UNIVERSIDAD NACIONAL DEL SUR

TESIS DE MAGÍSTER EN INGENIERÍA

Desarrollo de "IPs" basados en AMBA: interfaz serial y generador de números aleatorios de señal mixta.

Ing. Niria I. Osterman Sarracini

Bahía Blanca Argentina

2017

Prefacio

Esta Tesis se presenta como parte de los requisitos para optar al grado Académico de Magíster en Ingeniería, de la Universidad Nacional del Sur y no ha sido presentada previamente para la obtención de otro título en esta Universidad u otra. La misma contiene los resultados obtenidos en investigaciones llevadas a cabo en el ámbito del Departamento de Ingeniería Eléctrica y de Computadoras durante el período comprendido entre el 26 de Agosto de 2014 y el 29 de Noviembre de 2016, bajo la dirección del Dr. Pablo Sergio Mandolesi.

Niria Iris OSTERMAN SARRACINI
Departamento de Ingeniería Eléctrica y de Computadoras
UNIVERSIDAD NACIONAL DEL SUR

UNIVERSIDAD NACIONAL DEL SUR

Secretaría General de Posgrado y Educación Continua

La presente tesis ha sido aprobada el ..../..../....., mereciendo la calificación de .....(.....)

Resumen

Los avances en las tecnologías y en las herramientas de fabricación de circuitos integrados han permitido aumentar la cantidad de componentes que se integran en un mismo chip, dando lugar a sistemas más complejos denominados sistemas en un chip (SoCs). Para reducir costos y tiempos del mercado, una característica fundamental que buscan los fabricantes de SoCs es que sus diseños sean reutilizables. Por este motivo, para realizar la comunicación entre componentes se utilizan interfaces estandarizadas.

En esta tesis se presenta el desarrollo de dos periféricos con un bus de comunicación denominado AMBA, creado por ARM. En primer lugar se muestra el desarrollo de un generador de números aleatorios. Este periférico obtiene su característica a partir de un fenómeno físico aleatorio como es el ruido térmico proveniente de resistencias. El ruido generado se amplifica y se utiliza para controlar la frecuencia de salida de un oscilador. Con esta señal se realiza el muestreo de una de mayor frecuencia provista por un oscilador de anillos. La aleatoriedad de los bits resultantes del muestreo se comprueba con las pruebas estadísticas normalizadas por el National Institute of Standards and Technology (NIST). Luego, para el aprendizaje del bus AMBA, se muestra el desarrollo de un Receptor Transmisor Asíncrono Universal (UART).

Ambos periféricos se diseñan con una interfaz AMBA APB y tienen la posibilidad de interactuar con un bus AMBA AHB por medio de un dispositivo denominado puente APB. Los periféricos y su respectiva interfaz AMBA se implementan en un proceso de Tower Jazz de 180 nm.

Abstract

The advances in tool and technologies for manufacture of integrated circuits have allowed an increase in the number of components in the same chip, resulting in more complex systems called systems on a chip (SoCs). In order to reduce development costs and time to market, SoCs manufacturers require that their designs be reusable. For this reason, standardized interfaces are used for communication between components.

This thesis presents the development of two peripherals that use the standardized AMBA bus, created by ARM. As a first approach to understand the bus interface, a Universal Asynchronous Receiver-Transmitter (UART) is implemented. Next, a true random number generator is designed. Its implementation relies on the physical phenomena of thermal noise intrinsic from resistance. Then this noise is amplified and used to control an oscillator's output frequency. This signal is used to sample a larger frequency signal provided by a ring oscillator. This generates an output bit stream whose randomness is verified with statistical tests standardized by National institute of standards and technology (NIST).

Both peripherals are designed for work with an AMBA APB bus and have the possibility to interact with an AMBA bus through a bridge. The peripherals and their respective AMBA interfaces are implemented in a 180 nm Tower Jazz process.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

El desarrollo de los sistemas digitales durante las últimas décadas no tiene paralelo con ningún otro área de la tecnología. Las constantes mejoras en las técnicas de miniaturización de los dispositivos semiconductores han permitido un aumento sostenido en la densidad de transistores por unidad de área de silicio, lo que se traduce en una reducción continua del costo por transistor y un aumento en la complejidad de los sistemas. Este fenómeno, junto con las grandes mejoras en desempeño y el enorme aumento en el rango de aplicaciones para los sistemas digitales, han tenido como resultado la utilización de sistemas más complejos, aún para aplicaciones básicas. Los términos como procesadores *multi-core* y sistemas en chip (SoC) se han vuelto cotidianos en la jerga de la electrónica de consumo.

Sin embargo, a medida que aumenta la densidad de dispositivos en un chip, los problemas de la ingeniería de sistemas se vuelven más complicados. Por ejemplo, los dispositivos más pequeños deben operar a menores tensiones para mantener constantes los campos eléctricos y, al mismo tiempo, la mayor cantidad de dispositivos genera un aumento en el consumo de corriente, duplicando así la complejidad de la distribución de potencia. Asimismo, los temporizados y la distribución de las señales lógicas se vuelven cada vez más desafiantes (Oklobdzija, 2007).

1.1. Evolución de la tecnología de fabricación de circuitos integrados

A fines de la década del '70, Gordon Moore (Moore, 2006) observó que el número de dispositivos que podían fabricarse en un chip aumentaba exponencialmente en el tiempo, a una tasa aproximada de 50 % por año (ver Figura 1.1). Al mismo tiempo, el retardo de una compuerta básica decrece un 13 % por año, lo que significa que se reduce a la mitad cada 5 años. Si bien en la actualidad se está llegando a los límites físicos de esta predicción, la industria de semiconductores ha logrado mantener estas tendencias por décadas.

Por otra parte, el área efectiva de los circuitos integrados ha evidenciado también un aumento sostenido. Los chips han escalado de 2 mm $\times$ 2 mm en la década del '60, a más de 40 mm por lado en la actualidad. La combinación de la disminución en el ancho de una línea de metal con el aumento del tamaño del chip, ha generado un crecimiento del 22 % anual en el número de pistas que pueden ser colocadas lado a lado a lo largo de un chip (ver Figura 1.2).

1.2. Sistemas en chip

La reducción del tamaño de los dispositivos y la ampliación del área de silicio permiten diseños de circuitos integrados de alta complejidad, compuestos por billones de transistores. Esto proporciona nuevos niveles de integración, donde un mismo chip puede incluir varios componentes. Un SoC puede contener uno o más componentes programables, como procesadores de propósito general, procesadores digitales de señales (DSP), o bloques de propiedad intelectual, diseñados específicamente para alguna aplicación (IP). Además pueden contener un *front-end* analógico, memoria, dispositivos de entrada-salida (I/O), etc. En otras palabras, un SoC es un circuito integrado que puede implementar la mayoría de las funciones de un sistema electrónico (Benini y De Micheli, 2005).

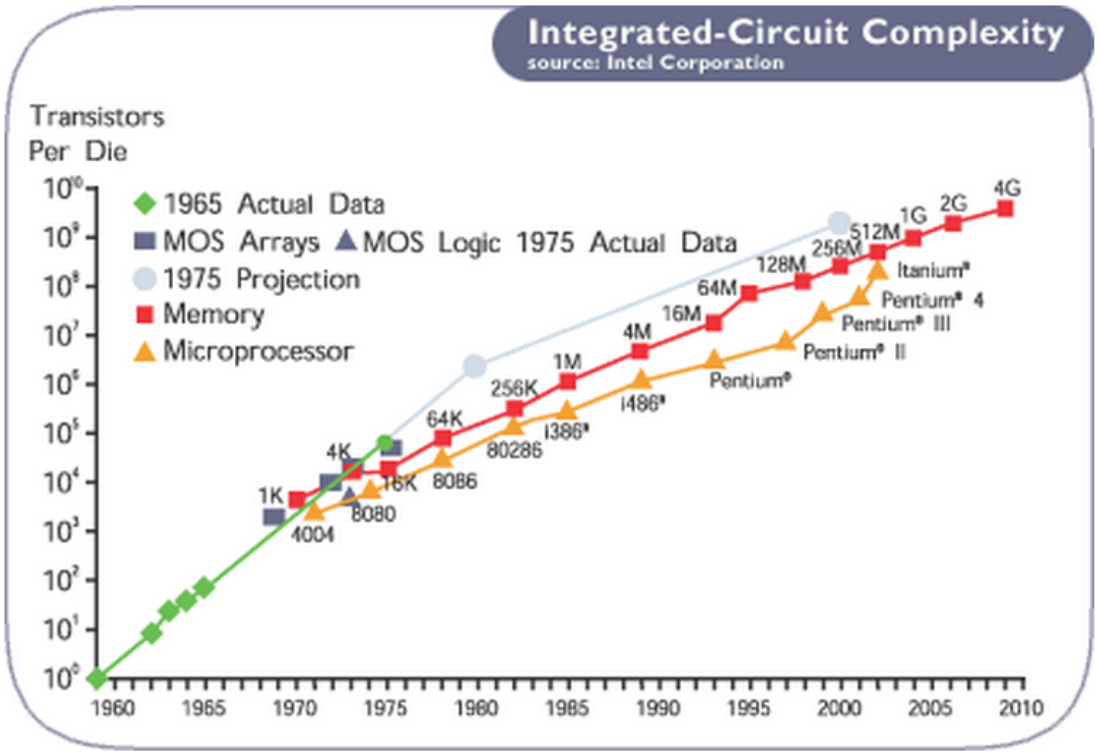


FIGURA 1.1: Descripción gráfica de la ley de Moore. Tendencia de la cantidad de transistores colocados dentro de un chip y los avances tecnológicos asociados. Fuente: Intel Corporation.

Para las aplicaciones de un SoC, la reusabilidad es una característica fundamental de los bloques constitutivos. Los equipos de diseño que generan módulos que van a ser reutilizados, suelen recurrir a interfaces estandarizadas que faciliten su compatibilidad con los demás bloques, y con futuros diseños (Horspool y Gorman, 2001). De esta manera, las interconexiones dentro del chip se convierten en uno de los mayores desafíos para los diseñadores. Aunque existen estrategias más avanzadas (Trobec, 2009), (Vivet y col., 2015), los buses siguen siendo el esquema de interconexión más utilizado en los sistemas digitales complejos. Un bus provee un protocolo de comunicación entre los componentes del circuito. También define un espacio de memoria, y la utilización de varias direcciones pertenecientes a ese espacio. Por ejemplo, un rango de direcciones pueden estar asignadas a un dispositivo particular, conectado al bus.

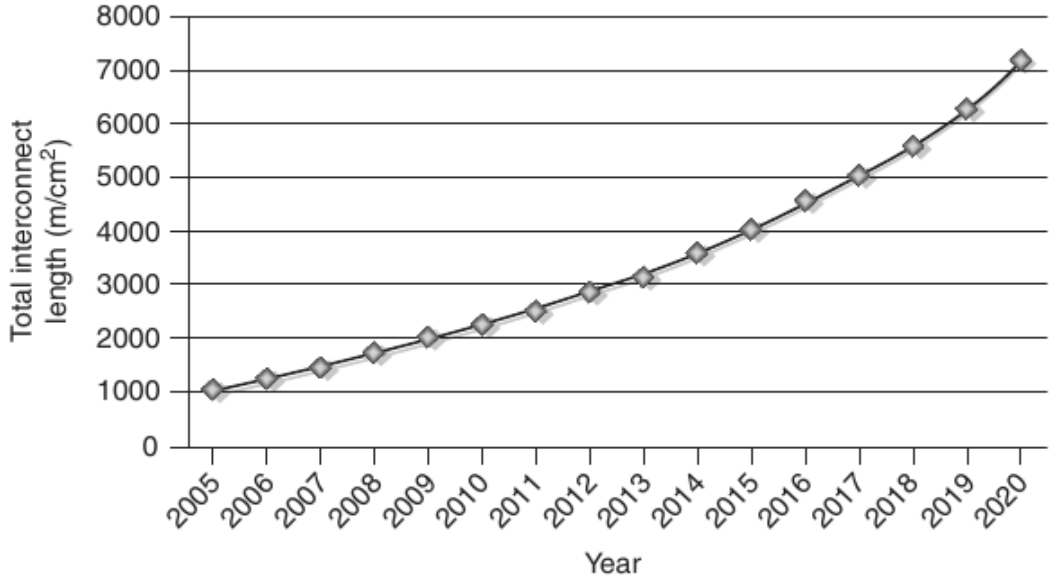


FIGURA 1.2: Tendencia de la longitud de interconexión total en un chip (*ITRS* 2005).

1.2.1. Buses de interconexión

Las tecnologías actuales ofrecen múltiples capas de metales (generalmente aluminio) y al menos una de polisilicio, que el diseñador puede utilizar para conectar los dispositivos que conforman el circuito. Incluso las capas de material altamente dopado $n+$ y $p+$, que se utilizan en las áreas de *drain* y *source*, pueden servir para conectar componentes. En el diagrama esquemático, estos cables se ven como líneas simples que no tienen impacto aparente en el desempeño del circuito. Sin embargo, la compleja geometría de las interconexiones generan efectos parásitos capacitivos, resistivos e inductivos, que tienen efectos negativos en el comportamiento del sistema:

1. Un aumento en el tiempo de propagación, que en la actualidad puede ser mayor que el retardo de una compuerta simple, y que degrada el desempeño del sistema.

2. Fuerte impacto en la disipación de energía y la distribución de potencia.

3. Introducción de fuentes de ruido, que afectan la confiabilidad del circuito.

Durante la mayor parte de la historia de los circuitos integrados, los cables de interconexión dentro del chip han sido considerados actores secundarios, que sólo deben ser tenidos en cuenta en situaciones especiales, o al realizar análisis de alta precisión. Desde la introducción de las tecnologías *deep-submicron*, este paradigma ha sufrido un cambio dramático. El impacto del escalado de la tecnología en los efectos parásitos que introducen los cables de interconexión es muy diferente al que evidencian los elementos activos, y tienden a ganar importancia cuando se reducen las dimensiones y se aumentan las velocidades de operación (ver Figura 1.3). De hecho, estos efectos parásitos han comenzado a dominar las métricas más relevantes en los circuitos integrados digitales, como la velocidad, el consumo de energía y la confiabilidad. El análisis cuidadoso del rol y el comportamiento de las líneas de interconexión en los dispositivos semiconductores ya no es un paso deseable, sino necesario.

