INSTITUTO TECNOLÓGICO DE COSTA RICA

ESCUELA DE INGENIERÍA ELECTRÓNICA

IMPLEMENTACIÓN DE UN AMBIENTE DE VERIFICACIÓN UART MEDIANTE MÉTODO UVM

CARLOS ALBERTO ÁLVAREZ HERNÁNDEZ

I SEMESTRE 2018

INSTITUTO TECNOLÓGICO DE COSTA RICA

ESCUELA DE INGENIERÍA ELECTRÓNICA

PROYECTO DE GRADUACIÓN

ACTA DE APROBACIÓN

Defensa de Proyecto de Graduación
Requisito para optar por el título de Ingeniero en Electrónica
Grado Académico de Licenciatura
Instituto Tecnológico de Costa Rica

El Tribunal Evaluador aprueba la defensa del proyecto de graduación denominado Implementación de un ambiente de verificación UART mediante método UVM, realizado por señor Carlos Alberto Álvarez Hernández y, hace constar que cumple con las normas establecidas por la Escuela de Ingeniería Electrónica del Instituto Tecnológico de Costa Rica.

Miembros del Tribunal Evaluador

Ing. Esteban Baradín Méndez

Profesor lector

Ing. Aníbal Ruiz Barquero

Profesor lector

Ing. Roberto Molina Robles

Profesor asesor

I. Resumen:

Este trabajo menciona la construcción de un microprocesador desarrollado por La Escuela de Ingeniería Electrónica del ITCR. Este microprocesador tiene como objetivo ser utilizado en dispositivos de aplicación médica implantable, por lo que a lo largo de este documento se expone el origen de la necesidad que conlleva la creación del dispositivo, como se diseña e implementa y finalmente como se verifica su funcionamiento.

Es precisamente en la verificación funcional donde se encuentra el enfoque de este proyecto, por lo que se muestran los pasos generales para implementar un ambiente de verificación. Estos pasos conllevan un estudio del dispositivo electrónico con el fin de comprender su funcionamiento y así desarrollar un entorno de pruebas capaz de evaluar su comportamiento.

El desarrollo de este documento se centra en la verificación del Módulo de UART del microprocesador. Este módulo se encarga de establecer la comunicación entre el dispositivo y un sistema externo.

Para este caso se utilizó el método de verificación UVM, utilizado altamente en la industria debido a su rapidez de implementación y su facilidad para ser reutilizado.

Tras la implementación del ambiente de verificación se logró evaluar el la UART, encontrando errores en su funcionamiento.

Cabe destacar, que el desarrollo de la UART continua en proceso de diseño e implementación, por lo que el ambiente UVM se encuentra susceptible a modificaciones posteriores, según sea necesario para cumplir los nuevos requerimientos del grupo de arquitectos, a cargo del proyecto.

II. Abstract

This work mentions the construction of a microprocessor developed by the School of Electronic Engineering of ITCR. This microprocessor is intended to be used in implantable medical application devices, so throughout this document is exposed the origin of the need that involves the creation of the device, how it is designed and finally how its operation is verified.

It is precisely in the functional verification where the focus of this project lies, that is why the general steps to implement a verification environment are shown. These steps involve a study of the electronic device to understand its operation and so they do with a testing environment for their behavior.

The development of this document focuses on the verification of the UART Module of the microprocessor. This module is responsible for establishing communication between the device and an external system.

For this case, the UVM verification method was used, it was highly used in the industry due to its rapid implementation and its ease of reuse.

III. Dedicatoria

Se agradece a:
Roberto Molina por su apoyo
durante el desarrollo del proyecto,
a mis padres y abuelos, Pedro Álvarez,
Dénice Hernández, Ramón Claudio
y Luz Casillo.

IV. Índice General

V. Índice de Figuras

VI. Índice de Cuadros

Capítulo 1. Introducción y Generalidades:

1.1. Generalidades

UVM - Método Universal de Verificación.

TEC – Instituto Tecnológico de Costa Rica.

CI – Circuito Integrado

RTL - Descripción a Nivel de Registros.

DUT – Diseño Bajo Prueba

OVM - Método de Verificación Abierta.

VMM - Método Manual de Verificación.

TLM – Modelado a Nivel de Transistores.

1.2. Alcance del Proyecto

Cada día la medicina actual demanda de nuevos instrumentos tecnológicos que ayuden a los médicos a combatir patologías que afectan a sus pacientes. Esos dispositivos tecnológicos deben tener alta capacidad de procesamiento para lograr englobar todas las variables que constituyen una enfermedad o padecimiento.

Debido a esto, La Escuela de Ingeniería Electrónica del Instituto Tecnológica de Costa Rica, busca desarrollar un microprocesador arquitectura RISC-V 32 bits de aplicación médica implantable.

Para lograr este objetivo, se crearon tres grandes grupos de trabajo, el primero conformado por profesores, los cuales son encargados de estipular los requerimientos de diseño del dispositivo, mientras que los dos grupos restantes, formados por estudiantes, se encargaron de implementar y verificar los módulos que constituyen la micro arquitectura del microprocesador.

Para fines de este documento, se expone solamente la etapa de verificación del módulo UART del microprocesador, estimulando y sometiendo sus entradas a diferentes condiciones, todo con el fin de determinar su funcionalidad aún en circunstancias no consideradas durante su etapa de diseño.

1.3. Meta y Objetivos

Meta:

Desarrollar los ambientes de verificación funcional que correspondan a cada uno de los módulos que componen el microprocesador RISC-V 32 bits de aplicación médica implantable.

Indicador: Existe un módulo de verificación para cada uno de los bloques funcionales (ALU, UART, RAM, entre otros) que componen el microprocesador.

Objetivo General:

Diseñar un ambiente de verificación funcional para el módulo UART de un microprocesador UVM RISC-V 32 bits, siguiendo la metodología UVM.

Indicador: Existe un ambiente de verificación funcional según la metodología UVM para el módulo UART desarrollado por el grupo de diseñadores del microprocesador RISC-V 32 bits.

Objetivos Específicos:

1) Desarrollar el secuenciador, driver y monitor del ambiente de verificación UVM.

Indicador: Las salidas y entradas transmitidas por el driver y monitor, se acoplan a las terminales del DUT. Además, el secuenciador genera las entradas necesarias para estimular la UART y consideradas en la Tabla de Coberturas.

2) Construir el modelo de referencia dentro del Scoreboard.

Indicador: El modelo de referencia cumple con el funcionamiento señalado en *Micro-UART* (Joon Lee, 2001), utilizado para el diseño de la UART.

3) Elaborar el plan de verificación.

Indicador: El plan de verificación cumple con lo descrito en la Tabla 3.

1.4. Metodología

1) Estudiar y desarrollar un plan de verificación de funcional para un módulo UART.

2) Implementar un ambiente de verificación siguiendo el plan de verificación.

3) Desarrollar las pruebas para obtener la Tabla de Coberturas del Ambiente de Verificación.

4) Analizar los resultados de cobertura y funcionamiento del ambiente de verificación.

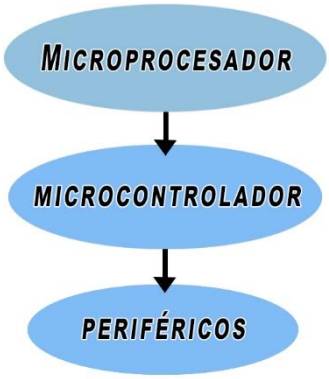
1.5. Dispositivos Electrónicos de Aplicación Médica

El avance científico ha influenciado la creación de nuevas tecnologías en beneficio del ser humano y la medicina no escapa a ello. Es por esto, que la industria médica ha buscado la manera desarrollar nuevos dispositivos que permitan mejorar el diagnostico, control y cura de enfermedades que aquejan la salud y la calidad de vida de las personas.

La electrónica moderna ha tomado un papel protagónico en el desarrollo de aplicación médica, capaces de convertirse en la mano derecha de los médicos, enfermeros y pacientes en la lucha contra sus padecimientos.

Estos dispositivos se encuentran formados por módulos generales que constituyen el sistema. Estos se visualizan a continuación:

Figura 1.1. Estructura básica de un dispositivo electrónico para aplicación médica



Tal y como muestra en el diagrama anterior, un sistema electrónico con finalidad médica posee un microcontrolador, un microprocesador y periféricos. Estos tres trabajan en conjunto para censar, procesar y controlar una aplicación específica. La descripción de los módulos se detalla de la siguiente manera:

Microcontrolador: Se encarga de controlar los periféricos que componen el sistema, ejecutando las acciones que deban de realizar o funcionando de interfaz para adquirir los datos de censado y enviarlos al microprocesador para que este los procese.

Microprocesador: Es el encargado de procesar las señales monitoreadas.

Periféricos: Son los que interactúan directamente con el paciente o sus síntomas. Se encargan de censar las variables físicas y transmitirlas al microcontrolador, esperar respuesta y ejecutar las acciones que procese el microprocesador.

Capítulo 2 Marco Teórico

2.1. Microprocesador

Una de las unidades esenciales de cualquier dispositivo electrónico es el microprocesador, el cual define su complejidad debido a la aplicación a la que se encuentra enfocado. Para el caso de una aplicación médica, este se basa en la cantidad de imágenes que deba procesar o la cantidad de cálculos matemáticos que realice (Díaz, 2015), por lo que a continuación se detalla una gráfica que muestra la función del microprocesador y el grado de complejidad que se necesita para realizar dicha tarea:

Figura 2.1. Complejidad de un microprocesador según su aplicación médica

Un CPU está formada por submódulos, los cuáles realizan una función específica. Estos bloques se encuentran internamente conectados y juntos se encargan de ejecutar los procesos del microprocesador (Tocci y Widmer, 2003).

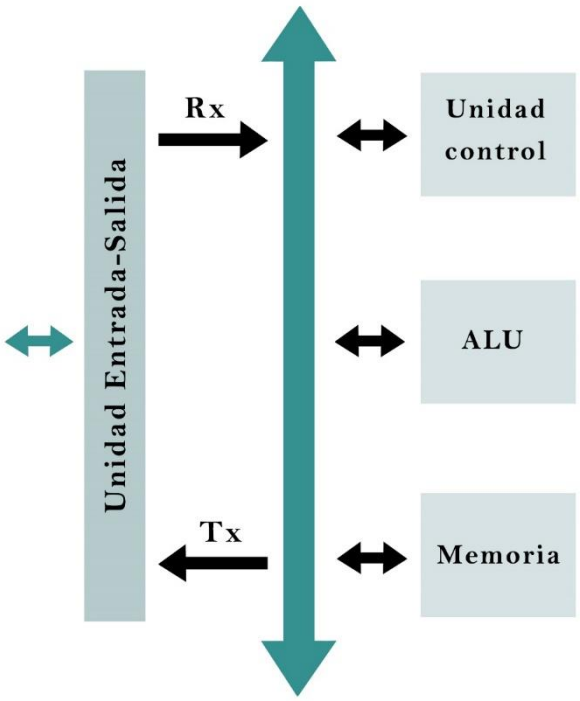


Figura 2.2. Estructura general de un microprocesador

En la Figura 2.2 se muestran las unidades generales que conforman la micro arquitectura de un CPU. Estos bloques se detallan a continuación:

Unidad Lógica – Aritmética (ALU): Es en la ALU donde se llevan a cabo las operaciones de datos. Estas operaciones se ejecutan según la instrucción que se desea realizar y pueden ir desde sumas o restas para cálculos aritméticos, hasta operaciones lógicas como AND y OR.

Unidad de Memoria: Se encarga de almacenar grupos de dígitos binarios (palabras) que representan las instrucciones que debe ejecutar la ALU ante la corrida de un programa específico. Adicionalmente la memoria almacena datos intermedios o finales de operaciones aritméticas o lógicas.

Unidad de Entrada y salida: Es la unidad encargada de conectar el microprocesador con el exterior. Este exterior puede estar constituido por sensores o demás circuitos electrónicos denominados periféricos.

Unidad de Control: Es la encargada de dirigir y controlar a todas las demás unidades. Este bloque posee sincronizadores que generan la señal requerida para ejecutar las instrucciones que componen el programa.

2.2. UART

Un microprocesador debe contener dentro de sus bloques internos, una unidad que establezca la entrada y salida de datos y su comunicación con el exterior.

Una unidad encargada de la comunicación entre un sistema y otro es el Transmisor – Receptor Asíncrono Universal UART, por sus siglas en inglés. Es un dispositivo encargado de controlar dos puertos seriales, uno que actúa como RX y el otro como TX. El funcionamiento de estos puertos se realiza de manera independiente y su objetivo es servir de interfaz entre algún sistema embebido y un periférico (J.Lee, 2001).

