon-Rodriguez, R. Castro-Gonzalez, A. Arnaud, M. Miguez, J. Gak, R. Molina-Robles, G. Madrigal-Boza, M. Oviedo-Hernandez, E. Solera-Bolanos, D. Salazar-Sibaja, D. Sanchez-Jimenez, M. Fonseca-Rodriguez, J. Arrieta-Solorzano, and R. Rimolo-Donadio, "Siwa: a risc-v rv32i based microcontroller for implantable medical applications," in *2020 IEEE 11th Latin American Symposium on Circuits & Systems (LASCAS)*, 2020, pp. 1–4.

[7] R. Garcia-Ramirez, A. Chacon-Rodriguez, R. Molina-Robles, R. Castro-Gonzalez, E. Solera-Bolanos, G. Madrigal-Boza, M. Oviedo-Hernandez, D. Salazar-Sibaja, D. Sanchez-Jimenez, M. Fonseca-Rodriguez, J. Arrieta-Solorzano, R. Rimolo-Donadio, A. Arnaud, M. Miguez, and J. Gak, "Siwa: A custom risc-v based system on chip (soc) for low power medical applications," *Microelectronics Journal*, vol. 98, p. 104753, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0026269219303787

[8] J. Nagar, T. Dworzak, S. Simon, U. Heinkel, and D. Lettnin, "Metapss: An automation framework for generation of portable stimulus model," in *DVCon Europe 2023; Design and Verification Conference and Exhibition Europe*, 2023, pp. 79–85.

[9] G. Moretti, "Accellera's support for esl verification and stimulus reuse," *IEEE Design & Test*, vol. 34, no. 4, pp. 69–75, 2017.

[10] Accellera, Systems Initiative. (2015) Universal Verification Methodology (UVM) 1.2 User's Guide. [Online]. Available: https://www.accellera.org/activities/working-groups/uvm

[11] ——. (2023) Portable Test and Stimulus Standard Version 2.1. [Online]. Available: https://www.accellera.org/activities/working-groups/portable-stimulus

[12] M. Ballance, "Designing a pss reuse strategy," *DVCon*, 2019. [Online]. Available: https://dvcon-proceedings.org/document/designing-a-pss-reuse-strategy-paper/

[13] G. Carranza-Otárola, "Diseño y caracterización de un ambiente de verificación funcional mixto para un Bus Digital, mediante SystemVerilog y Python," Tesis de Proyecto de Graduación, Instituto Tecnológico de Costa Rica, Cartago, Costa Rica, 2019.

[14] J. A. Rojas-Meza, "Diseño de un Modelo de Validación para la Interfaz de Bus de un Procesador RISCV, Mediante Verificación Formal Propietaria y Afirmaciones de SystemVerilog (SVA)," Informe de Trabajo Final de Graduación, Instituto Tecnológico de Costa Rica, Cartago, Costa Rica, 2021.

A. Anexo A. Hoja de Información del Proyecto

Información del estudiante:
Nombre: Mario Alberto Montero Marenco
Cédula: 1-1790-0631
Carné: 2019197658
Dirección de su residencia en época lectiva: Goicoechea, San José, Costa Rica
Teléfono: 7167-3346
E-mail: `mariomm@estudiantec.cr`

Información del proyecto:
Nombre del proyecto: Diseño de un ambiente de verificación funcional para un bus digital empleando los estándares UVM y PSS
Área del proyecto: Verificación Funcional de Circuitos Integrados

Información de la empresa:
Nombre: DCILab, Laboratorio de Diseño de Circuitos Integrados, Instituto Tecnológico de Costa Rica
Zona: Dulce Nombre, Cartago, Costa Rica
Dirección: Calle 15, Avenida 14, 1 km Sur de la Basílica de los Ángeles, 30109.
Teléfono: 2552-5333
Sitio web: `www.tec.ac.cr`
Actividad principal: Universidad Pública

Información del asesor en la empresa:
Nombre: Roberto Molina Robles
Puesto que ocupa: Profesor
Departamento: Escuela de Ingeniería Electrónica
Profesión: Ingeniero en Electrónica
Grado académico: Máster
Teléfono: 2550-1670 **Ext:** N/A
E-mail: `rmolina@itcr.ac.cr`