Las arquitecturas de comunicación dentro de un SoC están definidas por dos factores: la topología y los parámetros del protocolo. La topología de una arquitectura de comunicación se refiere a la forma en que los buses se encuentran interconectados, y cómo cada uno de los componentes son mapeados a cada bus. Los parámetros de un protocolo se refieren a aquellos como los esquemas de arbitraje, tamaño del bus, frecuencias del reloj del bus, tamaño de *buffers* y tamaño de las transferencias por ráfagas, los cuales son específicos para el protocolo utilizado por la arquitectura de comunicación.

1.2.2. Topologías de la arquitectura de comunicación

La topología de una arquitectura de comunicación define la estructura física por la cual se interconectan en el sistema los dispositivos maestros y esclavos. Existen numerosas topologías, que pueden clasificarse en los siguientes grupos:

- Bus compartido: el bus de sistema, o compartido, es la topología más simple para una arquitectura de comunicación, y de frecuente utilización en SoCs comerciales (Siemens, 1994). Varios dispositivos maestros y esclavos pueden

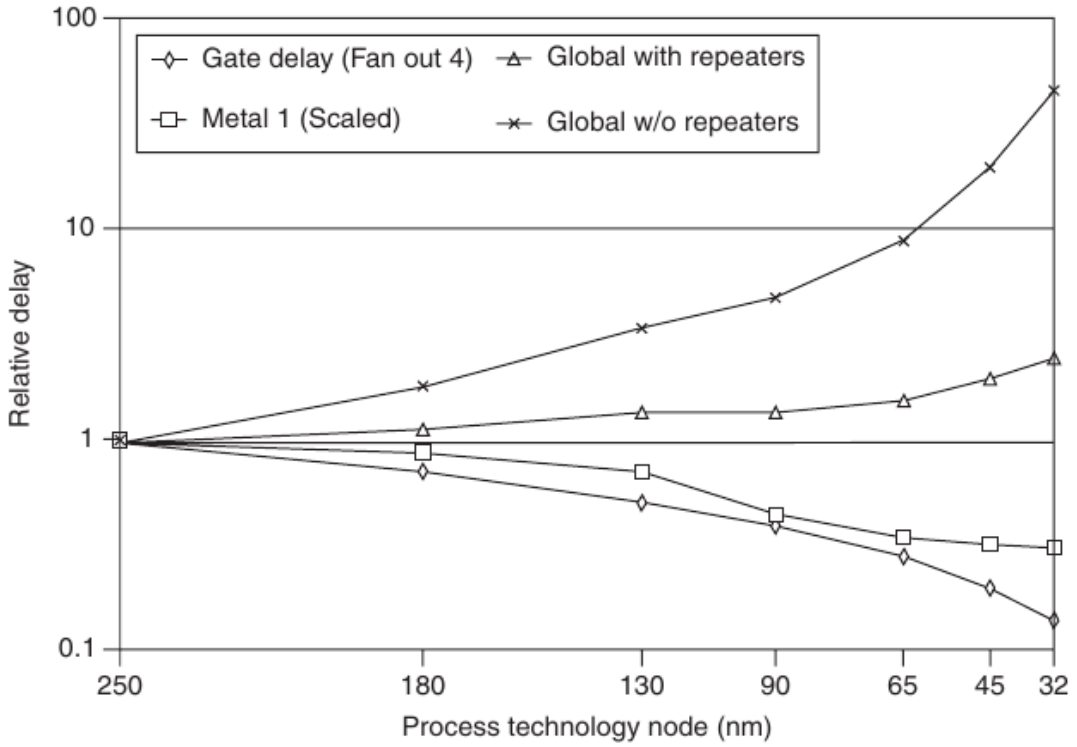


FIGURA 1.3: Comparación entre el retardo relativo de las conexiones vs. el proceso de tecnología (*ITRS* 2005).

conectarse al bus. Periódicamente, un bloque árbitro examina los pedidos de acceso acumulados de todos los dispositivos maestros, y otorga acceso al bus a uno de ellos, de acuerdo a un mecanismo de arbitraje definido por el protocolo. Las ventajas de esta topología son su arquitectura simple y fácil de construir y de extender, y su bajo costo en área. La principal desventaja es que el tiempo necesario para transferir datos es mayor que en otras alternativas. Además el consumo de energía también suele ser mayor, y su ancho de banda menor.

- Bus jerárquico: esta arquitectura consiste en varios buses compartidos, interconectados mediante puentes (*bridges*) de manera de formar una jerarquía. Los componentes del SoC se conectan al nivel de jerarquía apropiado para su requerimiento de desempeño. Los componentes de bajo desempeño se conectan a buses de bajo desempeño, y éstos se conectan a los buses de mayor jerarquía mediante *bridges* que evitan que carguen a los dispositivos de alto desempeño. Las transacciones entre buses requieren un *overhead* adicional,

y durante ese tiempo ambos buses se encuentran inaccesibles para el resto de los componentes. Esta topología es más eficiente en las transacciones debido a que la carga sobre los buses es menor, y además permite paralelizar transacciones entre diferentes buses.

- Anillo: en esta topología cada componente (maestro o esclavo) se conecta a uno o más buses en forma de anillos concéntricos. El dato se transfiere de la fuente al destino en una u otra dirección sobre el anillo, dependiendo de factores como son la disponibilidad del segmento del bus y la menor distancia al destino. Es una topología que permite operaciones en alta frecuencia.

1.2.3. Protocolos de comunicación

Los protocolos de comunicación consisten en distintos tipos de algoritmos de gestión de recursos, utilizados para determinar la prioridad de acceso a los canales de comunicación. En el resto de esta sección se describen brevemente los protocolos más utilizados.

- Static-priority: este protocolo, utilizado en arquitecturas de bus compartido, consiste en un único bloque árbitro que periódicamente examina los pedidos de acceso acumulados de todos los dispositivos maestros, y otorga acceso al de mayor prioridad (ARM, 1999).

  En este esquema los maestros de un bus tienen asignados valores de prioridad fijos. El maestro que tiene la mayor prioridad siempre accede al bus. Es un esquema que se puede implementar de forma preferente o no preferente. En la preferente, si una transferencia de baja prioridad está siendo llevada a cabo y ocurre un pedido del bus por una transferencia de mayor prioridad, la misma es interrumpida. En una implementación no preferente, se completa la transferencia de menor prioridad antes de ceder el bus a la transferencia de mayor prioridad. Este es un esquema simple de implementar y brinda buen desempeño si las transferencias críticas tienen prioridad alta. Por otro lado tiene la desventaja que dispositivos de baja prioridad pueda que nunca

obtengan el acceso al bus por accesos frecuentes de dispositivos de mayor prioridad.

- Time division multiple access (TDMA): este mecanismo está basado en un ciclo temporal, dentro del cual cada maestro tiene un slot reservado. Existen técnicas especiales que permiten aprovechar los slots no asociados con un dispositivo maestro (Sonics, 2002). Este esquema asegura alto ancho de banda para maestros que poseen transferencias críticas pero sin dejar que los de baja prioridad queden a la espera. En este esquema, a cada maestro se le asignan slots de tiempo de longitud variable, dependiendo de los requerimientos de ancho de banda del maestro. La elección de la cantidad de slots de tiempo que se asignan a cada maestro es de extrema importancia. La longitud del slot debe ser suficiente para completar una transferencia simple, pero no demasiado larga para que otras transferencias críticas tengan que esperar demasiado tiempo.

- Lottery: el bus compartido posee un bloque centralizado que administra un sorteo. Cada maestro que solicita el acceso al bus tiene asignado una etiqueta para el sorteo. El administrador de forma probabilística elije uno de los maestros como el ganador y le da acceso al bus. En este esquema se permiten múltiples transferencias, asegurando que un maestro no tome durante un largo tiempo el bus, controlando la cantidad máxima de transferencias.

- Token passing: este protocolo se utiliza en las arquitecturas tipo anillo. Una palabra especial, denominada token, circula por el anillo. Un dispositivo que recibe el token está autorizado a comenzar la transacción, y al final lo vuelve a liberar.

- Round-robin: el acceso al bus se cede, de manera circular, a cada maestro. Un maestro deja el control cuando no tiene datos para enviar o ha tenido el control del bus por un tiempo máximo permitido y pasa el control al siguiente maestro. Este es un esquema simple de implementar y garantiza una distribución equitativa del ancho de banda, y tiene la desventaja que transferencias críticas podrían tener que esperar mucho tiempo para realizarse.

1.2.4. Factores que afectan la organización y eficiencia del bus

Además de las distintas topologías y protocolos de comunicación de los buses de interconexión, existen otros factores que impactan directamente en la organización y eficiencia del bus. Algunos de ellos se mencionan a continuación:

- Modelo de programación: es el modelo al cual responden los componentes del sistema, de acuerdo a su función. Los dispositivos que envían solicitudes se denominan "maestros" y lo que las reciben se denominan "esclavos".
- Bus Split vs. Non-split: si existe un único mecanismo para arbitrar la comunicación entre maestros y esclavos, el bus se denomina non-split. En este caso, el maestro que inició la solicitud retiene el control del bus hasta que termina la operación de respuesta. Por otra parte, en un bus tipo split, el maestro cede el control después de enviar la solicitud, habilitando a otros maestros a iniciar solicitudes mientras se procesa la respuesta (Radulescu y Goosens, 2004).
- Latencia: es producto de dos factores: el tiempo de acceso al bus, hasta que el maestro está habilitado a utilizarlo, y el tiempo que se tarda en transferir el dato (Hennessy y Patterson, 2003).

1.3. Estándares de buses de comunicación

Con el aumento en la integración de SoCs y la necesidad de reutilización de bloques de propiedad intelectual (IP, del inglés *Intellectual Property*), distintos fabricantes han realizado diseños estándares de buses de comunicación. A continuación se mencionan las principales características de los más relevantes dentro de un circuito integrado.

1.3.1. AMBA

El protocolo ARM AMBA (del inglés *Advanced Microcontroller Bus Architecture*) es una especificación de interconexión dentro de un circuito integrado que permite gestionar los bloques funcionales en un SoC. El protocolo AMBA es un estándar disponible al público y promueve el diseño reusable definiendo una interfaz estándar de uso común.

El principal objetivo del bus AMBA es proveer independencia a los avances de la tecnología y fomentar el diseño de sistemas modulares. Apoyar al desarrollo de dispositivos periféricos que tengan reusabilidad mientras se minimiza el área de silicio utilizada.

Algunos de los beneficios que provee AMBA son:

- Reusabilidad de bloques de propiedad intelectual (IP).
- Flexibilidad de requerimientos.
- Arquitectura multicapa.
- Compatibilidad entre diseñadores y clientes.
- Amplio apoyo de la industria.

Un ejemplo de un sistema basado en AMBA cuyos componentes se encuentran agrupados en una topología jerárquica se muestra en la Figura 1.4.

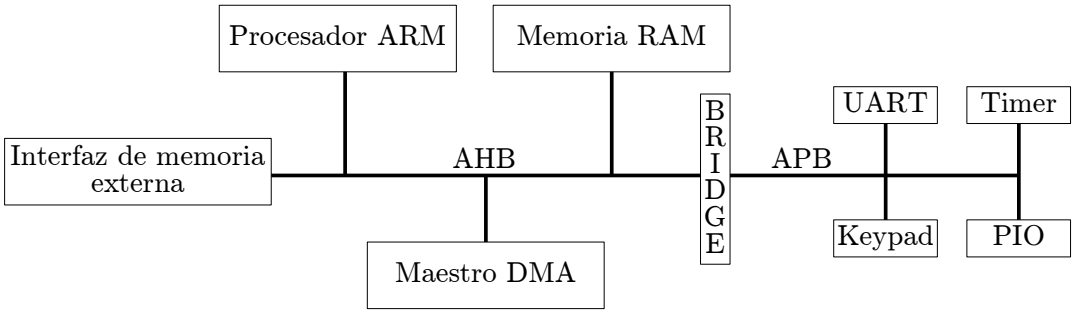


Figura 1.4: Ejemplo de sistema AMBA 2.0 (Pasricha y Dutt, 2010).

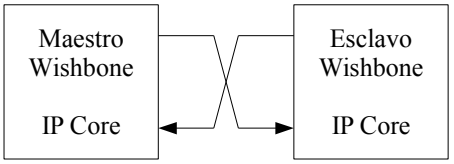
1.3.2. Wishbone

Wishbone es una arquitectura de interconexión abierta, desarrollada por Silicore Corporation. Consiste en dos tipos de interfaces: las interfaces "maestro" son bloques IP capaces de iniciar ciclos de bus, mientras que las interfaces "esclavo" son bloques capaces de aceptarlos (Ayala y col., 2006). Las implementaciones de hardware permiten varios tipos de topologías de interconexión, tales como:

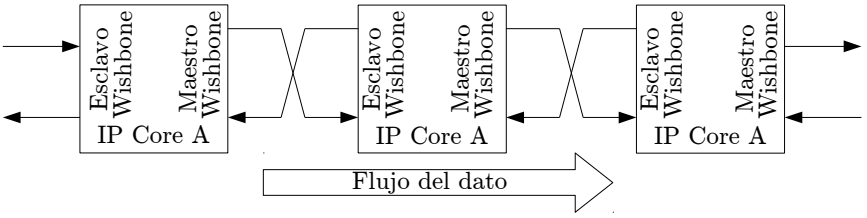
- Conexión punto a punto: este tipo de conexión se utiliza en sistemas pequeños para realizar transferencias de datos en conexiones directas entre dos dispositivos. Un esquema de este tipo de conexión se muestra en la Figura 1.5a.
- Conexión de flujo de datos: se utiliza en sistemas que poseen un flujo de datos secuencial, donde el dato fluye componente a componente como se muestra en la Figura 1.5b.
- Bus compartido: típicas en SoCs con procesadores de múltiples núcleos (MPSoCs) organizados alrededor de un único bus de sistema. Un ejemplo de esta topología se muestra en la Figura 1.5c.
- Conexión cruzada con switch: utilizada comúnmente en sistemas MPSoC donde más de un maestro pueden acceder simultáneamente a diferentes esclavos. En este caso, el maestro solicita al *switch* que le asigne un canal y, una vez establecido, transfiere los datos en modo punto a punto. Un esquema de dicha topología se observa en la Figura 1.5d.