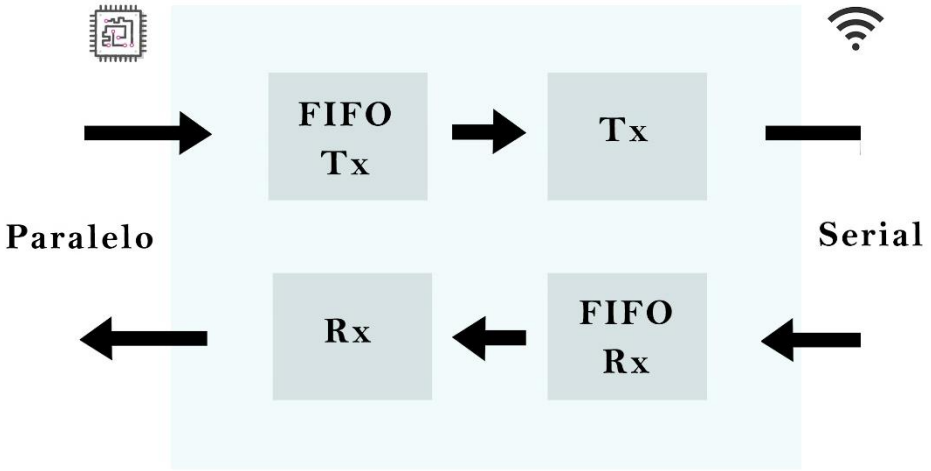


Figura 2.3. Estructura básica de un módulo UART

2.2.1. Protocolo de Operación UART

La comunicación UART posee parámetros que determinan el protocolo con el que se realiza el trasiego de información entres dispositivos. Estas variables se mencionan en la siguiente tabla:

Tabla 2.1Parámetros generales que debe contener una UART

Variable	Función
Largo de la Palabra	Establece la cantidad de Bits que constituye la palabra transmitida, esta tiene un mínimo de “0” y un máximo de 8. Usualmente las palabras son de “8” bits.
Reloj del Sistema	Establece el reloj de entrada al módulo UART.
Baud	Esta variable determina la velocidad con que se transmite y envían los datos a través del sistema. Los valores establecidos para esta variable son 1200, 2400, 9600, entre otras.

Además de los parámetros anteriores, el protocolo UART establece como se debe realizar el empaquetamiento de la información a través de los puertos seriales. Este formato establece como debe de iniciarse la transmisión de la palabra, como finaliza y además como se valida el dato enviado.

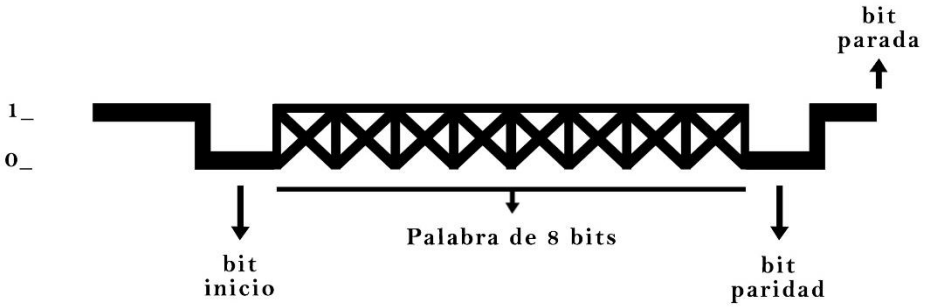


Figura 2.4 Protocolo de comunicación UART

De estas tres condiciones de operación, solo uno es opcional y puede variar de acuerdo con la implementación seleccionada. Las otros dos son contantes y tienen que ser utilizadas siempre que se desea transmitir o recibir información a través del protocolo UART. Estas tres condiciones se describen a continuación:

Bit de Inicio: Es el bit que se antepone al envío de la palabra y se denota con un cero lógico.

Bit de Parada: Es el bit que se escribe al final de la secuencia de la palabra y se denota con un uno lógico.

Bit de Paridad: Es un bit opcional, se puede utilizar paridad par o paridad impar, en caso de ser par se denota con un cero lógico y si es impar con uno lógico. Tal y como se muestra en la Figura 5, este se ubica entre el ultimo bit de la palabra enviada y el bit de parada.

Otro aspecto importante del protocolo es el estado binario cuando no se envían datos, pero existe comunicación con el dispositivo externo. En este caso el puerto serial debe enviar o recibir una cadena de unos, denota que existe comunicación entre dispositivos, pero no existe trasiego de datos.

2.2.2. Módulo de Transmisión UART

Este módulo como su nombre lo indica, se encarga de empaquetar y enviar lo datos a través del puerto serial. El bloque TX toma un dato en paralelo de un tamaño establecido previamente, (Tamaño de la palabra), lo empaqueta según el protocolo y lo envía bit a bit a la velocidad dictaminada por el Baud mencionado en la Tabla 1.

2.2.3. Módulo de Recepción UART

La recepción de la UART funciona de manera opuesta al módulo de transmisión. Esta se encarga de tomar un dato a través del puerto serial, desempaquetarlo y almacenarlo de manera paralela. El bloque RX sigue la misma norma de Baud y tamaño de la palabra mencionada en la Tabla 1.

2.2.4. FIFO

El último de los módulos que componen la base general de una UART es la FIFO, denominada así por sus siglas en inglés. La FIFO es una unidad de almacenamiento en forma de cola, donde el primer dato en entrar es el primer dato en salir.

Este módulo tiene como objetivo evitar un desface entre la información que se envía y la que se recibe. En otras palabras, como los datos entran más rápidos de lo que logran salir, se necesita un sistema que permita almacenar los datos que entran, para luego extraerlos y así evitar perdida de información en el proceso.

La capacidad de almacenamiento de la FIFO depende del diseño y para el caso de la UART esta almacena datos para TX y RX, ambos de forma independiente.

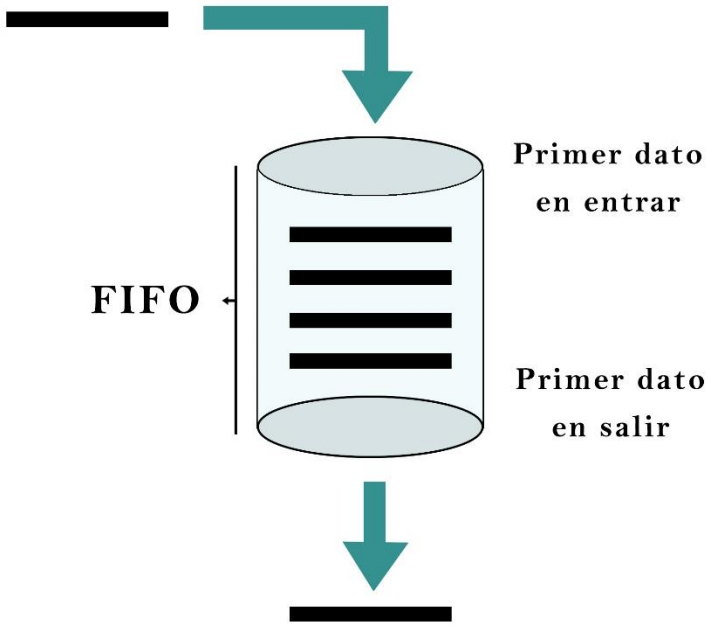


Figura 2.5. Funcionamiento de una FIFO (First Input – First Output)

2.2. Diseño de Circuitos Integrados

El microprocesador diseñado por el ITCR no es más que un circuito integrado, por lo que su desarrollo no es diferente al de cualquier otro CI. Este inicia con la necesidad de resolver un problema, por lo que para consolidar la idea que sufrague la necesidad planteada, se establece en primera instancia un equipo de mercadeo, el cual posee como objetivo estudiar el mercado y definir una lista de características que debe tener el dispositivo.

Posteriormente, un grupo que denominaremos arquitectos se encarga de establecer los parámetros de diseño del dispositivo, que describen como debe ser la funcionalidad del sistema, tomando en cuenta las características del equipo de mercadeo.

Conociendo como debe ser la funcionalidad, es donde entran en escena dos grupos encargados de trabajar en paralelo. Uno de estos dos grupos es el de diseñadores, los cuales se encargan de la implementación del sistema mediante RTL o descripción a través de registros.

El otro grupo en cuestión es el de verificación funcional, este grupo tiene la tarea de generar un ambiente de pruebas y un modelo de referencia que cumpla con las mismas características planteadas por el equipo de arquitectos. EL conjunto de ingenieros de verificación funcional debe de realizar pruebas o test al sistema creado por los diseñadores, con el fin de encontrar errores, informar de estos para su corrección y continuar nuevamente con el ciclo de retroalimentación entre ambas partes, que garantice el comportamiento del sistema desarrollado.

El trabajo en equipo entre verificadores y diseñadores, así como el resto del proceso para desarrollar un CI, se muestra en el siguiente diagrama:

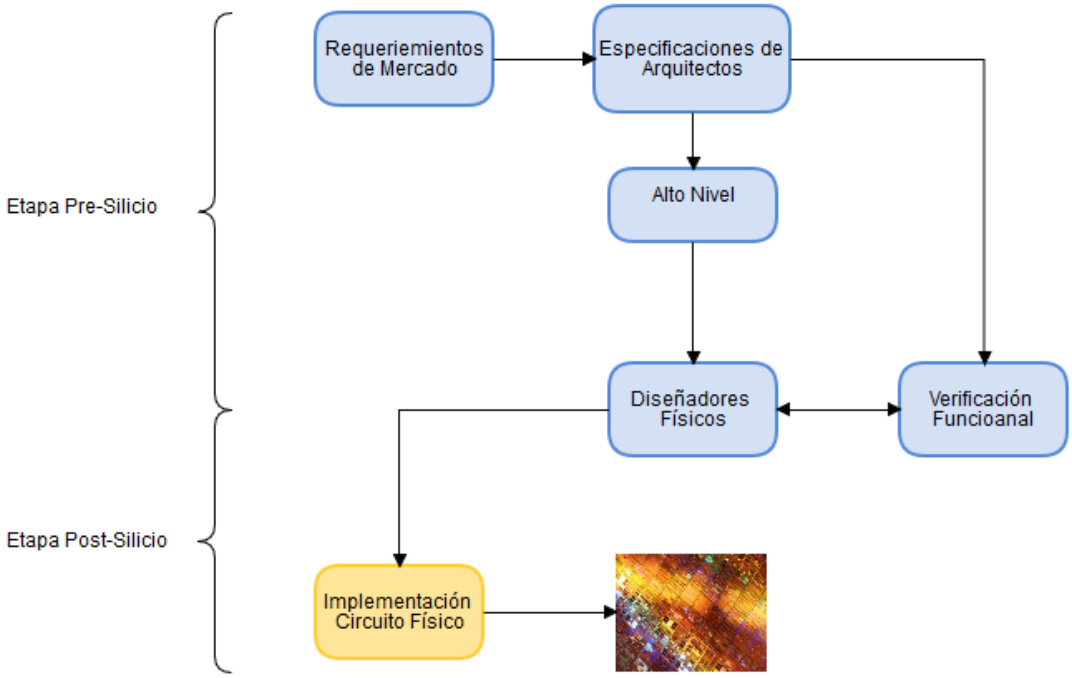


Figura 2.6 Desarrollo de fabricación de un circuito integrado

En el diagrama anterior se visualiza como después de la etapa de diseño y verificación se establece la etapa pos-silicio, que no es más que la implementación de lo creado por el equipo de diseñadores a un nivel físico.

La etapa pos-silicio inicia cuando los diseñadores físicos toman el RTL y realizan lo que se conoce como mapa de nodos, que no es más que reducir el circuito a nivel de transistores y la interconexión entre estos.

La última parte del desarrollo del circuito integrado es imprimir el dispositivo en silicio, esto en una fábrica. El inconveniente de este proceso es que resulta ser muy costoso debido a las tecnologías y al tamaño de los transistores, por lo que encontrar un error a esta altura del proceso es sumamente elevado a nivel monetario. Es debido a este punto que se justifica la necesidad de verificar la funcionalidad del sistema y corregir posibles errores, antes de ser llevado a fábrica para manufactura.

Cabe rescatar que ya con el CI construido se aplican pruebas de funcionalidad. Estas pruebas son elaboradas previamente por el equipo de verificación, grupo que elabora un circuito electrónico paralelo al sistema y que será albergado dentro del circuito integrado al momento de su construcción. Este circuito tiene como finalidad realizar pruebas y verificar el funcionamiento y así poder visualizar posibles errores no contemplados en la etapa pre-silicio. A este tipo de verificación se le conoce como Desing for Testability.

2.4. Especificaciones de la UART

La UART es un bloque encargado de transmitir y recibir datos a través de dos puertos seriales manejados de manera independiente.

Para el caso del microprocesador desarrollado por ITCR, el grupo de arquitectos ha planteado especificaciones especiales para el diseño del transmisor – receptor asíncrono universal. Dentro de estas especificaciones se encuentra la utilización de dos FIFO principales, una para TX y otra para RX, ambas con una capacidad 8 bytes.

Por otra parte, esta unidad cuenta una velocidad (Baud) contante e invariante a lo largo de su funcionamiento. En otras palabras, no se puede modificar la velocidad con que se envían o reciben datos cuando el dispositivo ya se encuentra en operación.

2.4.1. Interfaces

La siguiente imagen muestra las terminales de entada y salida del módulo UART

Figura 2.7. Interfaz UART

2.4.2. Comportamiento de la UART

Módulo Transmisión:

La siguiente figura muestra el comportamiento del módulo, según la variación de estado de sus interfaces.

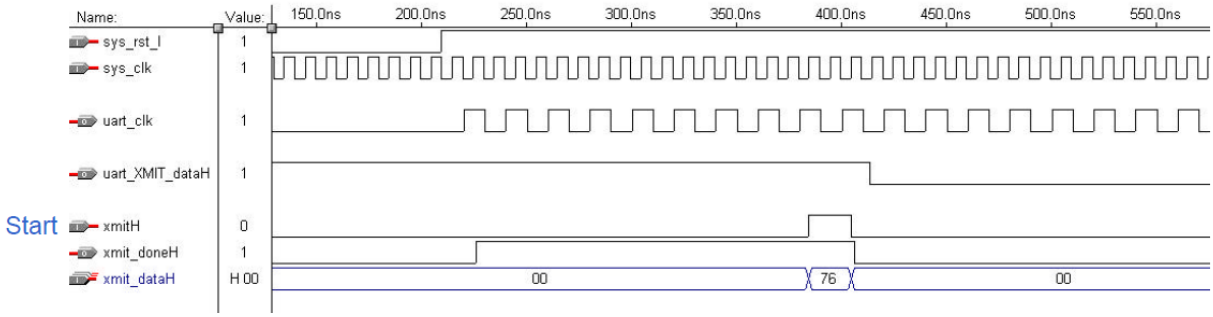


*Figura 2.8 Diagrama de tiempo TX (*J.Lee, 2001*)*

Tal y como se observa en la figura anterior, para que la se logre almacenar un dato en la FIFO de transmisión, se debe escribir un uno lógico en “uart_XMIT” y posteriormente colocar el dato paralelo en buffer “w_data”. En el caso que se complete el almacenamiento de los 8 datos que puede contener la FIFO, se arroja una bandera (tx_FULL), que indica que nuestra cola se encuentra totalmente llena y no puede recibir ningún otro dato.

Por otra parte, en caso de que se complete la transmisión de un dato a través del buffer “uart_XMIT_dataH”, el módulo debe arrojar una bandera en la terminal “xmit_doneH”, la cual indica que la transmisión del dato fue exitosa.

Módulo Recepción:

Según se muestra en el diagrama de la imagen Figura 8, la recepción de la UART consta de mayor cantidad de terminales que controlan la recepción de datos. La siguiente tabla, muestra el comportamiento del boque Rx.

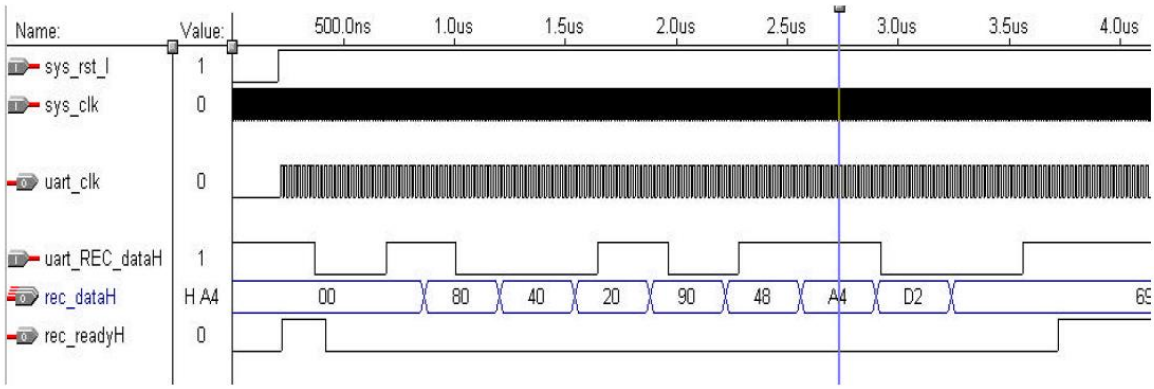


Figura 2.9. Diagrama de tiempo RX (J.Lee, 2001)

Tras la recepción de un dato a través de “uart_REC_dataH”, el sistema debe ser capaz notificar si el dato enviado presenta un error en el bit de paridad “parity_error” y si la recepción fue exitosa “rec_readyH”.

2.5. Conceptos de Verificación

El proceso de verificación funcional toma en cuenta dos conceptos, el factor humano y el Modelo de Reconvergencia (Pacheco, 2013). Esto se debe a que tras el proceso de diseño descrito en el diagrama de la Figura 7, el éxito de la implementación depende de que si el sistema se comparta y ejecuta lo descrito por el grupo de mercadeo y arquitectos.

El inconveniente en el desarrollo del dispositivo surge cuando un diseñador interpreta de manera errónea una especificación de diseño, o se equivoca al momento de la implementación, lo que deriva en un error involuntario y no previsto al describir mediante RTL el circuito integrado.

Indistintamente del error, este se debe a un factor humano, el cual en muchos casos puede ser difícil de diagnosticar, por lo que para mitigar este factor se encuentra el modelo denominado Modelo de Reconvergencia. Este modelo busca que, a partir de una misma especificación, se realicen dos procesos paralelos, uno el diseño del dispositivo y otro la verificación del mismo.

Establecido el Modelo de Reconvergencia, la verificación funcional se puede establecer de diferentes maneras, por lo que a continuación se muestran algunas utilizadas:

Modelado: Consiste en crear un ambiente de verificación mediante la utilización de un lenguaje de más alto nivel que el utilizado para la creación del RTL, este lenguaje puede ser System Verilog, que constituye una unión entre Verilog (Utilizado para la creación del RTL) y C++. Posteriormente se toma ambos sistemas (Ambiente de Verificación y dispositivo a nivel de RTL) y se estimulan con la misma entrada para contrastar sus señales de salida.

Emulación: Consiste en sintetizar el RTL en una FPGA o dispositivo similar. Esto permite estimular el prototipo más rápidamente que una simulación, pero tiene como principal inconveniente las limitaciones físicas de la FPGA, convirtiéndose en una no opción a considerar debido al sistema que se está desarrollando.

Verificación Formal: La verificación formal consiste en desarrollar un algoritmo matemático para comprobar que el código RTL cumple con las especificaciones de diseño. Este tipo de verificación solo es aplicable a módulos pequeños debido a su complejidad de implementación. En caso de aplicar este tipo de verificación a módulos de mayor envergadura, el tiempo de comprobación del código del sistema se verá incrementado.

Dentro de las maneras de implementar un ambiente de verificación, se encuentran diversas formas de generar los estimulas o pruebas necesarias. A continuación, se describen algunas de estas metodologías:

Verificación Caja Negra:

Contexto tomado del análisis de sistemas, funciona de manera tal que se realiza un estímulo al RTL con el fin de recolectar una respuesta. Esta se da solo en una interfaz específica y no se visualiza el código del RTL, por lo de ahí su denominación de caja negra.

Dentro de las ventajas de esta verificación es que se respeta el Modelo de Reconvergencia, al partir de la especificación de los arquitectos y evaluar la funcionalidad del sistema sin conocer la micro arquitectura desarrollada por los diseñadores.

Pruebas aleatorias restringidas:

Anteriormente las pruebas se desarrollaban para generar un estímulo al código de RTL, pero de manera limitada. Estas pruebas se desarrollaban de manera específica a un área, con el fin de activarla y estimularla, pero se veían limitadas al rango de pruebas y alcance del diseñador para generar los estímulos necesarios.

Debido a esta limitación, se optó por desarrollar un sistema de pruebas aleatorias que generen un mayor porcentaje de penetración y abarquen mayor cantidad de área estimulada en el circuito. Este método se logra al diseñar pruebas específicas para cada área y los detalles de las variables se ejecutan de manera aleatorizada y de manera repetida, logrando de esta manera simular cientos de escenarios, ampliando el rango de cobertura de la prueba.

Banco de Pruebas Auto Verificado:

Consiste en implementar un modelo de referencia con la misma funcionalidad del código de diseño descrito mediante RTL. Es precisamente en este método que se elimina la idea tradicional de pruebas para la funcionalidad del sistema y surge el entorno de un ambiente de verificación en paralelo al código RTL.

La idea del Banco de Pruebas es estimular el sistema RTL de manera constante para validar que su comportamiento se ajuste a la especificación del grupo de arquitectos.

Modelado a Nivel de Transistores:

Consiste en un lenguaje con él se comunican los componentes de la banca de pruebas. El TLM. denominado así por sus siglas en inglés, es el que permite pasar de un nivel de lenguaje a otro, logrando mantener compatible el sistema.

El claro ejemplo del funcionamiento del TLM, es cuando un sistema de pruebas envía una señal desde un lenguaje de más alto nivel como System Verilog y lo convierte a nivel de transistores para que sea leído por el DUT y este a su vez envía una respuesta que pasa por el mismo proceso solo que de manera inversa.

Capítulo 3 Metodología

3.1. Metodología UVM (Universal Verification Metology)

La Metodología de Verificación Universal (UVM por sus siglas en inglés), fue desarrollada por Acellera Systems Initiative (Rosenberg, 2010) y creada con la intención de establecer un estándar en una industria con una alta variabilidad de banca de pruebas.

La idea de establecer un estándar en la industria surge debido a la necesidad de homogenizar los diferentes módulos utilizados como banca de pruebas, por compañías como INTEL, AMD o Hewlett Packard. La idea de homogenizar es permitir la reutilización de los módulos, indistintamente si estos son creados por diferentes empresas.

3.1.1. Ambiente UVM

La estructura UVM consiste en un conjunto de bloques que interactúan entre sí. Cada uno de estos bloques se les denomina UVC (Universal Verification Component por sus siglas en inglés) y realizan una función específica al momento de la verificación (Rosenberg, S y Meade, K. A. 2010).

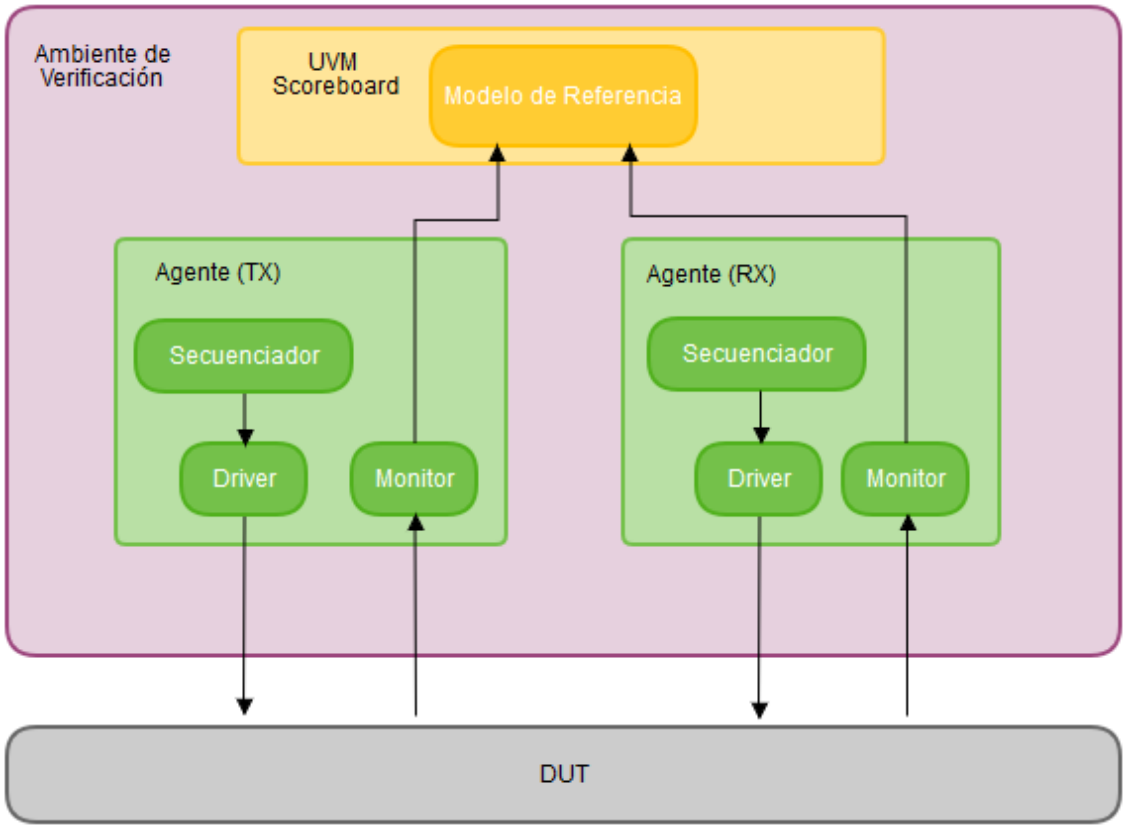


Figura 3.1. Ambiente de verificación metodología UVM

Al módulo de mayor de jerarquía se le denomina ambiente UVM. Este se encuentra dentro de un archivo de Sistem Verilog, el cual es programado con diferentes pruebas que generan un estímulo en partes específicas del DUT, provocando estrés en el sistema.