La Figura 1.5 muestra las topologías descriptas.

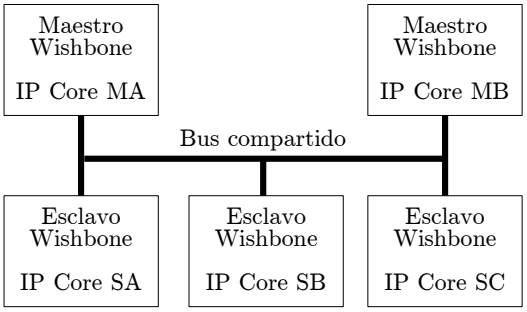
Wishbone soporta transferencias simples de lectura y de escritura y transferencias en ráfagas. En Wishbone no se definen buses jerárquicos. En aplicaciones donde se requieran dos buses de distintas velocidad, deben implementarse dos protocolos Wishbone.



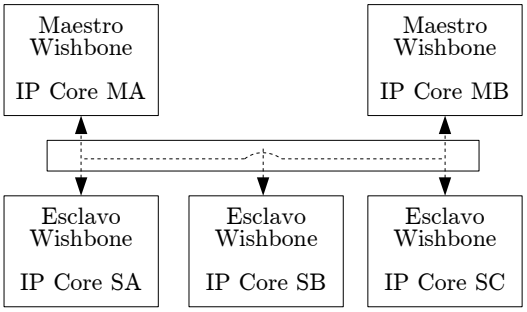
(A) Interconexión punto a punto.



(B) Conexión pipeline.



(C) Bus compartido.



(D) Cruzada. Las líneas punteadas indican una opción posible de conexión.

FIGURA 1.5: Posibles interconexiones de Wishbone.

1.3.3. Avalon

Avalon es una arquitectura de bus diseñada por Altera, para conectar procesadores y periféricos en un "sistema en chip programable" (SoPC). Se utiliza principalmente en diseños de SoC para FPGA, basados en el procesador Nios (Altera, 2005a), (Altera, 2005b).

Avalon tiene un conjunto de tipos de señales predefinidos, que el usuario puede utilizar para conectar bloques IP. Es una interfaz sincrónica, que especifica los puertos de comunicación de los dispositivos maestros y esclavos, y el temporizado para las comunicaciones entre ellos. El bus tiene líneas separadas para las direcciones, los datos y las señales de control. Además permite la conexión de múltiples maestros, que se comunican con los esclavos mediante una técnica de arbitraje denominada "slave-side".

El modelo del bus Avalon (denominado "switch fabric") provee los siguientes servicios a los dispositivos periféricos conectados: multiplexado de datos, decodificación de direcciones, dimensionamiento dinámico del bus, generación de estados de espera, asignación de prioridades de interrupciones, capacidad de realizar operaciones de lectura y escritura en bloques. Este modelo se genera automáticamente mediante la herramienta de desarrollo "SoPC Builder" de Altera. Un ejemplo de un sistema basado en esta arquitectura de interconexión se puede ver en la Figura 1.6.

1.3.4. CoreConnect

CoreConnect, desarrollado por IBM, es una estructura organizada jerárquicamente que consiste en tres buses de diferentes velocidades, que permiten una interconexión eficiente entre núcleos, macros de librería y lógica personalizada dentro de un SoC. La Figura 1.7 muestra un ejemplo de arquitectura CoreConnect.

- Bus local del procesador (PLB): es el bus principal del sistema. Es sincrónico, multi-maestro, y tiene un árbitro que permite obtener comunicaciones de

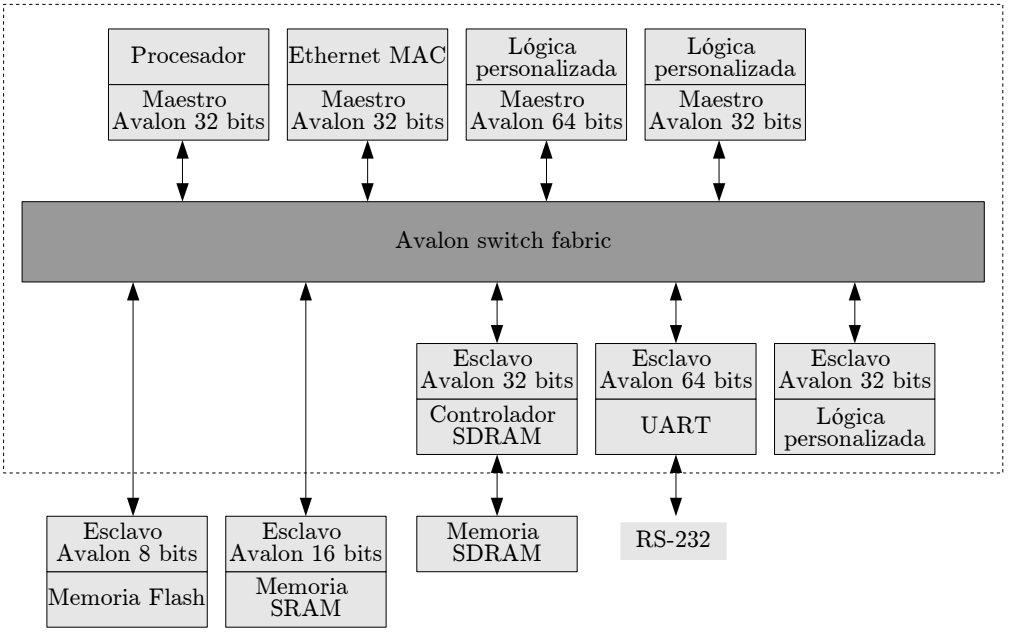


FIGURA 1.6: Sistema basado en la arquitectura de interconexión Avalon.

alto desempeño y baja latencia. Las direcciones y los datos utilizan buses separados, y permiten realizar operaciones de lectura y escritura concurrentes. Cada maestro se conecta al PLB mediante su propio conjunto de líneas de dirección, lectura, escritura y control, mientras que los esclavos utilizan líneas compartidas. El árbitro puede controlar hasta 16 dispositivos maestros, y no hay límite para el número de esclavos.

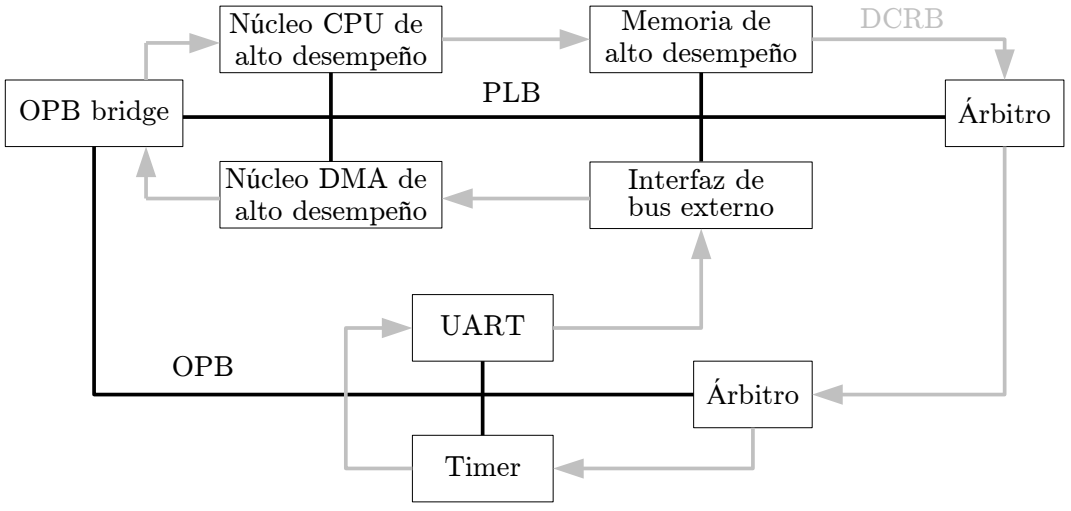


FIGURA 1.7: CoreConnect.

- Bus de periféricos (OPB): este bus está optimizado para conectar periféricos de baja velocidad y con bajo rendimiento, como UARTs o puertos serie y paralelo. Algunas de las características principales del OPB son: operación completamente sincrónica, dimensionamiento dinámico del bus, buses separados para direcciones y datos, soporte para múltiples maestros, transferencias de datos de un ciclo entre dispositivos, etc. Un maestro del PLB obtiene acceso a un periférico a través de un *bridge* OPB, que funciona como maestro para el OPB y como esclavo para el PLB.

- Bus de registro del control de dispositivos (DCRB): este es un bus auxiliar utilizado principalmente para: (1) comunicar información de estado y de configuración entre los registros de control del procesador principal y otros componentes del SoC, como memorias, procesadores auxiliares, periféricos, etc. (2) Funciones de DFT (Design for Testability). Es un bus sincrónico basado en una topología de anillo, y consiste en 10 líneas de dirección y 32 líneas de bits de datos. El árbitro utiliza un esquema de prioridad estático, pero que puede programarse.

1.3.5. Virtual Component Interface (VCI)

La VCI (VSI Alliance y On-Chip Bus Development Working Group, 2001) es una interfaz de bus estándar que define la interfaz de un componente que se conecta al bus de una arquitectura de comunicación. No define la implementación de la arquitectura del bus.

Cuenta con tres niveles de complejidad: VCI de periféricos (PVCI), VCI básica (BVCI) y VCI avanzada (AVCI). El PVCI provee una interfaz simple y de fácil implementación para aplicaciones que no necesitan todas las características del BVCI. La BVCI es adecuada para la mayoría de las aplicaciones. Posee un protocolo potente pero sin una gran complejidad. El AVCI adhiere características más sofisticadas para soportar aplicaciones con mayores requerimientos que los soportados por las dos anteriores.

El protocolo utilizado por las interfaces BVCI y AVCI no realiza ninguna conexión entre las solicitudes y las llegadas de las respuestas correspondientes. Como se mencionó anteriormente esta característica refiere a un protocolo del tipo "split".

Como interfaz, la VCI puede ser utilizada como una conexión punto a punto entre dos unidades denominadas iniciador y objetivos, donde la primera genera un pedido y la segunda responde. Como se muestra en la Figura 1.8, la interfaz puede utilizarse para una conexión a un bus. Un iniciador se conecta al bus por medio de un iniciador de bus. Un objetivo se conecta al bus por medio de un bus objetivo. Una vez que el bus iniciador y el bus destino han sido diseñados, cualquier IP puede conectarse al bus.

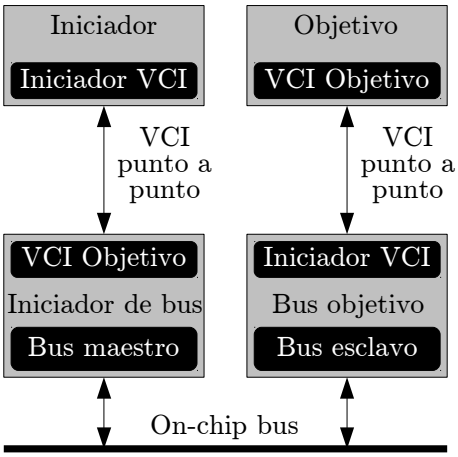


FIGURA 1.8: Dos conexiones VCI utilizadas para realizar una conexión a un bus.

1.3.6. STBus

STBus es la especificación de interconexión desarrollada por STMicroelectronics. Los protocolos e interfaces del STBus son similares a los del estándar VCI. En particular, utiliza los siguientes tipos de protocolos:

- Tipo I (Protocolo para periféricos): es un protocolo sincrónico simple con un conjunto de comandos limitado, apto para accesos a registros y periféricos lentos.

- Tipo II (Protocolo básico): este protocolo es más eficiente que el Tipo I ya que permite realizar transacciones de tipo "split" y agrega capacidad de *pipeline*. El conjunto de operaciones "read/write" permite utilizar distintos tamaños (hasta 64 bytes) y se agregan algunas operaciones especiales, como "read-modify-write" o "swap". Las transacciones también pueden agruparse en bloques para asegurarse que el flujo de datos no sea interrumpido. Este protocolo es especialmente útil para controladores de memoria externos.

- Tipo III (Protocolo avanzado): este es el protocolo más eficiente, ya que además de las transacciones del Tipo II, permite ejecuciones fuera de orden y comunicaciones asimétricas. Este protocolo es utilizado por CPUs, DMA multicanales y controladores de DDR.

El STBus tiene una naturaleza modular, y permite que dispositivos maestros y esclavos se comuniquen, independientemente de su tipo y tamaño, mediante convertidores apropiados. Tiene varias políticas disponibles para arbitrar el acceso al bus (limitación del ancho de banda, control de la latencia, arbitraje basado en prioridades, etc.) y puede implementarse en diferentes topologías (Ayala y col., 2006):

- Bus compartido: apropiado para implementaciones simples, de bajo desempeño. Esta topología se caracteriza por optimizar la cantidad de líneas necesarias, pero tiene poca escalabilidad.

- Full crossbar: dedicada a sistemas de alto desempeño, su área de cableado es grande.

- Partial crossbar: utilizada en sistemas de desempeño medio, es un buen compromiso entre las topologías anteriores.

La Figura 1.9 representa un ejemplo de utilización del STBus.