El ambiente UVM está conformado por los siguientes bloques:

- Dos módulos Agentes (Uno para Rx y otro para Tx).
- Un módulo Scorecoard.

La constitución del ambiente UVM se visualiza en la gráfica de la Figura 8, donde se aprecia la jerarquía de bloques funcionales, así como su flujo de señales.

3.1.2. Agente UVM

Los módulos Agentes son los bloques bases del ambiente UVM. Cada uno de estos Agentes se encuentra formado por tres bloques, tal y como se muestra en la Figura 3.1 (bloques en color verde).

Los bloques visualizados en la Figura 8, se encargan básicamente generar las variables necesarias para estimular el dispositivo en RTL, convertirlas de un lenguaje alto nivel a uno de transistores (lenguaje máquina) y monitorear las señales de entrada y salida del sistema. Estos módulos generales conforman el Agente y son los que normalmente se reutilizan para implementar diferentes testbench.

Un aspecto por rescatar es que cada agente debe recibir tres parámetros de entrada, el primero consiste en una identificación de prueba, seguido por una bandera TX de entrada y finalmente una bandera RX.

El accionar de estos tres parámetros inicia cuando el entorno UVM debe pasar el ID (identificación de prueba) al agente TX UVM para que este conozca la prueba que debe ejecutar. Este proceso activará la bandera TX de entrada, indicando que el agente UVM se encuentra activo y controlará el DUT, al mismo tiempo que se ejecuta la bandera RX para activar el UVM como controlador de salida del DUT. La siguiente tabla de verdad muestra esta configuración:

Tabla 3.1. Configuración Agente UVM

Bandera RX	**Bandera TX**	**Funcionalidad**
0	0	Agente UVM OFF: • Driver OFF. • Monitor OFF. • Secuenciador OFF.
0	1	Agente UVM Modo TX: • Driver **ON**. • Monitor **ON**. • Secuenciador **ON**.
1	0	Agente UVM Modo RX: Driver OFF. • Monitor **ON**. • Secuenciador OFF.
1	1	Estado Invalido: • Driver OFF. • Monitor OFF. • Secuenciador OFF.

3.1.3. Secuenciador

Es un bloque en el que se generan todas las combinaciones necesarias para estimular el DUT en busca de algún fallo en su funcionamiento. Estas combinaciones pueden ser aleatorias o previamente configuradas y se envía de manera secuencial a través de la interfaz del Driver.

La funcionalidad del Secuenciador se explica en la Tabla 3.1, donde se observa que este bloque solo se ejecuta cuando la bandera transmisión (TX) se encuentra activa, situación que traza con el objetivo del módulo de enviar información al DUT para corroborar su comportamiento.

3.1.4. Driver

El driver es un bloque que forma parte del Agente y funciona como interfaz entre el DUT y el ambiente de verificación. La configuración del Driver es de transmisión TX, por lo que su labor es transformar las señales generadas por el Secuenciador y transmitirlas al dispositivo en RTL.

Para generar los estímulos al DUT el Driver debe de recibir dos señales, una el estímulo proveniente del Secuenciador y una bandera que indica si el bloque debe estar en modo escritura o lectura (TX o RX), tal y como se aprecia en la Tabla 3.1.

3.1.5. Monitor

El Monitor es el último bloque Agente y su propósito es observar las señales que se encuentran conectadas entre sus terminales. Estas señales no son modificadas por el monitor ya que actúa como elemento pasivo, por lo que solo funciona de interfaz entre los estímulos generados por el secuenciador, la información enviada por el DUT y el Scoreboard.

Cabe rescatar que el funcionamiento del Monitor también se encuentra en la Tabla 2, donde se aprecia que este bloque se encuentra activo tanto para TX como RX.

Un aspecto importante de este módulo es que recolecta la información proveniente del DUT. Esta información es la que responde al estímulo generado por el Secuenciador, por lo que esta es transmitida directamente al Scoreboard para ser contrastada con la enviada por el Modelo de Referencia.

3.1.6. Scoreboard

El Scoreboard es un bloque albergado dentro del Ambiente de Verificación y cuyo funcionamiento se basa en dictaminar el acierto o desacierto de la prueba ejecutada al DUT. Este módulo dictamina el acierto de la prueba contrastando las señales de salida tanto del Modelo de referencia como del DUT. Este asocie de las señales se realiza al mismo tiempo, por lo que si no presentan el mismo resultado se arrojara una bandera que indique el fallo de la prueba, o en caso contrario se expresara su ejecución satisfactoria.

Este bloque posee comunicación con los dos Agentes, uno configurado como TX y el otro como RX, además es el encardo de contener el Modelo de Referencia y generar el reporte que indica el acierto o desacierto de la prueba, así como el lugar o la sucesión que provoco el fallo.

3.1.7. Modelo de Referencia

El Modelo de Referencia se encuentra contenido dentro del Scoreboard y representa el módulo de mayor complejidad en el ambiente de verificación.

Este bloque es diseñado con las mismas especificaciones que el DUT, por lo que su funcionamiento debe ser el mismo si este no posee fallos. La implementación del Modelo de Referencia es diferente al DUT, ya que este se hace utilizando un lenguaje de más alto nivel, facilitado así su desarrollo por parte del equipo verificación a cargo.

3.2. Metodología OVM

La Metodología de Verificación Abierta (OVM por sus siglas en ingles), consiste en una biblioteca de objetos y procedimientos que buscan generar un estímulo en el DUT. Al igual que el UVM, OVM trabaja en lenguajes como SystemVerilog o SystemC y permite una creación de pruebas dirigidas a nivel de transaciones y cobertura funcional.

Cabe destacar, OVM contribuyo al desarrollo de la Metodología Universal Verificación, por lo que se considera una de sus predecesoras, aunque en la actualidad no se utiliza en el mercado, por lo que no fue contemplada para la implementación del ambiente de verificación (Aldec, 2015).

3.3. Metodología VMM

La Metodología Manual de Verificación (VMM), fue el primer conjunto de pruebas prácticas y exitosas realizadas por la industria, para la creación de entornos de verificación reutilizables.

Creado por Synopsys, uno de los principales defensores de SystemVerilog, VMM utiliza funciones de programación orientadas a creación de objetos, restricciones y cobertura funcional para la creación de entornos de verificación. Se debe rescatar que al igual que OVM, VMM contribuyó al desarrollo UVM, por lo que hoy día se encuentra obsoleta y descartada en la industria (Aldec, 2015).

3.4. Plan de Verificación

El plan de verificación se desarrolla con la metodología UVM y consta de tres secciones, la primera consiste en desarrollar el ambiente de verificación, casos de prueba y la otra cobertura funcional.

Desarrollar el ambiente de verificación, consiste en construir los bloques que constituyen la metodología UVM, mientras que los casos de prueba consisten en generar los estímulos de entrada al DUT, mediante secuencias que lleven al dispositivo a zonas de funcionamiento no pensadas o analizadas durante su etapa de implementación. Cabe destacar que, al ser una aleatorización restringida, se debe evaluar el alcance que generan las pruebas generadas, por lo que en segunda instancia del plan de verificación de debe evaluar la cobertura funcional.

Tabla 3.2. Datos por corroborar en el UVM

	Funcionalidad
1	Corroborar los bit de paridad, inicio y parada
2	Corroborar el funcionamiento de la FIFO del UART
3	Corroborar la trama de 8 bits.
4	Corroborar la velocidad del reloj de la UART

La aleatoriedad de los valores anteriores busca evaluar el funcionamiento global del DUT, abarcando la sección de transmisión y la recepción. La limitante que poseen estas pruebas consiste en el hecho de que se necesitan una cantidad de iteraciones que permita generar las secuencias que desencadenen un rango de cobertura total para el análisis de funcionamiento del DUT.

Es debido al alcance de las pruebas sobre del DUT, es que en la tercera etapa del plan de verificación consiste en determinar el porcentaje de estados que recorre el DUT. La idea fundamental en conocer el verdadero alcance de las pruebas realizadas y en caso de ser necesario, crear nuevas pruebas que permitan incrementar el porcentaje de cobertura.

Capítulo 4 Implementación

4.1. Ambiente de Verificación

El ambiente de verificación se basa en la metodología UVM, esta se compone de dos bloques principales, el agente y el scoreboard. Para el caso de este proyecto, se utilizó como referencia el ambiente de verificación de una memoria desarrollada en Verigication Guide e implementada en el servidor de EDA Playground. Cabe destacar que ambas páginas y servidores son de uso libre y gratuito.

El ejemplo de la memoria contiene 12 bloques funcionales que componen el ambiente de verificación.

UART_Seq_Item: Este bloque se encarga de generar las variables que se envía al DUT y al modelo de referencia. Estas instancias son de control y datos por lo que se generan de manera aleatoria e limitadas a valores reales que pueden ser enviados al sistema.

UART_Sequence: Se encarga de seleccionar la secuencia que se va a ejecutar, solicitar los datos aleatorios del bloque “UART_Seq_Item” y esperar una nueva solicitud de envío de datos.

UART_Sequencer: Es el encargado de generar el objeto que define la secuencia que se va implementar.

UART_Driver: Su función de convertir los datos de alto nivel, a nivel de señales eléctricas. Este módulo se divide en dos secciones, una se encarga de la transmisión y la otra de la recepción de datos. Para el caso de TX, el bloque toma lo datos provenientes del secuenciador y los envía al DUT, mientras que para la sección de RX, el Driver adquiere las variables provenientes del DUT y las transmite al Scoreboard. El funcionamiento de este módulo se describe mediante el siguiente diagrama de flujo:

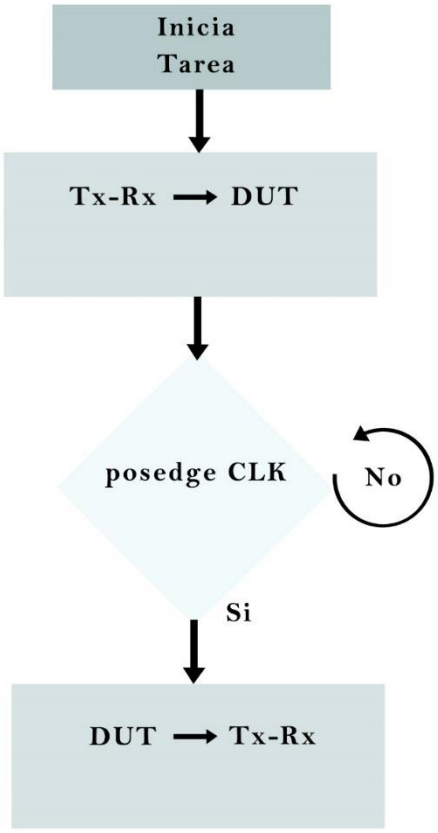


Figura 4.1. Diagrama del Driver del UVM

Como se observa en el diagrama anterior, este módulo posee un flanco de reloj de diferencia entre las señales que se envía al DUT y las que se reciben. Esto es debido a que se debe generar un retardo de tiempo al leer las señales provenientes del DUT, para darle espacio a este de que pueda enviar una respuesta tras el estímulo realizado. Este tiempo de retardo nos ayuda a equipar en un mismo lapso los datos provenientes del DUT y los generados por el modelo de referencia, para ser comparadas en igualdad de condiciones por el Scoreboard.

UART_Monitor: Este bloque corresponde al módulo del Monitor, elemento pasivo que conforma el sistema, donde su función es servir como visualizador de los datos provenientes del driver y que se dirigen al Scoreboard. Cabe destacar que al igual que el driver, existe un retardo para los datos provenientes del DUT. Este se compone de medio periodo y se realiza con la intención de darle tiempo al Driver de procesar los datos provenientes del DUT, para que lleguen en fase al Scoreboard.

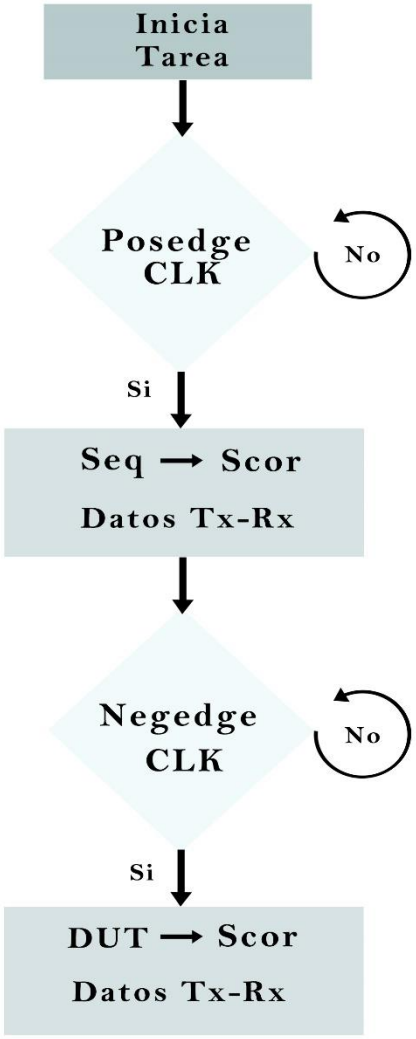


Figura 4.2. Diagrama del Monitor UVM

UART_Agent: Corresponde a la instanciación de los bloques que constituyen el Agente. Estos módulos son el UART_Seq_Item, UART_Sequence, UART_Sequence, que corresponde al Secunciador y los restantes como lo es UART_Driver y UART_Monitor.

UART_Scoreboard: Es uno de los sistemas más complejos del ambiente de verificación, se compone de dos secciones, una que contiene el modelo de referencia y la otra que posee las condiciones que evalúan si los datos contrastados entre Modelo de Referencia y DUT son los mismos. Cabe destacar que todo el módulo posee una cuenta de reloj que establece los baudios con que funciona el sistema, por lo que para ciclo completado de la constante CLK, se inicializan las dos condiciones que conforman el Scoreboard. Este bloque se detallará más adelante debido a su complejidad e importancia.

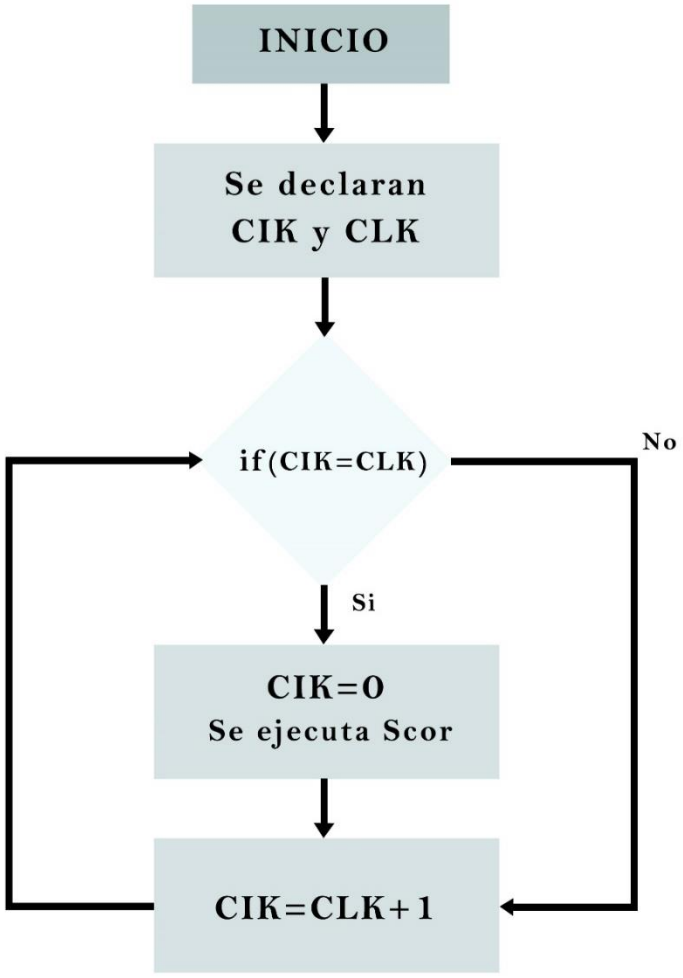


Figura 4.3. Diagrama de Reloj que controla el Modelo de Referencia

UART_env: Corresponde a la instanciación de los módulos UART_Scoreboard y UART_Agent.

UART_Base_test: No es más que un bloque que evalúa el estado del test (PASS o FAIL). Este se determina mediante una condición, la cual depende de que si la variable “uvm_error” se encuntra activada dentro del Scoreboard. Adicionalmente este módulo incluye a UART_env.

UART_Interface: Establece la interfase que se conecta al DUT. En otras palabras, todas las entradas y salidas que corresponden al ambiente de verificación se generan en ente bloque, clasificándose según módulo en la que se encuentran.

UART_Rx_TX_Test: Este bloque contiene las secuencias que se pueden ejecutar para el testbench. Más adelante se explica con mayor detalle los casos de prueba que contiene este módulo.

Tesbench: Modúlo Top, este contiene todo el ambiente de verificación, por lo que en él se instancian los bloques UART_Interface, UART_Rx_TX_Test y UART_Base_test

4.2. Creación de Variables Aleatorias

Para la creación de los datos que estimulan el DUT, se utiliza la función denominada “rand”, al momento de la declaración de la variable. Esta función genera un dato aleatorio cada vez que se llama al bloque que la contiene, arrojando diferentes combinaciones de datos. A continuación, se describe una tabla que contiene las variables generadas en UART_Seq_Item:

Tabla 4.1. Datos aleatorios generados

Declaración	Variable	Sección	Función
rand bit	ld_tx_data	TX	Simula la petición de cargar un dato a la FIFO de TX
rand bit	tx_data	TX	Genera un dato de 8 bits para TX
Bit	tx_out	TX	Salida serial TX
Bit	tx_empty	TX	Indica si la FIFO TX se encuentra llena
rand bit	rd_uart_R	RX	Simula la generación de un dato serial para ser leído por RX
rand bit	uart_REC_dataH	RX	Petición de enviar un dato a través del puerto serial.
Bit	rx_out	RX	Salida del dato en paralelo de RX
Bit	pndng_R	RX	Bandera que indica que se está a la espera de recibir un dato
Bit	tx_empty	TX	Indica que la FIFO RX está llena.

4.3. Casos de Prueba

Existen tres casos de prueba implementados en el ambiente de verificación, el primero corresponde a la sección de transmisión, el segundo a recepción y por último una mezcla de ambos.

Tabla 4.2 Casos de prueba

Prueba	Función	Descripción
Nombre: Tx_sequence	Verifica el módulo de transmisión de la UART.	Estimula solo la sección de transmisión de la UART, desactivando la recepción.
Nombre: Rx_sequence	Verifica el módulo de recepción de la UART.	Estimula solo la sección de recepción de la UART, desactivando la transmisión.
Nombre: Rx_TX_sequence	Verifica el módulo de transmisión y recepción de manera simultánea.	Genera estímulos aleatorios simultáneos para TX y RX.

Prueba Transmisión: Consiste en generar datos aleatorios dirigidos a la sección de TX, por lo que se desactiva en su totalidad RX de la UART. A continuación, se representa un diagrama de flujo con el funcionamiento de esta prueba:

Prueba Recepción: Su funcionamiento se basa en estimular mediante datos aleatorios la sección RX, por lo que se desactiva en su totalidad TX.

Prueba Transmisión - Recepción: Se basa fundamentalmente en estimular de forma paralela RX y TX, aleatorizando datos en secciones de tiempo, en otras palabras, en momentos se desactiva RX y solo funcionan TX o viceversa. También durante la prueba se activan los dos al mismo tiempo, todo con el fin de verificar el funcionamiento del UART y la independencia de sus módulos.

4.4. Modelo de Referencia

Se encuentra contenido dentro del Scoreboard y para su implementación se siguieron las especificaciones de diseño detalladas en la sección 3.1.

El modelo referencia se clasifica en dos grandes secciones, una encargada de la transmisión y otra de la recepción de datos, ambas ejecutadas de forma paralela.

Transmisión:

Esta sección se subdivide en tres partes, las cuales se ejecutan de manera lineal y entre todas conforman la transmisión de información.

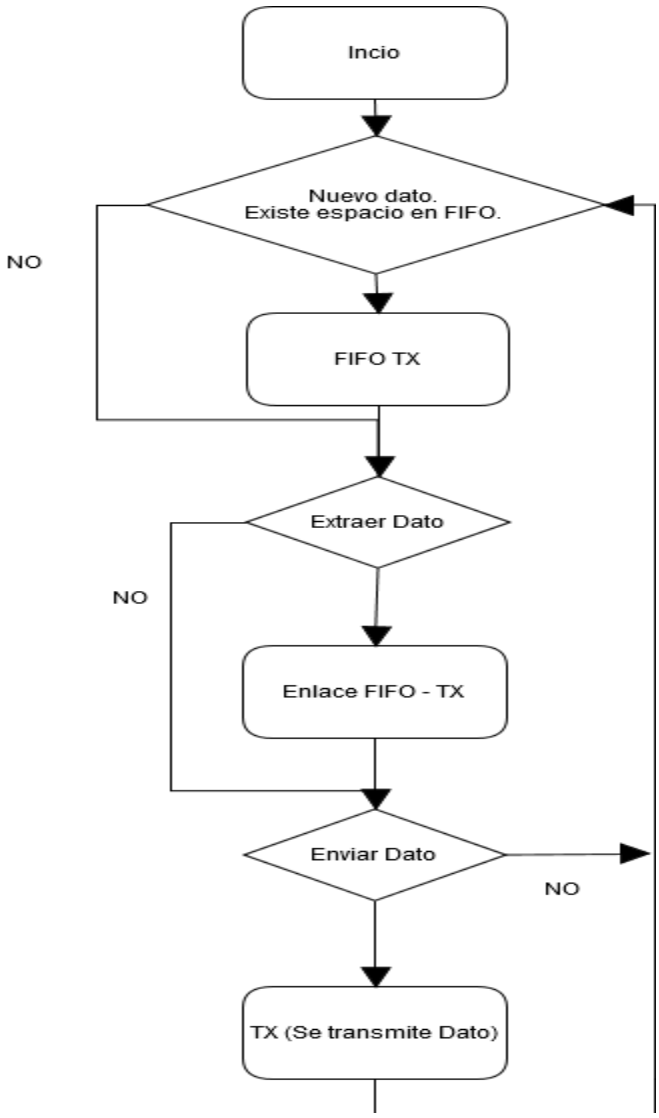


Figura 4.4 Diagrama módulo Transmisión

FIFO TX: Se encarga de almacenar la información proveniente del puerto paralelo. Posee una capacidad de 8 bytes y se activa cuando la variable ld_TX se encuentra activa, informando de esta manera que cargue un nuevo dato. La carga de este dato está condicionada a que si la FIFO se encuentra llena por lo que, en caso contrario, será almacenado en el lugar que indique la variable de cuenta ubicada dentro de la FIFO.