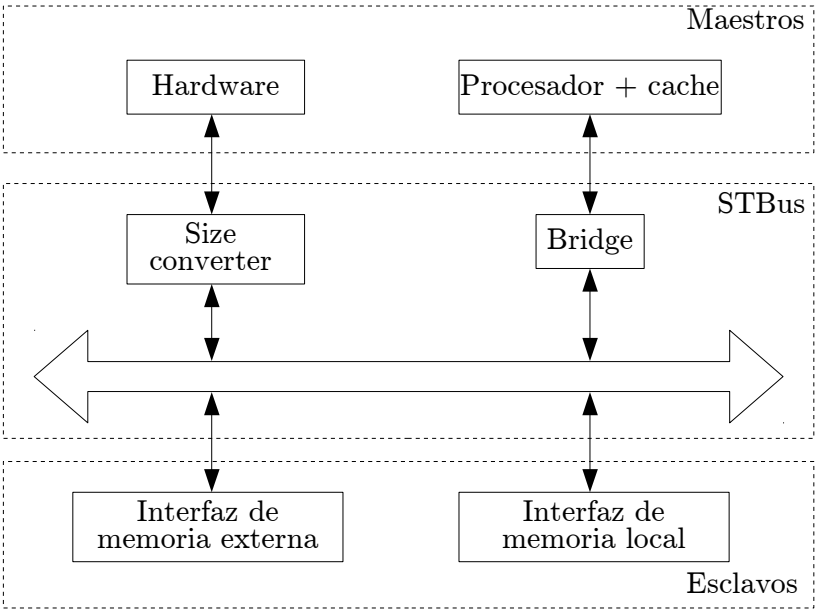


FIGURA 1.9: STBus.

1.3.7. CoreFrame

CoreFrame es una arquitectura de bajo consumo para integración de bloques de alto desempeño en un SoC. Desde un alto nivel de abstracción, esta arquitectura puede verse como un sistema de tres buses: CPU bus, PalmBus y MBus. El CPU bus se conecta al PalmBus mediante un controlador, y al MBus mediante un *bridge*. El PalmBus y el MBus son independientes, y pueden funcionar en paralelo, de manera de maximizar los recursos de ancho de banda. La Figura 1.10 ejemplifica este tipo de interconexión. PalmBus está diseñado para manejar accesos de baja velocidad desde el CPU, generalmente para acceder o configurar bloques periféricos. No se utiliza para acceder a memoria. También está diseñado para minimizar el consumo de potencia (Palmchip, 2006). Por otra parte, el MBus está diseñado para accesos de alta velocidad desde el CPU, generalmente a memorias o a periféricos de alto desempeño (Cordan, 2006).

1.3.8. Bus PI

El Bus PI (Peripheral Interconnect) fue desarrollado por varias empresas europeas en el marco del proyecto OMI (Open Microprocessor Initiative). Consiste en un

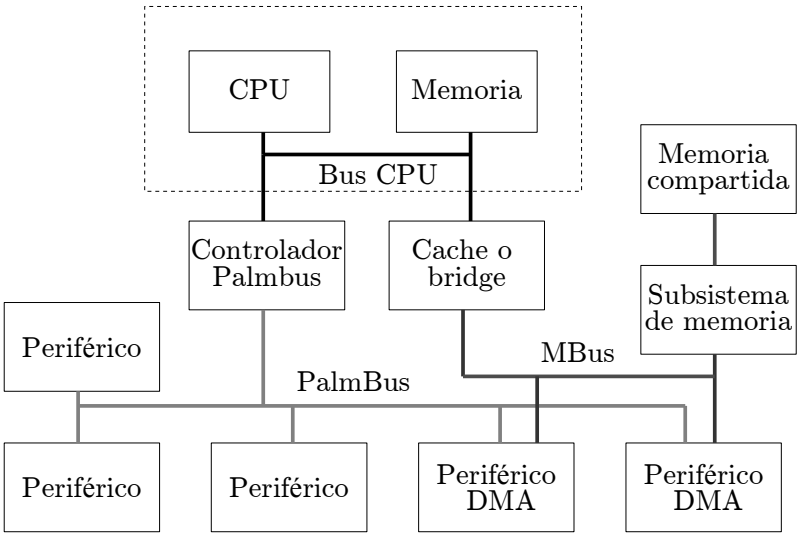


FIGURA 1.10: CoreFrame.

estándar abierto, con un conjunto de herramientas desarrollado específicamente para el diseño de SoCs. El código VHDL para bloques maestro, esclavos y unidades de control se distribuye sin costo, y varios *scripts* de síntesis para ASICs y tecnologías FPGA se encuentran disponibles (Siemens, 1994).

La arquitectura, como puede verse en la Figura 1.11, consiste en un bus sincrónico con direcciones y líneas de datos sin multiplexar, que permite la operación de múltiples maestros y *bridges*. Está diseñada para realizar transferencias de datos entre los agentes del bus, mediante mapeo de memoria. Los agentes del bus son módulos con interfaz PI bus, que actúan como maestros cuando inician una operación "read/write", y como esclavos cuando realizan la operación solicitada. Los bloques maestros típicos son procesadores, co-procesadores o módulos DMA, mientras que los bloques esclavos suelen ser unidades de memoria internas al chip e interfaces de entrada/salida.

1.3.9. Protocolo Open Core

El protocolo Open Core (OCP) es una interfaz estándar que interconecta bloques IP a un bus on-chip. Posee una interfaz configurable que posibilita a un diseñador seleccionar señales y características de configuración del protocolo que necesite

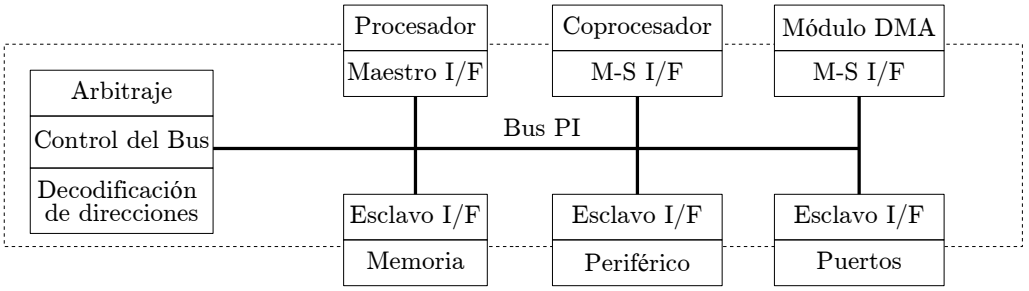


FIGURA 1.11: Bus PI.

para cumplir sus requerimientos. Las características principales del protocolo son: (1) interfaz maestro-esclavo con señales unidireccionales; (2) control de las señales por flancos positivos del reloj de la interfaz; (3) completamente sincrónico; (4) todas las señales son punto a punto (excepto clock y reset); (5) protocolo de simple pedido y respuesta; (6) soporta transferencias del dato en cada ciclo de reloj; (7) permite al maestro o al esclavo controlar la frecuencia de la transferencia; (8) ancho de palabra configurable; (9) ancho de bits de dirección configurable; (10) soporta transferencias *pipeline* y por ráfagas (Rowen, 2004).

La interfaz OCP es configurable por el usuario, de tal forma el diseñador puede definir atributos de la interfaz, como el tamaño del bus de datos y de direcciones. Además de la versión básica del OCP hay cuatro extensiones: extensión simple, extensión compleja, extensión de banda lateral, y la interfaz de depuración. La interfaz OCP básica incluye sólo señales del flujo de datos y se basa en un protocolo simple de pedido y respuesta. Sin embargo, las extensiones opcionales soportan más funcionalidades en control, verificación y validación. Extensiones simples y complejas soportan transferencias por ráfagas y operaciones de escritura con *pipeline*. Además, la extensión banda lateral soporta señales definidas por el usuario y reset asincrónico. La extensión de depuración soporta JTAG (del inglés *Joint Test Action Group*) y control de reloj.

La Figura 1.12 muestra un ejemplo de SoC que consiste en tres núcleos IP con interfaces OCP conectadas a un bus on-chip. El bus on-chip puede pertenecer a cualquiera de las arquitecturas de comunicación basadas en buses, como AMBA,

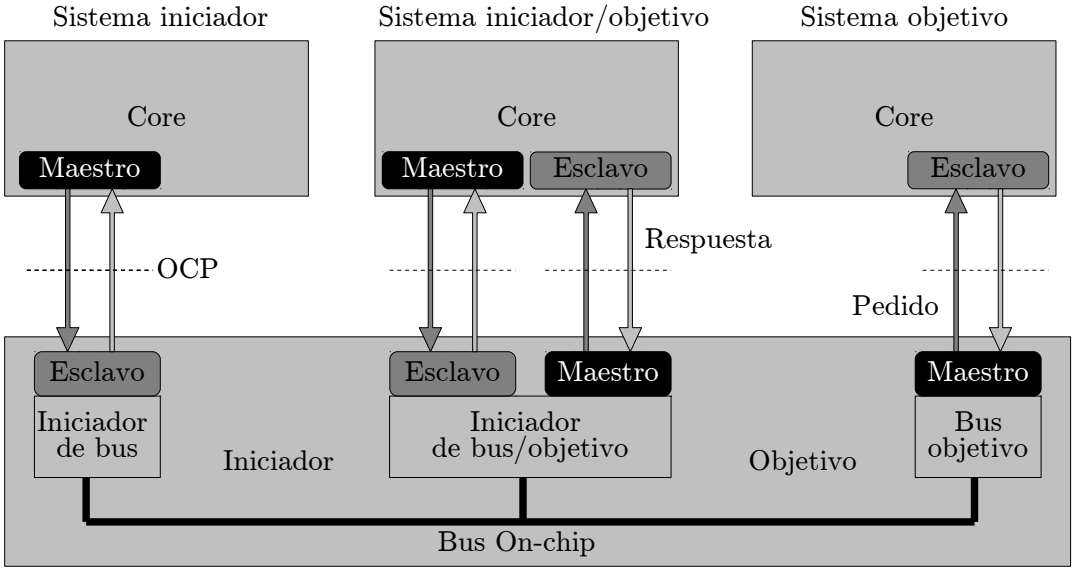


FIGURA 1.12: Instancias del OCP.

CoreConnect o STBus. Un módulo especial se requiere para traducir y mapear la interfaz OCP de los bloques IP a las señales del bus on chip. Una transferencia en este sistema funciona de la siguiente forma: el maestro (iniciador) envía la dirección, dato y señales de control por la interfaz OCP a su correspondiente interfaz esclavo traductora de bus (objetivo). Esta interfaz convierte el pedido OCP a un pedido del tipo de bus on-chip que corresponda, el cual es transmitido al destino. El pedido se recibe por el módulo de interfaz al destino, y se convierte desde un pedido de bus on-chip a un OCP. Este pedido OCP se transfiere desde el módulo de interfaz maestro al esclavo destino.

1.3.10. SiliconBackplane μNetwork

Sonics μNetwork consiste en un conjunto de arquitecturas y herramientas de diseño de SoC. SiliconBackplane es una arquitectura de comunicación para interconexiones en un circuito integrado. Implementa dos niveles de arbitraje, basado en TDMA y round-robin.

SiliconBackplane μNetwork es una red que conecta bloques IP en un SoC. μNetwork aísla los bloques IP de la red haciendo que éstos utilicen una interfaz OCP. Cada bloque IP se comunica por medio de la interfaz de bus, la cual se denomina agente

en μNetwork. Los agentes se comunican con otros a través de la red μNetwork. Si los requerimientos del sistema cambian, la interfaz soporta modificaciones de los parámetros como esquemas de arbitraje y definición del espacio de direcciones en tiempo real. En la Figura 1.13 se muestra un esquema de esta arquitectura.

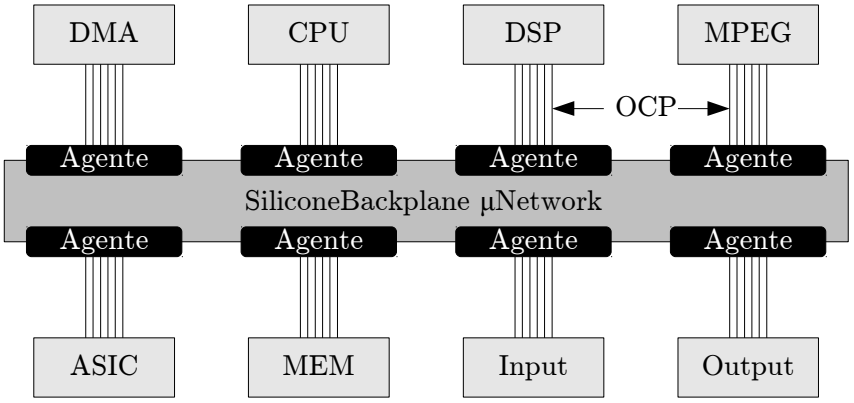


FIGURA 1.13: SiliconBackplane μNetwork.

1.4. Elección de un bus de comunicación

Las interconexiones dentro de un circuito integrado se han convertido en un sistema complejo, que no sólo involucra la funcionalidad y conectividad del diseño, sino que se ha vuelto determinante en índices como la confiabilidad y el desempeño. Por tal motivo, la elección y diseño de la arquitectura de comunicación dentro de un circuito integrado son temas importantes.

Luego de considerar los distintos estándares de comunicación más populares dentro de un circuito integrado, se opta por utilizar el bus AMBA como sistema de interconexión para este trabajo.

Como se mencionó en la Sección 1.3, el bus AMBA fue desarrollado por ARM. ARM ha dominado durante los últimos años el mercado de los sistemas digitales basados en SoCs, impulsado principalmente por la demanda de dispositivos más inteligentes y eficientes. Desde el lanzamiento de su línea Cortex-A, en el año 2010, su participación en el mercado de SoCs para teléfonos inteligentes, tablets y laptops

ha sido mayor al 86 %, y siempre en aumento como se observa en la Figura 1.14. Con dicho crecimiento, e impulsada por este dominio de ARM, la especificación AMBA se ha popularizado notablemente, considerándose en la actualidad como el estándar por omisión para las interconexiones de un SoC. Además, su disponibilidad de documentación por su naturaleza *open source* provee una ventaja considerable para elección.