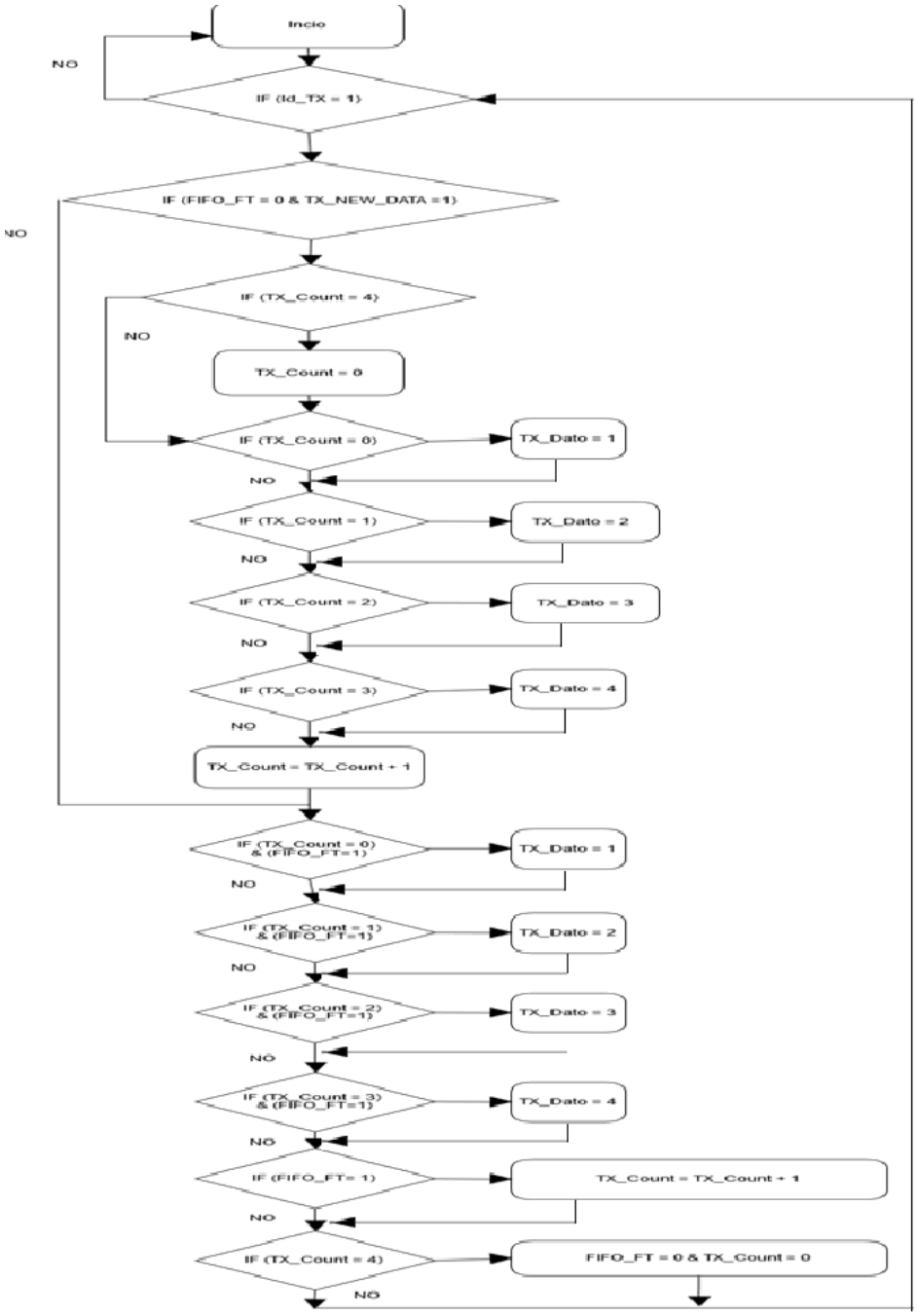


Figura 4.5 FIFO – TX

Enlace FIFO – TX: Su objetivo es servir de enlace entre la FIFO y el bloque TX. Su funcionamiento consiste en extraer el dato de la FIFO, categorizarlo en un registro como actual, para que este sea leído por TX. Cada vez que se completa el ciclo de TX, cambia el estado de una variable, la cual le informa al Enlace FIFO – TX, que debe saltar en la cuenta y tomar un nuevo dato y categorizarlo como actual.

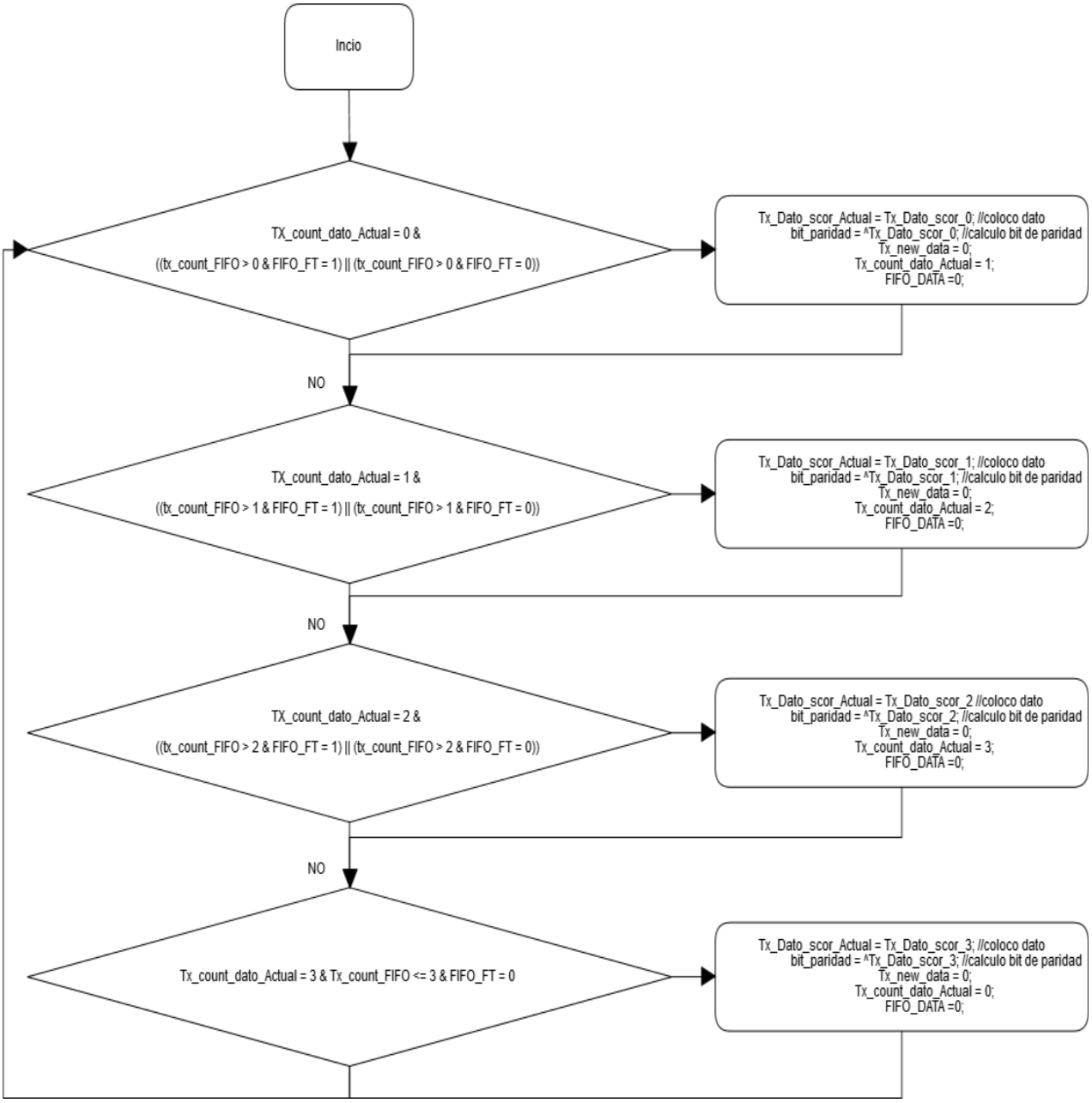


Figura 4.6 Diagrama Enlace FIFO - TX

TX: Es el encargado de convertir el dato paralelo que se encuentra categorizado como actual y transmitirlo de forma serial en la terminal de salida. Adicionalmente, empaqueta la trama bits según protocolo UART, asignando bit de inicio, bit de parada y el bit de paridad par.

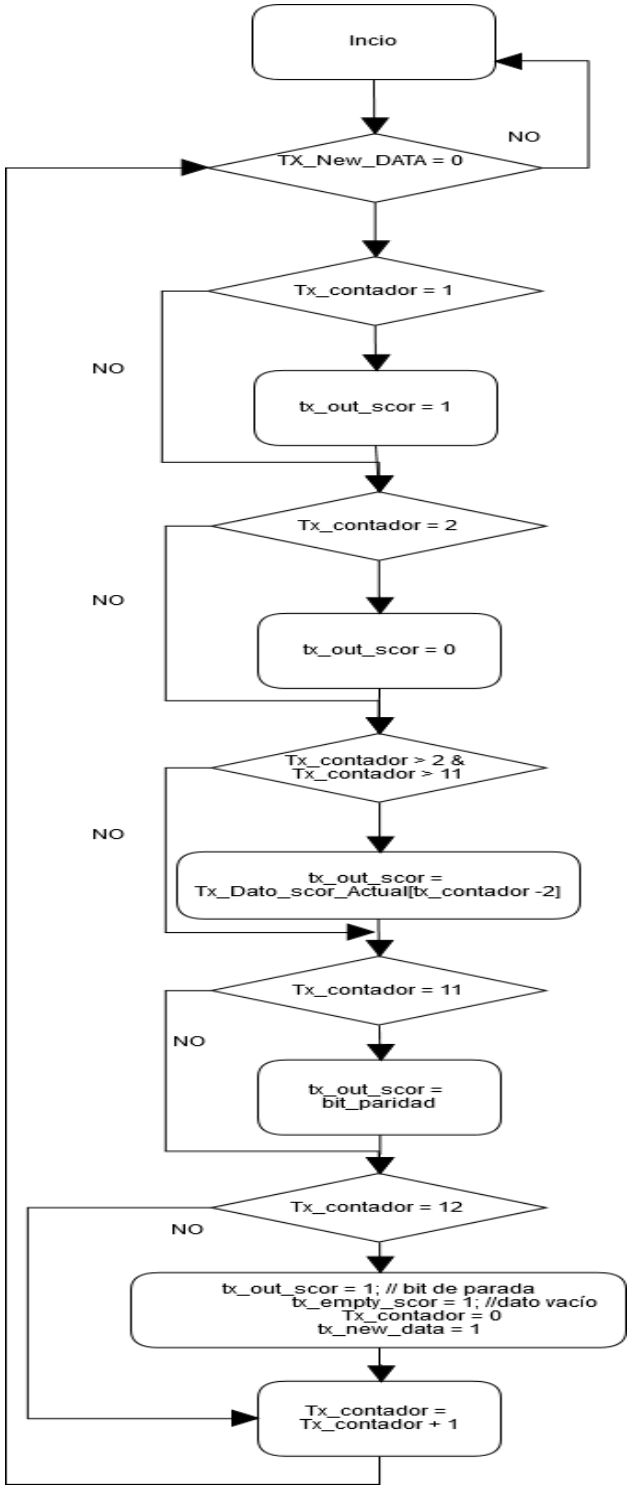


Figura 4.7 Diagrama TX

Recepción:

Al igual que la transmisión se subdivide en tres secciones, las cuales se describen a continuación:

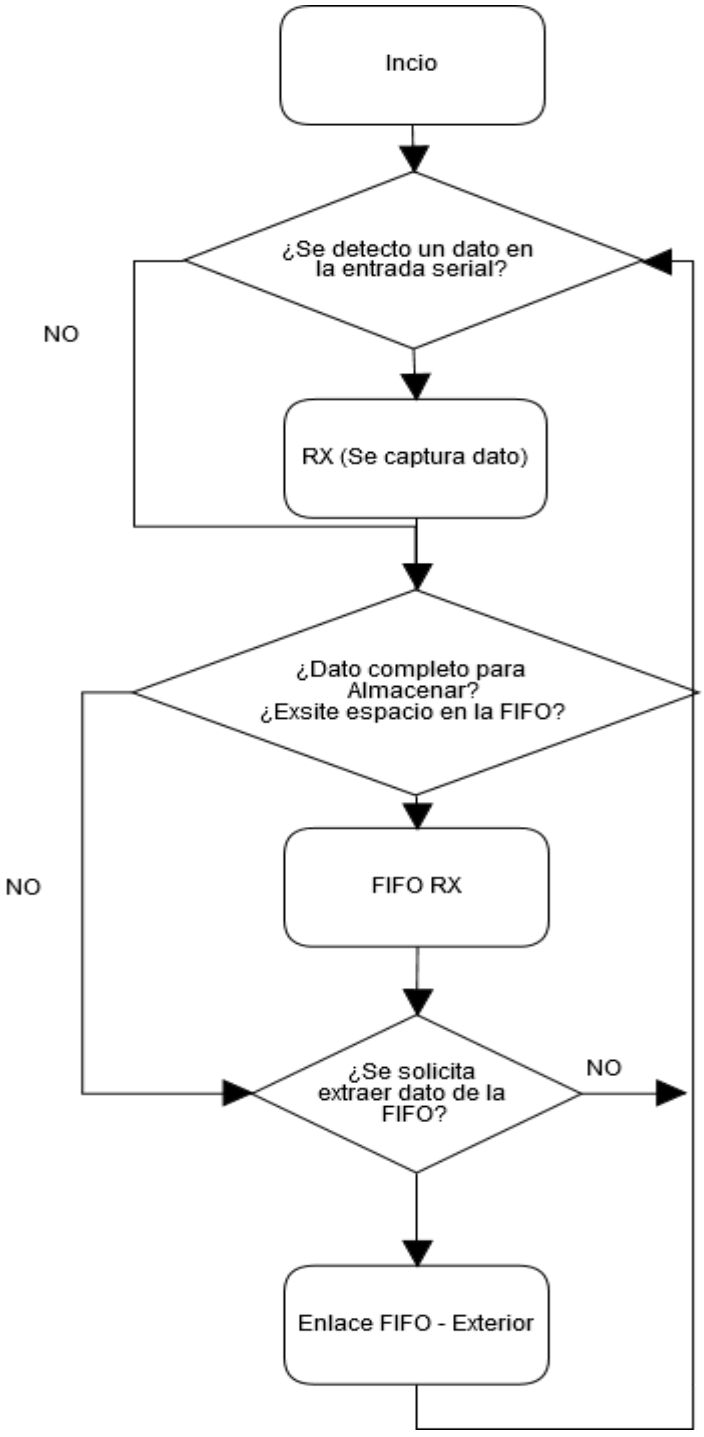


Figura 4.8 Diagrama Recepción

RX: Su función es recolectar los datos a través del puerto serial. Este bloque se activa cuando se detecta el bit de inicio, iniciando la ejecución de recolección de información y almacenándolo bit a bit dentro de un registro temporal de 8 bits.