Market share

	*Mobile device**	*Enterprise infrastructure*	*Embedded intelligence***
2010	59%	0%	8%
2011	71%	0%	15%
2012	78%	1%	18%
2013	85%	5%	19%
2014	86%	10%	24%

* Main applications processor in a mobile computing device, including smartphones, tablets and laptops.

** Including microcontrollers and smartcards.

FIGURA 1.14: Participación de ARM en el mercado en distintas aplicaciones. Fuente: ARM.

1.5. Organización de la tesis

En este capítulo se presentaron de forma sintética las redes de interconexión de mayor popularidad dentro de un circuito integrado, como AMBA, Avalon, CoreConnect, STBus, Wishbone, etc. Además, se describieron brevemente las principales características de los buses considerados con respecto a la topología, métodos de arbitraje, tamaño del bus y tipos de transferencias.

En el siguiente capítulo se profundiza el análisis de la arquitectura de comunicación AMBA. Se realiza una reseña de la evolución del bus y de los distintos estándares.

Luego se describe el bus APB, con la especificación del protocolo presentando la descripción de las señales, los tipos de transferencias y los estados de operación. Este bus es de gran importancia en el diseño, ya que permitirá conectar a través del puente los dispositivos periféricos con el bus AMBA AHB. También se presenta una descripción del puente APB-AHB con el fin de entender su funcionamiento básico, y se muestran simulaciones realizadas sobre el mismo.

En el Capítulo 3 se presenta uno de los dispositivos periféricos diseñados, un generador de números aleatorios. En primer lugar se realiza una introducción de los tipos de generadores y los requerimientos según la aplicación. Luego se describen los métodos más utilizados para verificar la aleatoriedad de los números generados. A continuación se presentan distintas implementaciones de generadores que se encuentran en la bibliografía. Con un panorama de las distintas opciones, se realiza un análisis comparativo de los generadores presentados en el capítulo y luego se especifica y justifica en cuál diseño se basa la implementación. El generador propuesto consta de una fuente de ruido térmico proveniente de resistencias, de un amplificador, de un oscilador controlado por tensión y de un oscilador de anillos. Además del diseño y su interfaz APB, se muestran simulaciones y la implementación física en un proceso de 0,18 μm de TowerJazz .

Un Receptor-Transmisor Asíncrono Universal (UART) se describe en el Capítulo 4, a pesar de que su desarrollo es anterior al generador de números aleatorios. Este dispositivo se realizó para fortalecer el aprendizaje del bus AMBA y del puente APB-AHB, además por ser un elemento de gran utilidad. Se realiza una descripción de la comunicación serie y el protocolo utilizado. A continuación se presenta el diseño propuesto, con las características y la descripción de cada bloque. Luego, se muestran las simulaciones de cada bloque y la interfaz APB para comunicarse con el bus. Por último, se especifica la síntesis lógica con las simulaciones resultantes y la síntesis física, para dar como resultado el diseño físico del sistema completo.

Capítulo 2

Especificación AMBA

La especificación AMBA (*Advanced Microcontroller Bus Architecture*) ideada por ARM, es un estándar abierto que define las especificaciones de interconexión dentro de un circuito integrado para la conexión y gestión de bloques funcionales en un SoC.

Actualmente, AMBA es uno de los estándares de comunicación dentro de un circuito integrado más utilizados. Su meta es proveer una especificación flexible de una arquitectura de bus de comunicación de alto desempeño, independiente de la tecnología, con la mínima utilización de área de silicio, y promover la reutilización de IPs. Esto último es esencial para reducir los costos y tiempos en el desarrollo de los SoCs. La reutilización de IPs requiere de un estándar común que brinde soporte a una amplia variedad de SoCs con diferentes consumos, desempeños y requerimientos de área, aportando flexibilidad.

AMBA lidera la industria de los sistemas de buses de interconexión utilizados hoy en día, dando conectividad a IPs entre los que se pueden mencionar controladores de memoria, GPUs y CPUs. El estándar fomenta el diseño de sistemas modulares y brinda compatibilidad entre componentes de distintos equipos de diseños y comerciantes.

Esta especificación contiene varias interfaces que permiten interconectar sistemas de distinta complejidad y con diferentes requerimientos como velocidad y potencia,

dependiendo del tipo de dispositivo o periférico a ser conectado. A continuación se mencionan los estándares actuales:

- CHI (Coherent Hub Interface): es el bus de mayor rendimiento. Utilizado en procesadores para redes de datos y servidores, para cubrir las necesidades de los SoCs actuales en donde núcleos individuales deben acceder y compartir datos a través del chip. Creado por ARM para proveer comunicación entre la nueva serie de procesadores Cortex-A50.
- ACE (AXI Coherency Extensions): utilizado en las nuevas tecnologías compuestas por distintas arquitecturas de procesadores con el fin de aumentar el desempeño a la vez que se reduce el consumo. Estas tecnologías son muy utilizadas en tabletas y teléfonos inteligentes.
- AXI (Advanced eXtensible Interface): ampliamente utilizado en nuevos diseños que cuentan con varios procesadores, controladores gráficos y otros periféricos sofisticados en donde el bus AHB no alcanza a cumplir con las demandas de los requerimientos, a su vez que mantiene compatibilidad entre dicho bus y el APB.
- AHB (Advanced High-Performance Bus): es el bus AMBA mayormente utilizado en procesadores, memorias on-chip e interfaces para memorias externas en un bus de alta velocidad de reloj.
- APB (Advanced Peripheral Bus): utilizado para periféricos de baja velocidad.
- ATB (Advanced Trace Bus): para propósitos de depuración.

2.1. Evolución del bus AMBA

A lo largo del tiempo, el bus AMBA fue actualizando su especificación de manera de proveer mayores prestaciones al continuo crecimiento en las capacidades y

requerimientos de SoCs. En la mayoría de los casos, las distintas versiones son compatibles entre sí, y la selección de la especificación depende de los requerimientos de los dispositivos conectados al bus.

La primer versión del protocolo lanzada en 1995, contenía solo dos buses, el *Advanced System Bus* (ASB) y el *Advanced Peripheral Bus* (APB). En 1999, ARM lanza la segunda versión denominada AMBA 2. En la misma se incluyó el *High Performance Bus* (AHB) ampliamente utilizado en arquitecturas ARM7, ARM9 y diseños basados en Cortex-M.

En 2003, ARM introdujo la tercera generación AMBA 3 que define cuatro interfaces que, en conjunto, cubren los requerimientos del tráfico de datos dentro de un circuito integrado. Entre ellas se encuentra la interfaz AXI3 que soporta operaciones a alta velocidad, consideraciones en el consumo y transacciones de lectura y de escritura simultáneas, permitiendo el tráfico de datos con alto desempeño. El bus AHB3 para interconexiones eficientes entre periféricos simples donde no se requiere la interfaz AXI. Además, incluye el bus APB3 para transferencias de bajo ancho de banda para acceder a registros de control y para la interconexión de periféricos. Por último, se encuentra el bus ATB que provee un sistema de seguimiento para propósitos de depuración.

En 2010 se introduce la especificación AMBA 4 comenzando con AXI4, y en 2011 se extiende con AMBA 4 ACE, interfaces utilizadas en los procesadores de ARM Cortex-A9 y Cortex-A15. Poseen características que permiten interconexiones de alta velocidad, típicas en aplicaciones móviles. Esta especificación sumó cinco interfaces adicionales a las que poseía el protocolo AMBA 3. ACE (*AXI Coherency Extensions*), ACE-Lite, AXI4, AXI4-Lite, AXI4-Stream. El ACE permite la comunicación entre procesadores y habilita el uso de nuevas tecnologías. El ACE-Lite es un subconjunto del ACE. El AXI4 es una nueva versión del bus AXI3 que mejora el desempeño y las interconexiones cuando se utilizan múltiples maestros. AXI4-Lite es un subconjunto del AXI4 para menores requerimientos. El bus AXI4-Stream fue diseñado para transferencias de dato unidireccional desde el maestro hacia el esclavo con la reducción de señales e ideal para una implementación en FPGA.

Por último, en 2013, se introduce la especificación AMBA 5 CHI permitiendo comunicaciones a mayores velocidades y con nuevas características de diseño para reducir la congestión. La última versión del protocolo AMBA 5 cuenta con el bus AHB5 que es el más utilizado en los procesadores Cortex-M para diseños embebidos y SoCs de baja latencia. La interfaz AMBA 5 CHI es una interfaz para la nueva generación de procesadores como el Cortex-A57 y Cortex-A53 y controladores de memoria dinámica. Su arquitectura se ha desarrollado de tal forma de mantener el desempeño del bus a pesar de un aumento en los componentes y en el tráfico de información. Es un bus optimizado para lograr un óptimo entre desempeño, consumo y área. Por último, la interfaz AHB5 tiene la base de la especificación AHB4 pero se le suman nuevas características para interactuar con la interfaz AMBA AXI4.

2.2. Selección de la especificación AMBA

El bus AMBA conecta distintos bloques que conforman un sistema, brindándoles la posibilidad de comunicarse entre sí por medio de señales que controlan la comunicación. En el desarrollo de este trabajo, los periféricos realizados son una interfaz de comunicación serie asincrónica o UART (por sus siglas en inglés, *Universal Asynchronous Receiver-Transmitter*), y un generador de números aleatorios o RNG (por sus siglas en inglés, *Random Number Generator*). Debido a los requerimientos y a la aplicación de estos dispositivos, ambos se realizan con una interfaz AMBA APB y se encuentran conectados a un bus común. Como estos periféricos necesitan de un control externo y debido a que varias arquitecturas de microprocesadores se comunican con AHB, es necesario que el sistema cuente con el control brindado por un dispositivo, denominado puente o *bridge* AHB-APB que transforma las señales APB en señales del bus AHB. Un esquema se muestra en la Figura 2.1.

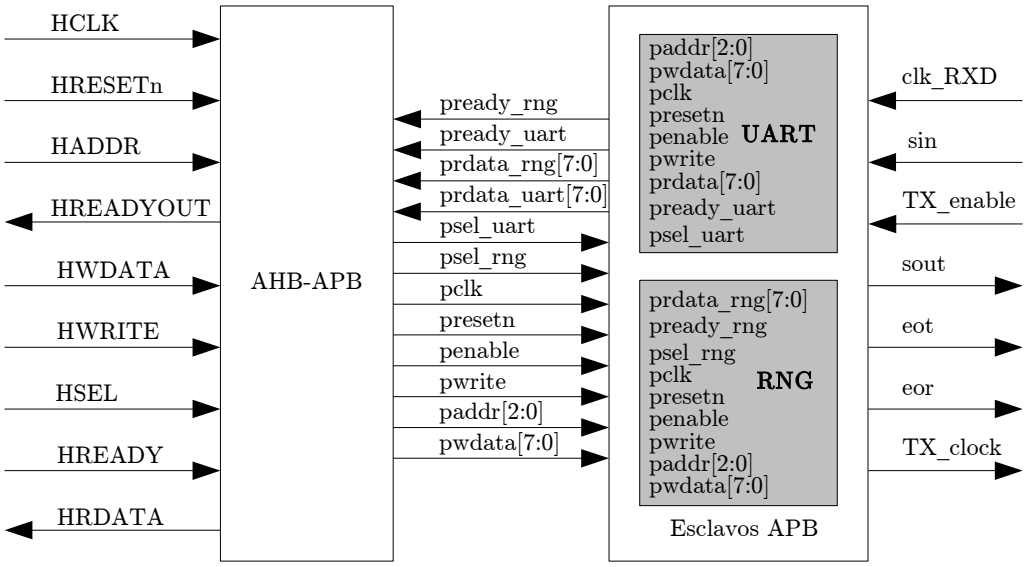


FIGURA 2.1: Diagrama de interconexión de los dispositivos y el bus AMBA.

El *bridge*, como maestro APB, controla el acceso de los periféricos al bus. Este dispositivo, como esclavo AHB, se conecta a un bus de este tipo para ser controlado por un único maestro que interprete el protocolo AHB, como un microprocesador.

2.3. Bus AMBA APB

El protocolo APB define un bus optimizado para reducir el consumo y la complejidad de la interfaz. Fue diseñado con el objetivo de conectar dispositivos que no requieran características avanzadas en desempeño y complejidad. Se utiliza, por ejemplo, para el acceso a registros de control programables de dispositivos periféricos. El bus APB cuenta con un único maestro encargado de controlar las transferencias con uno o varios esclavos. Este maestro permite administrar la conexión entre los dispositivos que funcionan con el bus APB con buses más avanzados como el AHB, AHB-Lite, AXI o AXI-4-Lite.

En la Tabla 2.1 se muestran las señales del bus APB. Todas se sincronizan con flancos positivos de la señal de reloj y cada transferencia demora al menos dos

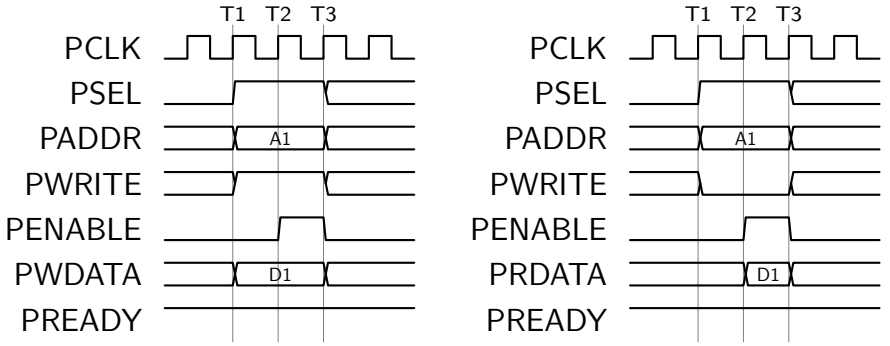
Tabla 2.1: Señales del bus AMBA APB.

Señal	Fuente	Descripción
PCLK	Fuente de reloj	Las transferencias APB se controlan por flancos positivos de PCLK.
PRESETn	Reset del bus	Señal activa en 0.
PADDR	Maestro	Señal de dirección, puede ser de hasta 32 bits.
PSELx	Maestro	Indica que un esclavo está seleccionado y que hay una transferencia. Hay una señal PSEL para cada esclavo.
PENABLE	Maestro	Señal que indica el segundo ciclo y ciclos subsecuentes de una transferencia.
PWRITE	Maestro	Indica una transferencia de escritura cuando es 1 y una de lectura cuando es 0.
PWDATA	Maestro	Dato de escritura del bus cuando PWRITE es 1. Puede ser de hasta 32 bits.
PREADY	Esclavo	El esclavo utiliza esta señal para indicar que necesita extender una transferencia APB.
PRDATA	Esclavo	Dato de lectura del bus cuando PWRITE es 0. Puede ser de hasta 32 bits.

ciclos. Las transferencias tienen dos fases, una denominada *setup*, y otra *acceso*. Entre transferencias es imposible que estas fases se solapen.

El bus APB contiene dos canales, uno para las operaciones de lectura, y otro para las operaciones de escritura. Dichas operaciones no pueden realizarse en simultáneo debido a que existe una única señal que las controla, denominada PWRITE.

En la Figura 2.2 se muestran los diagramas de temporizado que involucran transferencias simples del bus. En la Figura 2.2a se muestra una transferencia de escritura en donde la transmisión se realiza en dos ciclos de reloj (transferencia simple). En el flanco de reloj indicado por T1, el maestro inicia una transferencia colocando en 1 la señal de selección PSEL, la dirección A1 del esclavo correspondiente, la señal PWRITE en 1 indicando una escritura, y la señal PWDATA con el dato que se desea escribir en el esclavo. Este ciclo corresponde a la fase de *setup*. En el flanco de T2 el maestro coloca en 1 la señal PENABLE. El esclavo interpreta que existe una transferencia en T2, y luego del flanco de T3 la escritura se hace efectiva. El



(A) Transferencia de escritura simple. (B) Transferencia de lectura simple.

FIGURA 2.2: Transferencias simples del bus APB.

maestro no debe cambiar las señales de control ni las de selección durante toda la transferencia, caso contrario la misma no se efectuará.

En la Figura 2.2b el maestro inicia una transferencia colocando las señales de selección y de control, en este caso la señal PWRITE la coloca en 0 indicando que se efectuará una lectura desde el esclavo. En T2 coloca en 1 la señal PENABLE y el esclavo escribe el dato en el bus antes del flanco de T3. El maestro lee el dato en T3.

En la Figura 2.3 se muestran las señales del maestro al iniciar una transferencia de escritura cuando el esclavo necesita de mayor cantidad de ciclos para terminarla. Como en el caso anterior, en T1 el maestro coloca en 1 la señal de selección del esclavo, la dirección en PADDR, la señal PWRITE en 1 indicando una escritura y el dato que desea escribir en PWDATA. En T2 el esclavo reconoce que hay una transferencia pero como no es capaz de efectuarla en el próximo flanco, coloca la señal PREADY en 0 hasta que se encuentre disponible para finalizarla. Durante este tiempo, el maestro debe mantener las señales de selección y control, en T4 el esclavo coloca en 1 la señal PREADY, y en T5 se escribe el dato de PWDATA, momento en que finaliza la transferencia.

La lectura de un dato cuando se necesita mayor cantidad de ciclos para terminarla se muestra en la Figura 2.4. En T1 el maestro inicia la transferencia seleccionando el esclavo con la señal PSEL, colocando la dirección en PADDR, y la señal PWRITE en 0, indicando una lectura. En T2 el esclavo reconoce la transferencia pero

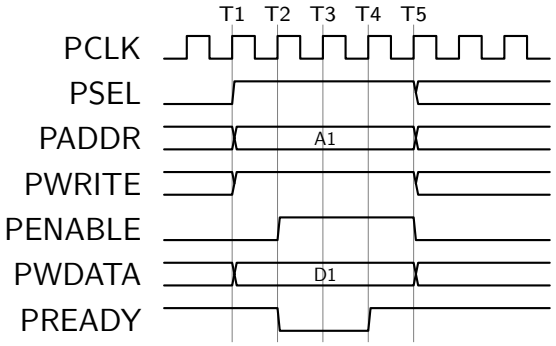


FIGURA 2.3: Transferencia de escritura con estados de espera.

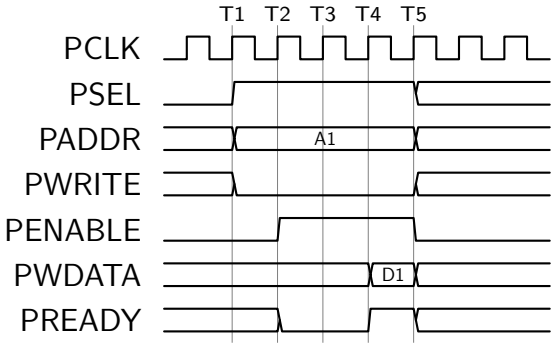


FIGURA 2.4: Transferencia de lectura con estados de espera.

como no le es posible efectuarla todavía, coloca la señal PREADY en cero. En T4 establece en 1 la señal PREADY y antes del flanco de T5, el dato en PWDATA. En T5 el maestro lee el dato del esclavo y la transferencia se completa.

Las transferencias APB se pueden describir por el diagrama de estados de la Figura 2.5. **IDLE** es el estado inicial, en el que no hay transferencias. El estado **SETUP** comienza cuando se selecciona el esclavo indicando el comienzo de una transferencia. Se mantiene en este estado durante un ciclo de reloj y luego se realiza una transición al estado **ACCESS**. En el mismo, las señales de dirección, dato y control deben permanecer activas. Se permanece en él si el esclavo extiende la transferencia colocando la señal PREADY en 0, o se vuelve a **SETUP** si se ejecuta una transferencia inmediata. Si la señal de selección PSEL y la señal de dirección PADDR indican que no se producirá una transferencia, se retorna a **IDLE**. En cada estado, el maestro APB controla las salidas PENABLE que indica la fase de acceso de una transferencia y PSEL para seleccionar un esclavo.

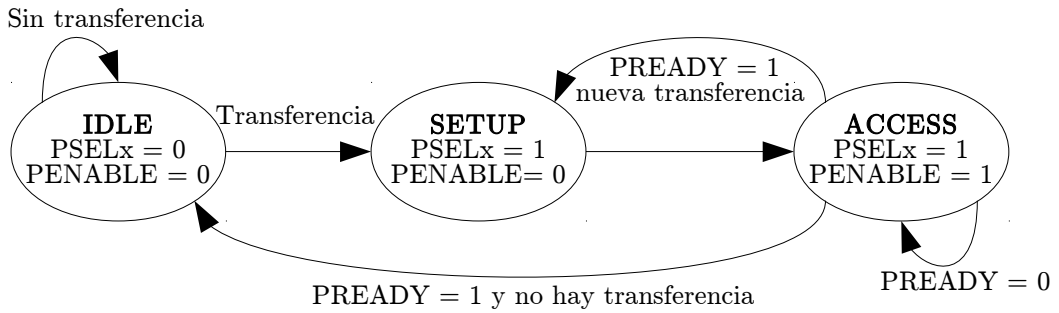


FIGURA 2.5: Diagrama de estados APB.

2.4. Bus AMBA AHB

El bus AHB es un bus de alto desempeño utilizado para cumplir con los requerimientos de dispositivos como microprocesadores, controladores de acceso directo a memoria y bloques de memoria dentro de un circuito integrado. El protocolo AHB opera con flancos positivos de reloj, permite transferencias por ráfagas, posee un bus de datos variable y admite operaciones *pipelined*. Las señales que intervienen en el bus AHB se indican en la Tabla 2.2.

En lo que respecta a la operación del bus, el maestro AHB inicia una transferencia colocando la señal de dirección y las señales de control. Una escritura en el bus realiza el movimiento del dato desde el maestro hacia el esclavo, y la operación de lectura mueve el dato desde el esclavo hacia el maestro. Cada transferencia consiste de una fase de acceso conformada por un ciclo de control y de dirección, y una fase del dato conformada por uno o más ciclos. A diferencia del APB, la fase de dirección de una transferencia ocurre durante la fase del dato de la transferencia previa. Esta superposición de la dirección y del dato le proporciona al bus AHB la característica de *pipelined*. Esta estructura aumenta el desempeño del bus permitiendo, a su vez, la operación con esclavos más lentos por medio de la señal HREADY. Cuando esta señal se encuentra en 0, inserta estados de espera en la transferencia y habilita al esclavo a tener tiempo extra para escribir o muestrear un dato.

En las Figuras 2.6 y 2.7 se muestran los diagramas de temporizado para una transferencia de lectura y para una transferencia de escritura cuando no se insertan

Tabla 2.2: Señales del bus AHB.

Señal	Fuente	Descripción
HCLK	Fuente de reloj	Controla las transferencias del bus por medio de flancos positivos de esta señal.
HRESETn	Reset del bus	Única señal activa en 0.
HADDR	Maestro	Bus de dirección de hasta 32 bits.
HWDATA	Maestro	En operaciones de escritura, es el dato escrito por el maestro hacia el esclavo.
HWRITE	Maestro	Indica la dirección de la transferencia. Cuando es 1 indica transferencia de escritura, cuando es 0, de lectura.
HRDATA	Esclavo	En operaciones de lectura, es el dato leído desde el esclavo seleccionado.
HREADYOUT	Esclavo	Cuando esta señal es 1, indica que la transferencia finalizó. El esclavo coloca esta señal en 0 para extender una transferencia.
HRESP	Esclavo	Indica si hubo un error en la transferencia.
HTRANS	Maestro	Indica el tipo de transferencia a realizar.
HBURST	Maestro	Indica si es una transferencia simple o forma parte de una ráfaga.
HSEL	Maestro	Señal de selección de un esclavo.
HMASTLOCK	Maestro	Indica que la transferencia tiene que ser procesada antes que cualquier otra transacción.
HPROT	Maestro	Provee información adicional sobre el tipo de transferencia.
HSIZE	Maestro	Indica el tamaño de la transferencia.
HREADY	Multiplexor	Señal que indica al maestro y esclavos AHB que las transferencias previas se completaron.

estados de espera. En ambos casos, la fase de dirección se cumple en solo un ciclo de reloj HCLK, lo mismo ocurre con la fase del dato. Operaciones simples se llevan a cabo de la siguiente manera:

- El maestro AHB coloca en el bus la señal de dirección y las señales de control luego del flanco de subida de HCLK.

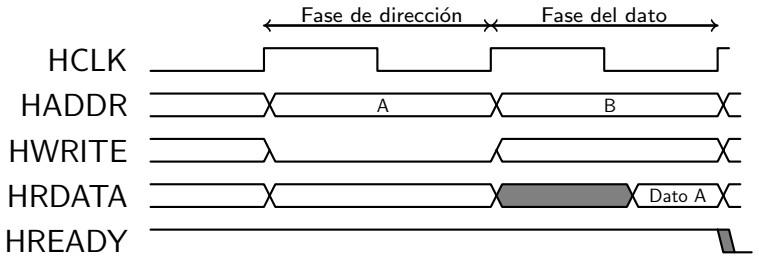


FIGURA 2.6: Transferencia de lectura del bus AHB.

- El esclavo muestrea la dirección y las señales de control en el siguiente flanco positivo de HCLK.
- Según la señal HWRITE se realiza una lectura (Figura 2.6) o una escritura (Figura 2.7).
- En el tercer flanco positivo de HCLK la transferencia culmina.

Un esclavo puede extender una transferencia con la señal HREADY como se muestra en las Figuras 2.8 y 2.9. La operación es idéntica al caso de transferencias simples, a diferencia que una vez que el esclavo muestrea la dirección y las señales de control, coloca en 0 la señal HREADY insertando ciclos de espera en la fase del dato.

En el caso de la lectura, la fase de dirección sigue siendo de un solo ciclo (aunque puede ser extendida debido a transferencias previas), pero la fase del dato se extiende y la transacción termina en el flanco positivo de reloj en el que la señal HREADY es 1, en estas transferencias el esclavo puede dar el dato válido cuando la transferencia se encuentre por finalizar. En transferencias de escritura extendidas, el maestro debe mantener el dato estable durante los ciclos extendidos y la transferencia termina en el flanco positivo de reloj cuando la señal HREADY es 1.

2.5. Puente APB - AHB

En un sistema basado en una arquitectura AMBA, con buses agrupados en una topología de bus jerárquico, los dispositivos conectados a un bus AHB se comunican

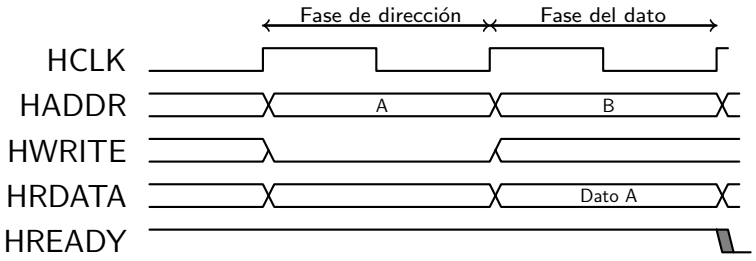


FIGURA 2.7: Transferencia de escritura del bus AHB.

por medio de un puente o *bridge* al bus APB, al cual se conectan la mayoría de los periféricos de menor ancho de banda como temporizadores, UARTs, controladores de interfaz del usuario, etc.