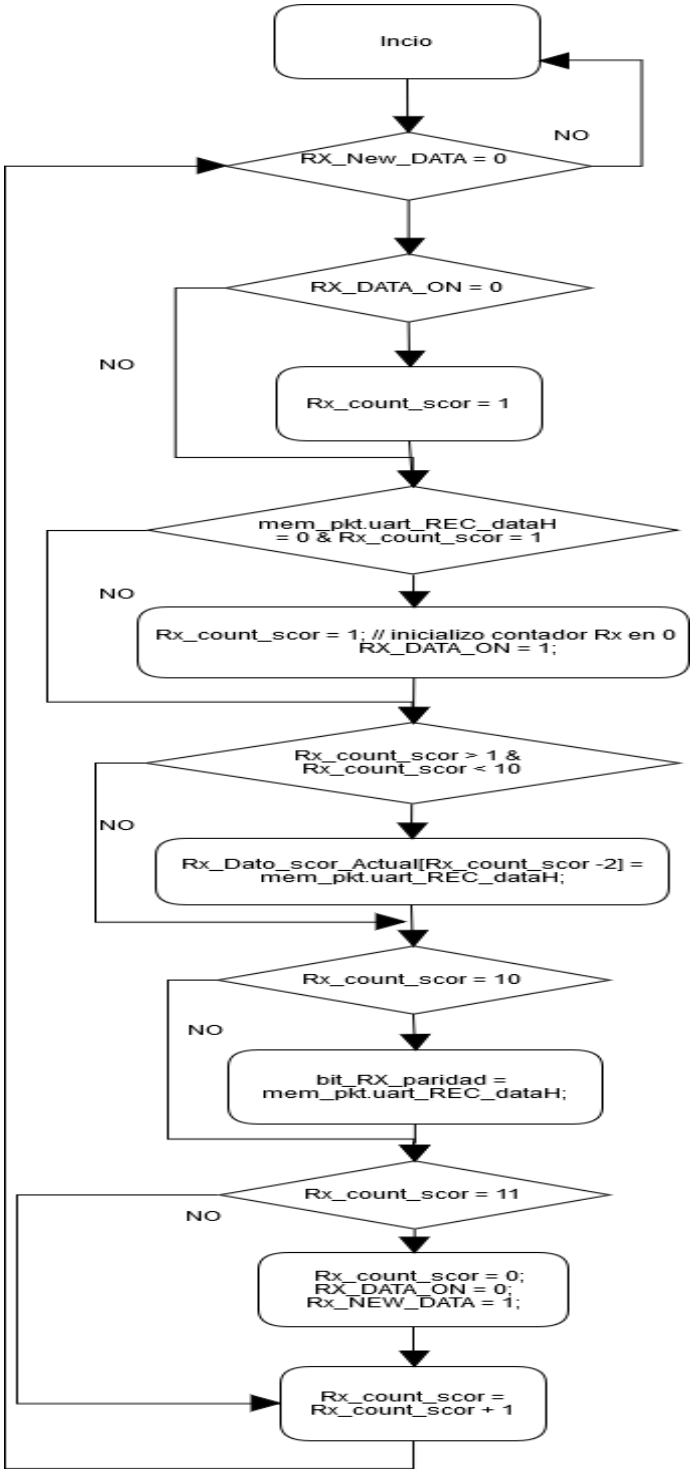


Figura 4.9 Módulo RX

FIFO RX: Almacena el dato recolectado por RX en la dirección que especifica el contador que lleva interno la FIFO. Al igual que la FIFO TX, esta posee una capacidad de 8 bytes y posee una bandera que informa cuando esta se encuentra totalmente llena, impidiendo que sobre escriba un dato que aún no se ha exportado de la UART.

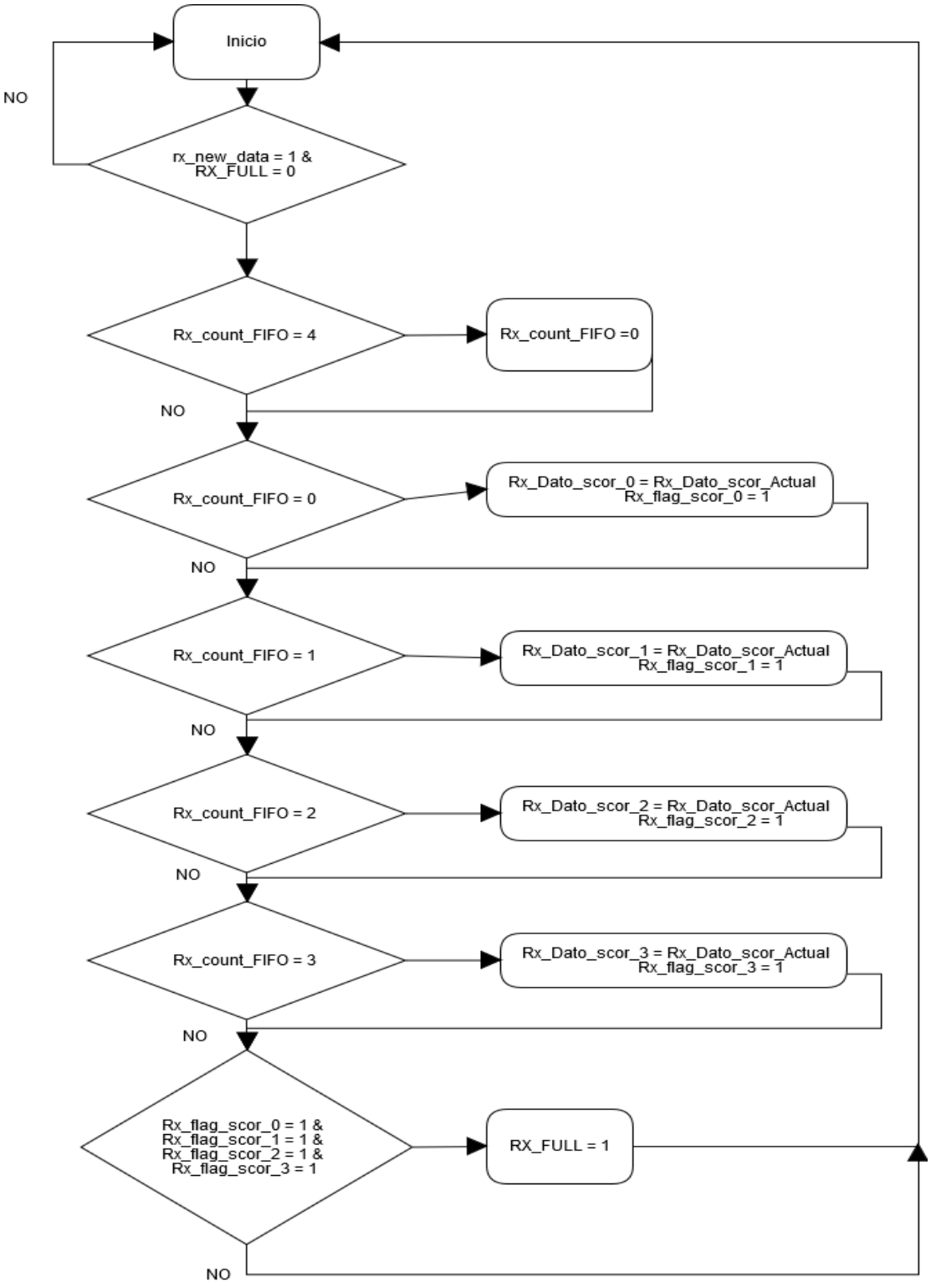


Figura 4.10 FIFO RX

Enlace FIFO – Exterior: Este bloque se encarga de extraer la información almacenada en la FIFO y colocarla en el puerto paralelo de salida. Su funcionamiento es mediante un contador que establece la trama próxima en salir, con la condición de que exista un dato en la FIFO que extraer. Esta condición se conoce mediante una bandera, la cual es levantada para cada uno de los espacios de memoria que la constituye.

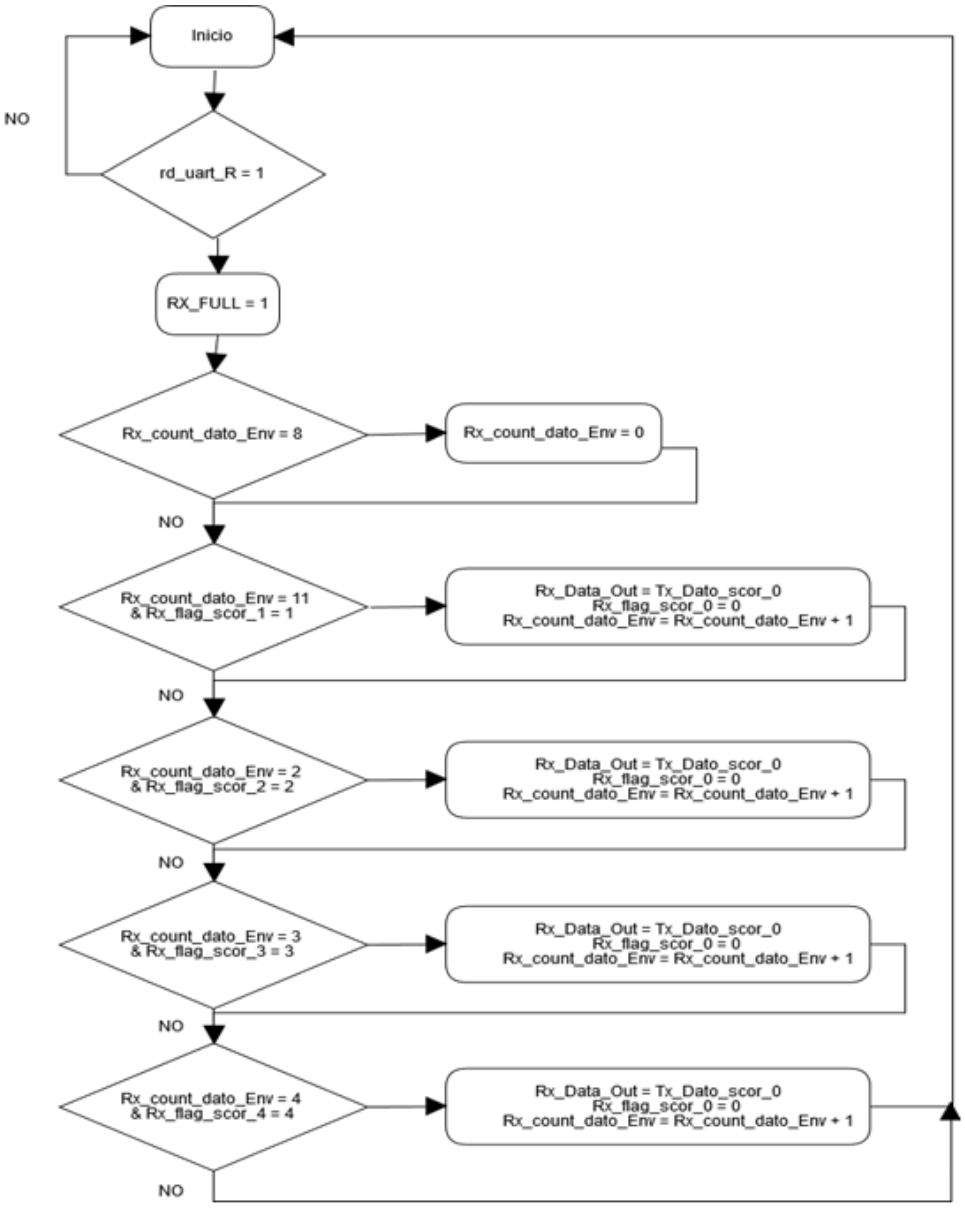


Figura 4.11 Enlace FIFO – Exterior

4.5. Score

Es la última sección del scoreboard, su objetivo es contrastar los datos de salida arrojados por el modelo de referencia, con la información proveniente del DUT. Esta parte trabaja mediante una serie de condiciones que evalúa bit por bit el funcionamiento del DUT y en caso de encontrar una irregularidad, este arroja una bandera de error en el sistema, indicando cual fue y en que iteración se realizó.