El *bridge* funciona como esclavo del bus AHB y es el único maestro que puede tener el bus APB. Este dispositivo realiza el temporizado necesario añadiendo estados de espera y registrando señales para adaptar las necesidades de ambos buses. Por ejemplo, el bus AHB solapa la fase del dato con la de dirección de la transferencia siguiente, mientras que el APB necesita dos ciclos de reloj para realizar una única transferencia, manteniendo las señales de control y dirección. El dispositivo puente se encarga de adaptar estas características. Un diagrama en bloques se muestra en la Figura 2.10.

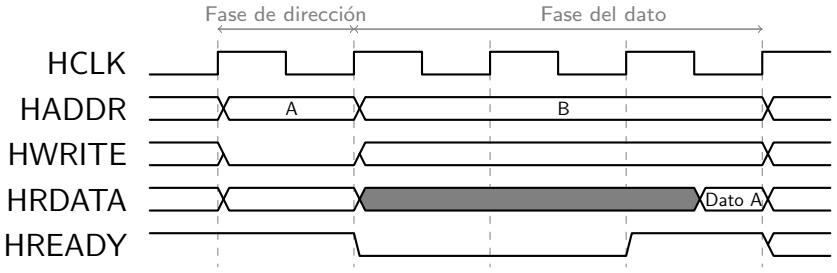


FIGURA 2.8: Transferencia de lectura del bus AHB con ciclos extendidos.

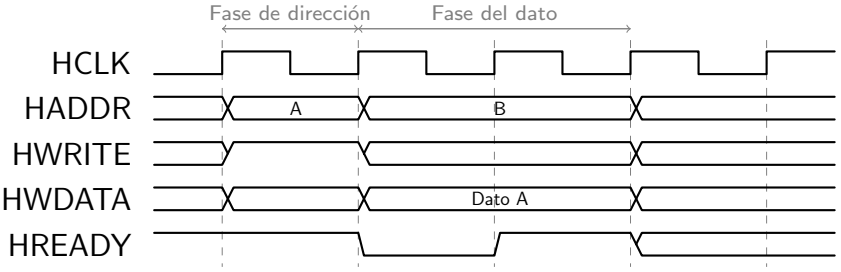


FIGURA 2.9: Transferencia de escritura del bus AHB con ciclos extendidos.

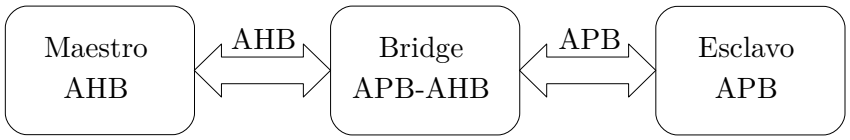


FIGURA 2.10: Diagrama en bloques del puente APB.

A continuación se describen las señales que ingresan al *bridge* desde el maestro AHB.

- HCLK: reloj del sistema, es el que controla todas las transferencias del bus AHB.
- HRESETn: esta es la señal de reset del sistema y el bus. Es la única señal que se activa en 0.
- HADDR: señal de dirección del bus.
- HWDATA: esta señal se utiliza para transferir señales del maestro AHB a los esclavos APB en operaciones de escritura.
- HWRITE: señal que indica si se realiza una operación de lectura o de escritura.
- HSEL: el maestro AHB selecciona a un esclavo, en este caso el *bridge*, por medio de esta señal.
- HREADY: esta señal se envía por el maestro AHB cuando hay otros periféricos ocupando el bus.

Las salidas del *bridge* hacia el maestro AHB son:

- HREADYOUT: cuando un periférico APB inserta estados de espera, el *bridge* se lo comunica al maestro AHB por medio de esta señal.

- HRDATA: dato que ingresa desde un periférico APB en una operación de lectura.

Las salidas del *bridge* hacia los periféricos del bus APB son:

- PSEL: señal de selección de los esclavos APB.
- PCLK: señal de reloj del sistema APB.
- PRESETn: señal de reset del sistema y del bus.
- PENABLE: señal necesaria para realizar transferencias con el protocolo AMBA APB.
- PWRITE : señal que indica a los periféricos APB si es una operación de escritura o de lectura.
- PADDR : señal de dirección necesaria para seleccionar los periféricos APB y comenzar una transferencia.
- PWDATA : en esta señal se encuentra el dato a escribir en el periférico APB.

Y por último las señales desde los periféricos APB hacia el *bridge*:

- PREADY: un esclavo APB ingresa estados de espera.
- PRDATA: es el dato leído en una operación de lectura.

En las Figuras 2.11 y 2.12 se muestra el protocolo cuando el bus AHB realiza una operación de lectura y de escritura en el bus APB. La transferencia comienza en el bus AHB en T1 y la dirección se muestrea por el *bridge* APB en T2, donde comienza la fase de setup y luego la de acceso del bus APB. El bus APB coloca el dato en PRDATA y la transferencia se completa en T4 donde el maestro AHB tiene el dato disponible en el bus.

En las Figuras 2.13 y 2.14 se muestran lecturas y escrituras sucesivas entre el bus AHB y el *bridge* APB. Cuando existen múltiples transferencias y alguna de ellas

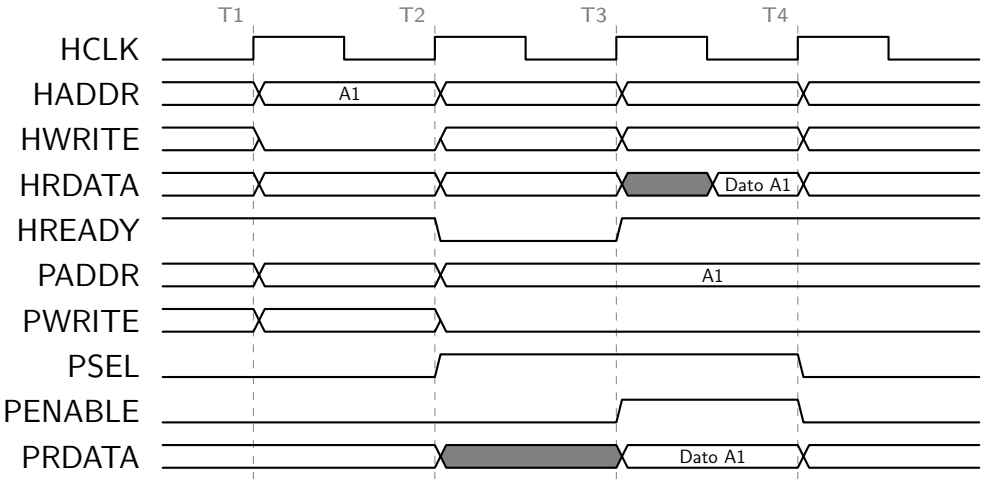


FIGURA 2.11: Transferencia de lectura al bus AHB.

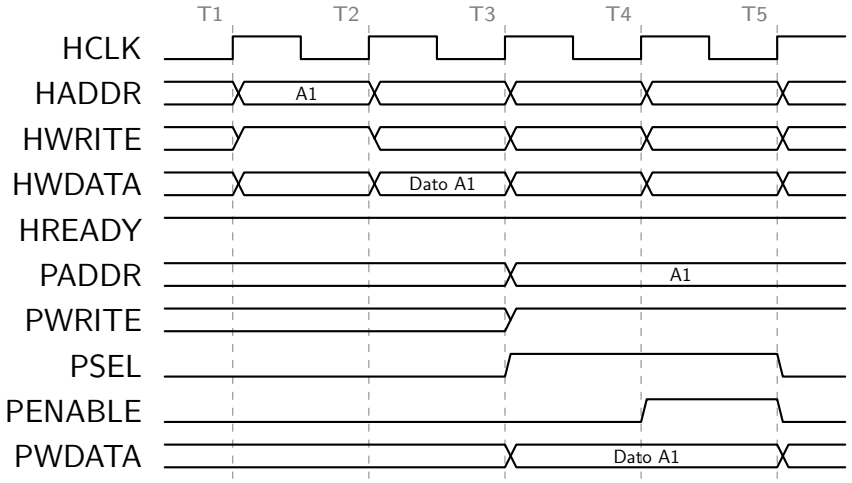


FIGURA 2.12: Transferencia de escritura desde el bus AHB.

es extendida, se extiende también la fase de dirección de la transferencia siguiente. En el caso de la lectura, si se realizan transferencias en ráfagas desde el bus AHB, todas las transferencias requieren de un solo estado de espera como se observa desde la señal HREADY.

En el caso de las transferencias de escritura, la primera se puede completar sin estados de espera, mientras que si se continúan realizando transferencias, las mismas requieren de un estado de espera.

Las funciones principales con las que debe cumplir el *bridge* APB son:

- Mantener la dirección válida durante la transferencia.

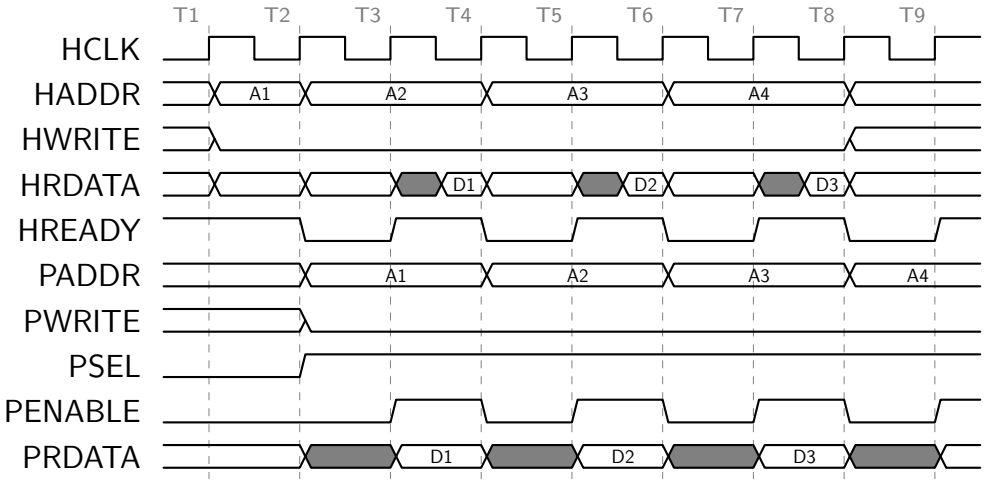


FIGURA 2.13: Transferencia de lectura al bus AHB en ráfagas.

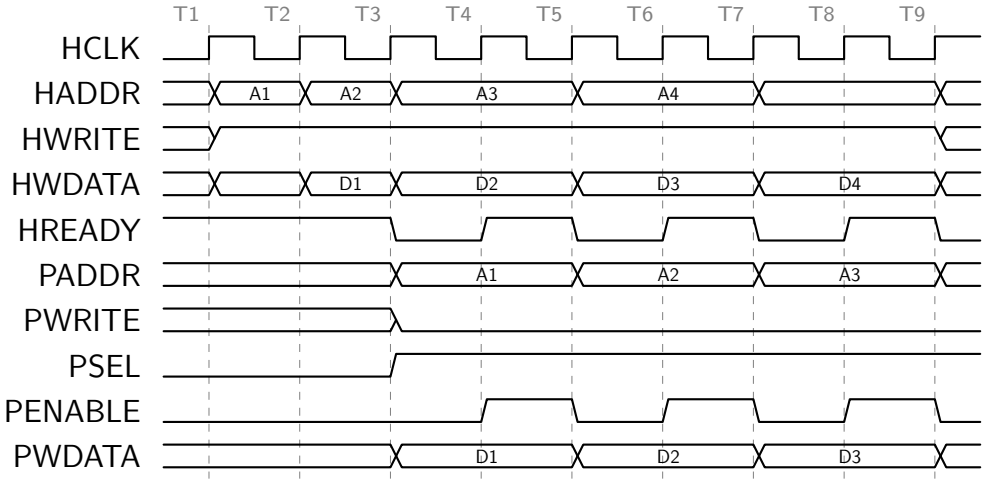


FIGURA 2.14: Transferencia de escritura desde el bus AHB en ráfagas.

- Decodificar la dirección y seleccionar el periférico que corresponda con la señal PSEL.
- Llevar el dato al APB durante una transferencia de escritura.
- Llevar el dato del APB al bus del sistema durante una transferencia de lectura.
- Generar la señal de sincronismo PENABLE necesaria para la transferencia.

2.5.1. Diseño del puente APB-AHB

El diseño del *bridge* cuenta con las señales AHB y una lógica para generar, con el debido temporizado, las señales hacia los esclavos APB. A su vez, realiza la decodificación de la dirección y el control de las señales de respuesta de los periféricos.

En la Figura 2.15 se muestra un diagrama en bloques del diseño del puente APB-AHB, con las señales que soportan los periféricos diseñados.

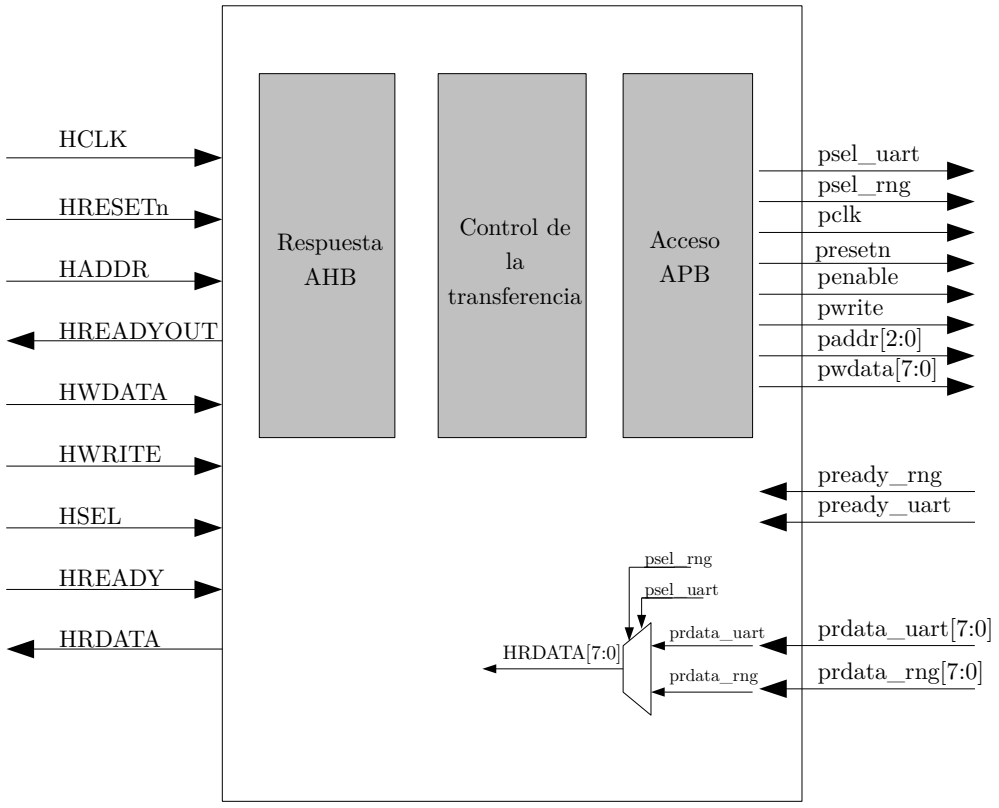


FIGURA 2.15: Diagrama del puente AHB-APB.