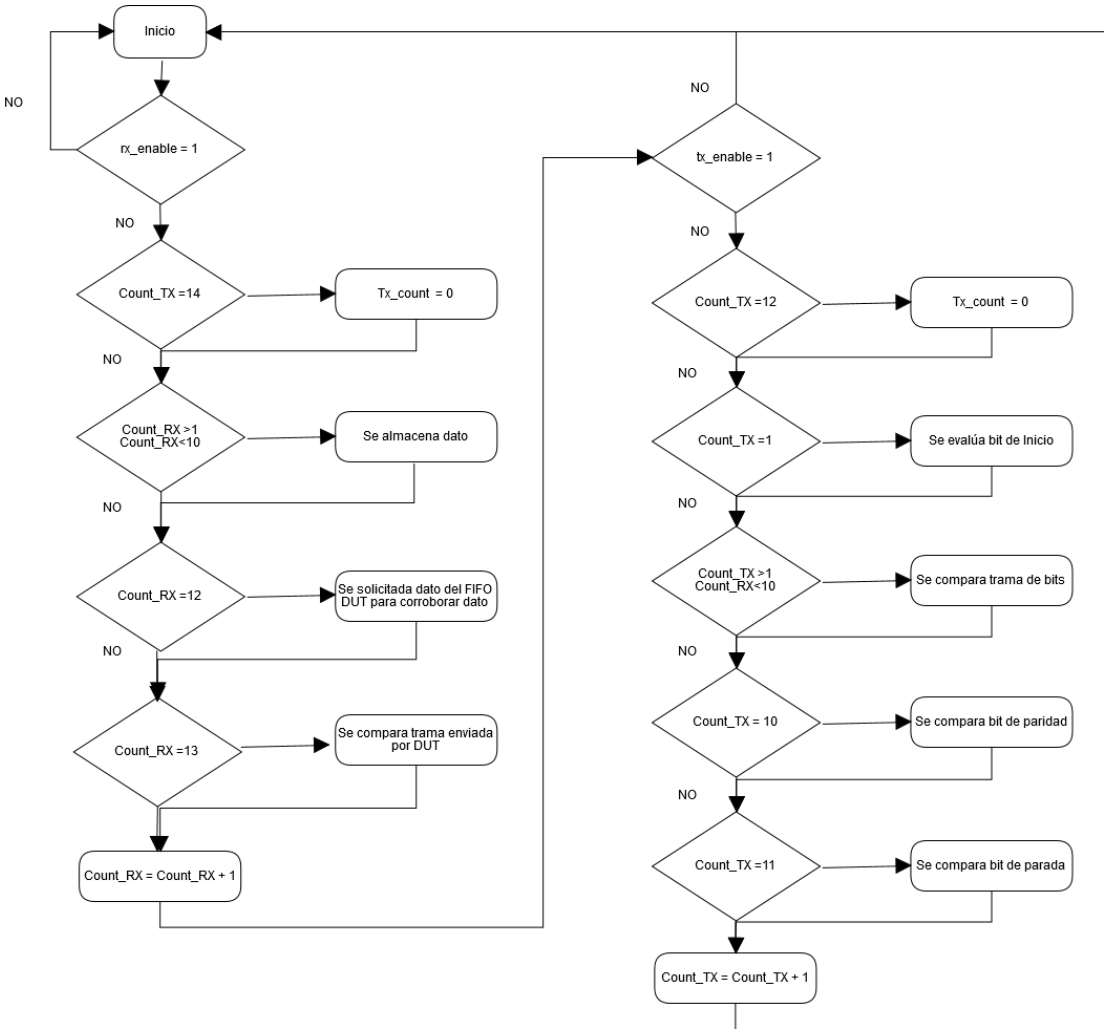


Figura 4.12 Diagrama Score

Capítulo 5 Resultados y Análisis

5.1. Puntos de Cobertura

Tabla 5.1 Puntos de Cobertura

Punto de Cobertura	Función	Iteraciones	Porcentaje
tx_data	Evalúa la cantidad de iteraciones necesaria para generar la todas las combinaciones de datos para TX.	2000	100%
ld_tx_data	Evalúa la cantidad de iteraciones para abarcar los dos estados para carga un dato.	2000	100%
Cruzado ld_tx_data vs tx_data	Evalúa las iteraciones necesarias para generar carga de dato y todas las combinaciones de datos en TX.	2000	98,04%
rd_uart_R	Evalúa la cantidad de iteraciones para abarcar los dos estados necesarios para enviar un dato por el puerto serial.	1000	100%
rd_uart_data	Evalúa la cantidad de iteraciones necesaria para generar la todas las combinaciones de datos para RX en el puerto serial.	1000	100%

Según la tabla anterior se necesitan de al menos 2000 iteraciones para generar las combinaciones de estímulo para la sección de transmisión y recepción de datos cruzados.

5.2. Lista de fallos encontrados

A lo largo del proyecto se encontraron errores que no tenían que ver con una prueba en específico, por lo que o fueron refenciados bajo el formato de fallos visualizados en las tablas inferiores (Tablas 5.2, 5.3 y 5.4), pero que impedían el funcionamiento de los módulos que componían el DUT. Estos errores son:

- No existía un comando “Include” en el encabezado con cada uno de los archivos “.v”, que permitiera incluir dentro de una instanciación los demás documentos.

- En la primera versión entregada al equipo de verificación, el DUT no era capaz de extraer desde su archivo TOP, las constantes generales que necesitaban el resto de bloques para poder inicializarse y funcionar debidamente.

Esto errores fueron corregidos rápidamente, por lo que se procedió a ejecutar las pruebas mencionadas en el capítulo de implementación, arrojando los siguientes resultados:

Tabla 5.2 Fallo #1

Nombre del error	Mal funcionamiento de la bandera TX_FULL
¿Dónde está localizado?	Bloque de diseño, sección transmisión.
Prioridad (Bajo, Medio, Alto)	Medium
¿Cúal prueba encontró el error?	UART_tx_Test
Descripción del error	La FIFO levanta la bandera que indica que la FIFO se encuentra totalmente llena, aun cuando esta no posee datos.
Se corrigió el error	El error fue corregido

Tabla 5.3 Fallo #2

Nombre del error	Mal funcionamiento del reloj de la UART
¿Dónde está localizado?	Bloque de diseño, sección CLK.
Prioridad (Bajo, Medio, Alto)	Alto
¿Cúal prueba encontró el error?	UART_tx_Test
Descripción del error	No se genera el error que indica la velocidad con la que trabaja la UART.
Se corrigió el error	El error fue corregido

Tabla 5.4 Fallo #3

Nombre del error	Erróneo almacenamiento Bit paridad RX
¿Dónde está localizado?	Bloque de diseño, sección recepción.
Prioridad (Bajo, Medio, Alto)	Alto
Cúal prueba encontró el error?	UART_rx_Test
Descripción del error	La sección de recepción invierte el bit de paridad.
Se corrigió el error	El error fue corregido

Capítulo 6 Conclusiones y Recomendaciones

6.1. Conclusiones

Tras el análisis de resultados se concluye que la metodología UVM permite una rápida y ágil implementación de un entorno de pruebas de validación de un circuito integrado, debido a la reutilización de bloques ya utilizados con anterioridad, citado en el capítulo 2.

Por otro lado, los porcentajes mostrados en la Tabla 5.1 demuestran el cumplimiento de los objetivos, ya que se puede corroborar el trasiego de datos tanto de envió y como recepción.

Un aspecto por rescatar es la valía del Modelo de Referencia, ya que se logró contrastar resultados con los obtenidos directamente del DUT, encontrando los fallos mostrados en la sección 5.2.

En lo que respecta a los parámetros de coberturas, se concluye que se puede alcanzar un 100% si se incrementa la cantidad de iteraciones del sistema en más de 1000.

Finalmente, existieron problemas a lo largo de la implementación del ambiente de verificación, inconvenientes como el desface de las señales en el Score y la lectura de datos provenientes del DUT, por lo que se concluye que es mejor realizar la adquisición de información en el flanco negativo, para evitar una lectura prematura de los mismos, además que debe de existir un desfasamiento de un período de reloj en el driver, esto para los datos enviados al Modelo de Referencia, tal y como se puede visualizar en la implementación de estos bloques.

6.2. Recomendaciones

Dentro de las recomendaciones que se pueden señalar, podemos citar que la FIFO de la UART podría trabajar a una velocidad mayor a la del protocolo de comunicación, para que no exista una pérdida de datos al momento de la lectura de información. En otras palabras, la entrada de la FIFO puede capturar datos más rápidamente de lo que salen, funcionando como acople entre el interior y el exterior del microprocesador.

Por otro lado debido a que el tamaño de la FIFO es cambiante, se puede generar una programación con variables dinámicas donde estas dependan de una constante declarada en el encabezado del código. Adicionalmente, se recomienda dividir en archivos separados las secciones que componen el Modelo de Referencia, debido a su extensión

Tal y como se observó en el capítulo 4, no es necesaria la implementación de dos módulos agentes por separado, ya que estos pueden estar constituidos en un solo bloque debidamente clasificada la sección de lectura y escritura.

Uno de los principales inconvenientes en el desarrollo del proyecto, fue la limitante del servidor en el que se trabajó, por lo que se recomienda utilizar otro servidor u aplicación que posea una menor limitante de información mostrada.

Por último, como recomendación final, se cita que es mejor trabajar la mayor cantidad de código en alto nivel (C++) y no tanto en Verilog, para acelerar el procesamiento de datos por parte del servidor, tal y como se visualizó en la implementación del Modelo de referencia y la creación de variables aleatorios.

Capítulo 7 Referencias Bibliográficas

Aldec, (2015). *OVM/UVM for FPGAs:The End of Burn and Churn*. Versión Digital.

Bruce, W. Goss, J. Roesner, W. (2005). *Comprehensive Functional Verifiction:The Complete Industry Cycle.* Elseiver: USA.

Cadence (2012). UVM e reference ow user guide, version 1.1. Disponible en:
http://forums.accellera.org/_les/_le/70-uvm-reference-ow-version-11/.

C. Spear. (2006). SystemVerilog for Verification: *A Guide to Learning the Testbench Language Features*, Springer.

Díaz, A. (2015). *Microprocesadores para aplicaciones*. Universidad de la Palmas de la Gran Canaria. Recuperado de: http://www.iuma.ulpgc.es/~nunez/clases-micros-para-com/mpc0809-trabajos/mpc0809-HimarAlonso-Medicina.pdf

Digilent. (2016). Nexys 4 FPGA Board Reference Manual (Datasheed). Recuperado de: https://reference.digilentinc.com/_media/nexys:nexys4:nexys4_rm.pdf

K. A. A. Waterman. (2017). *The RISC-V Instruction Set Manual* Recuperado de: https://riscv.org/specifications/.

Laung, T. Cheng, Wen. (2006). *VLSI Test Principles and Architectures*. Elsevier Inc

Lee, J. (2001). *Micro-UART*. CMOSEXOD

Pacheco, A (2013). *Verificación Funcional pre-silicio del UART 16550 de Open Cores mediante la metodología UVM.* Universidad de Costa Rica, Facultada de Ingeniería.

Patterson, D. Hennessey, J. (2017). *Computer Organization and Design RISC-V Edition.* Morgan Kauffman.

Vasudevan, S. (2006). E_ective functional veri_cation: Principles and proces-ses. Springer.

Verdino, G. (2010). *Micro Marketing*. Mc Graw Hill.

Rosenberg, S y Meade, K. A. (2010). *A practical guide to adopting the Universal Verfication Methodology (UVM)*. Cadence Desing Systems.

Tocci, J y Widmer, S (2003). *Sistema Digitales, Principios y Aplicaciones*. Pearson Prentice Hall.