Para el diseño en particular, se realizan las siguientes consideraciones con respecto a las señales que intervienen en la transferencia:

- Solo existe un periférico AHB y es el *bridge*, por lo que en este caso HREADY y HREADYOUT son iguales.

- La señal HRDATA se encuentra multiplexada por las señales PRDATA de la UART y del generador.
- *psel_uart* es la señal de selección del esclavo APB correspondiente a la UART.
- *psel_rng* es la señal de selección del generador de números aleatorios.
- La señal de reloj *pclk* es la misma señal que HCLK del AHB.
- La señal *presetn* es la misma señal que HRESETn del bus AHB.
- La señal *pwdata* contiene el dato a escribir en el periférico. En el diseño, el único periférico que soporta una operación de escritura es la UART.
- Por medio de la señal pready_rng el generador ingresa estados de espera cuando todavía no ha generado un nuevo número aleatorio.
- Por medio de la señal *pready_uart* el esclavo APB inserta estados de espera. En el caso de la UART, la misma cumple con las transferencias en dos ciclos de reloj por lo tanto esta señal siempre está en 1 y no se considera en la implementación del *bridge*.
- La señal *prdata_uart* es el dato leído desde la UART, cuando este dispositivo se encuentra seleccionado y direccionado.
- La señal *prdata_rng* es el dato leído desde el generador de números aleatorios.

A diferencia del bus AHB, el bus APB no implementa *pipeline*. Debido a esto, se insertan estados de espera durante transferencias donde el bus AHB tiene que esperar al APB. La descripción VHDL del *bridge* se realizó implementando una máquina de tres estados, **IDLE**, **SETUP** y **ACCESO** de manera similar a la que controla las transferencias APB. Del estado **IDLE** se transiciona al estado **SETUP** siempre que haya una transferencia. Existe una transferencia siempre que la señal HSEL y HREADY se encuentren en 1. El estado **SETUP** tiene una duración de un ciclo, y en el siguiente flanco de reloj se transiciona al estado **ACCESO**. Se puede permanecer en este estado una cierta cantidad de ciclos según la señal PREADY, la cual si se encuentra en 0 inserta estados de espera.

Los cambios de estados son controlados por flancos positivos de la señal de reloj del bus. En el caso del diseño, los dos sistemas utilizan el mismo reloj y el dato de HWDATA pasa directamente a la señal PWDATA.

El control de la salida HREADYOUT se realiza considerando el estado siguiente de la máquina de estados. HREADYOUT es una versión registrada de la señal HREADY_next. Esta última toma valores según el siguiente estado como se muestra en la Figura 2.16. Cuando el siguiente estado es **SETUP**, HREADY_next es 0, y cuando es **ACCESO**, toma el valor de la señal PREADY para contemplar estados de espera agregados por el esclavo APB.

La señal HREADY_next es 0 cuando el siguiente estado es **SETUP**, y es PREADY cuando el siguiente estado es **ACCESO**. En el caso que se muestra en la figura, se entiende que la señal PREADY es 1, es decir que el esclavo no agrega ciclos de espera, caso contrario la fase de acceso no estaría restringida a la espera de un solo ciclo. Agregando estados de espera, se elimina la característica de bus *pipelined* que tiene el bus AHB.

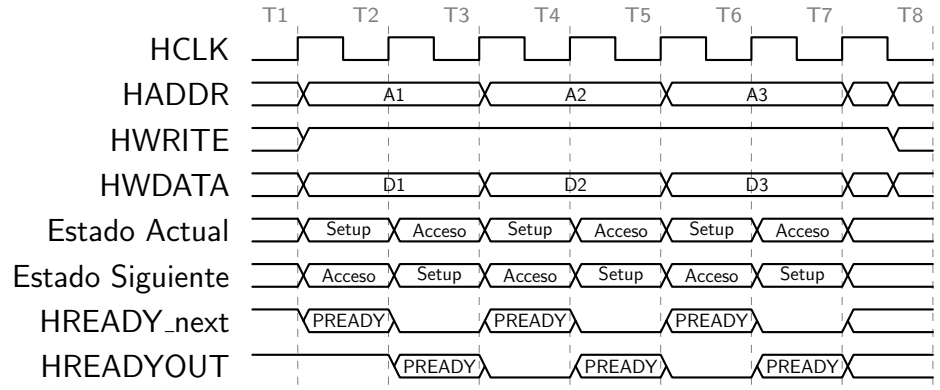


FIGURA 2.16: Control de la señal HREADY.

El control de las señales de dirección y control se muestra en la Figura 2.17. Dos señales se utilizan para controlar el pase de la dirección de un bus a otro dentro del *bridge*. Estas señales son APBen, que es 1 cuando el siguiente estado del bus es SETUP, y es 0 en caso contrario. La otra señal es HADRR_mux. Esta señal es la salida de un multiplexor, siendo HADDR cuando el siguiente estado es **SETUP**, y last_HADDR en caso contrario. Esta última señal es la versión registrada de

HADDR siempre que el esclavo AHB esté seleccionado y la señal HREADY sea 1; si la señal HREADY es 0, se mantiene en last_HADDR su valor.

Como puede observarse en la Figura 2.17, cuando APBen toma el valor 1 se habilita el registro que pasa a la salida PADDR el dato de HADDR_mux.

La señal de control PENABLE del bus APB también se controla por medio del *bridge*. La señal PENABLE_next es 1 cuando el siguiente estado es **ACCESO**, caso contrario es cero. La señal PENABLE es una versión registrada de esta señal.

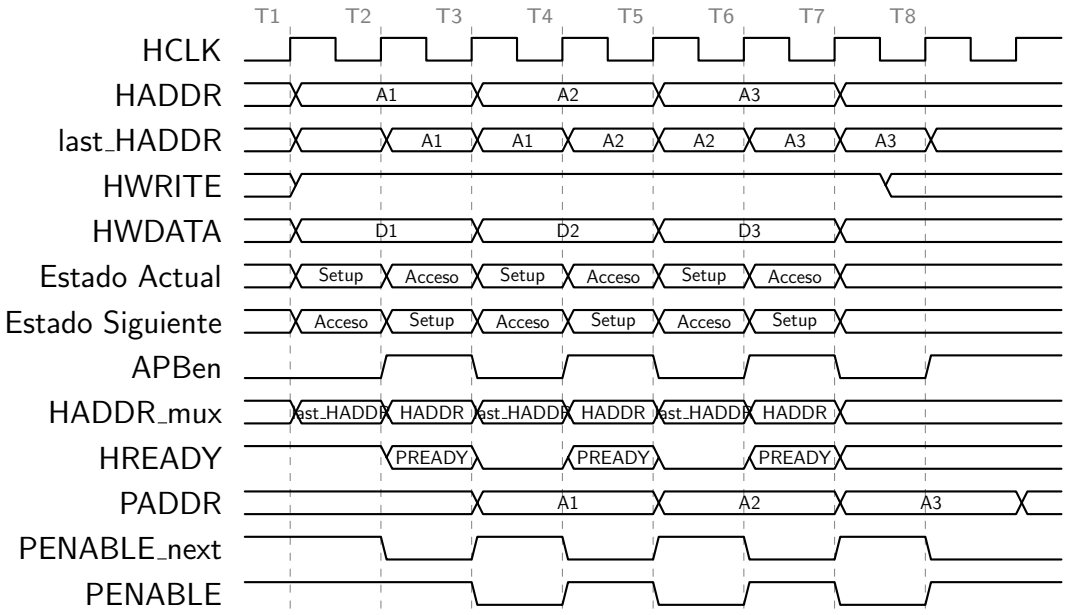


FIGURA 2.17: Control de las señales de dirección y control.

2.5.2. Simulaciones del puente AHB-APB

En esta sección se muestran las simulaciones del diseño del puente AHB-APB.

En la Figura 2.18 se muestra la simulación de las señales de la interfaz APB de un dispositivo. La transferencia comienza cuando se selecciona el dispositivo con la señal *psel2* en 1 y la señal *paddr* = "111". Luego de la fase de dirección, la transferencia se extiende porque *pready_rng* vale 0 y finaliza cuando en el flanco positivo de reloj se muestrea el dato leído, en este caso "00100111" y la señal *pready_rng* se encuentra en 1.

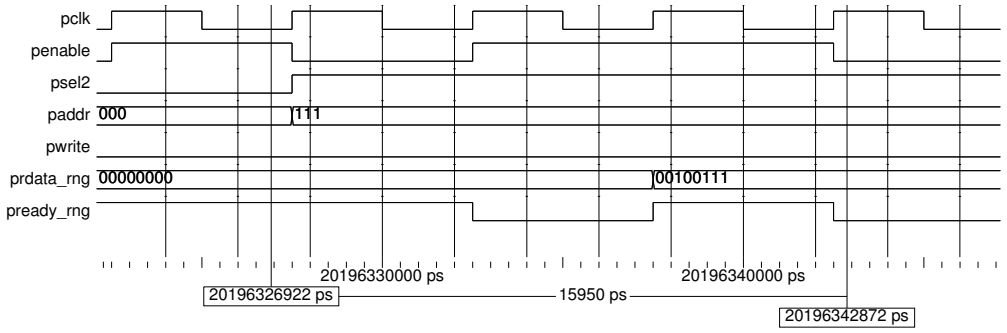


FIGURA 2.18: Simulación del *bridge* APB para una transferencia de lectura al dispositivo seleccionado con la señal *psel2*.

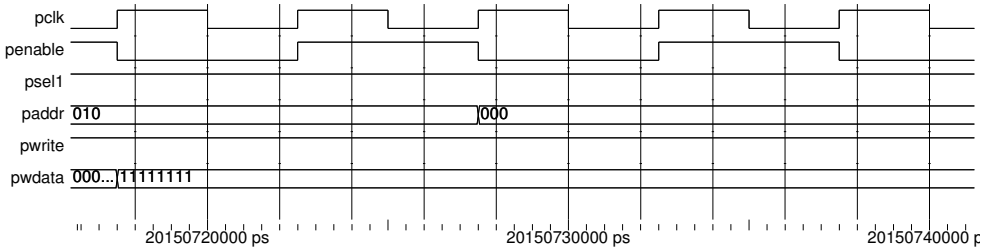


FIGURA 2.19: Simulación del *bridge* APB para una transferencia de escritura al dispositivo seleccionado con la señal *psel1*.

En la Figura 2.19 se muestran dos transferencias de escritura a un dispositivo conectado al bus cuya señal de selección es *psel1*. En la primer transferencia se escribe la dirección "010", que puede corresponder, por ejemplo, a un registro del dispositivo. Luego se realiza otra transferencia a la dirección "000", que corresponde a otro registro del mismo dispositivo.

En la Figura 2.20 se muestra la lectura de un dispositivo seleccionado por la señal *psel1* en 1 y la dirección *haddr* = "001". Esta transferencia se ve extendida por una transferencia previa del bus a través de la señal *hready*. La transferencia culmina en el quinto flanco de reloj cuando se lee desde el APB el dato "11001000". Luego se realiza una transferencia de lectura a la dirección *paddr* = "000" del esclavo seleccionado.

Las operaciones de lectura del dispositivo seleccionado por la señal *psel2* en general se extienden por más de un ciclo con la señal *pready*. Esta señal se transfiere al *bridge* por medio de la señal *hready* como se muestra en la Figura 2.21. Se observa que la fase de dirección, cuando *haddr* es "111", se encuentra extendida por una

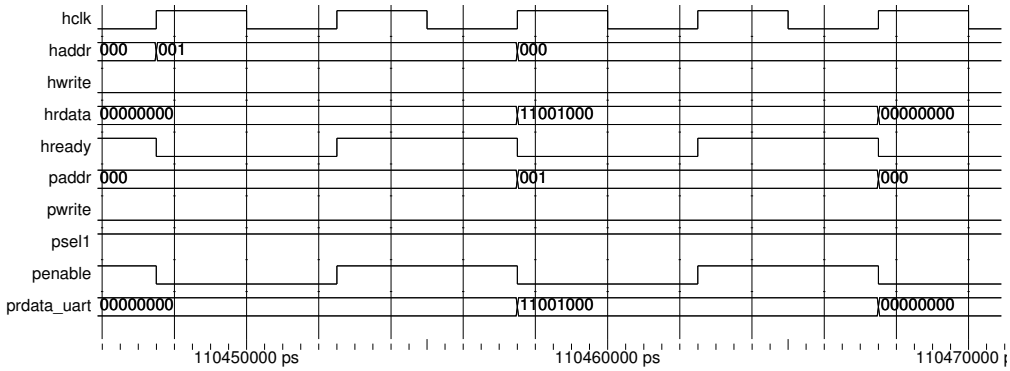


FIGURA 2.20: Simulación de una transferencia de lectura al dispositivo seleccionado con la señal *psel1*.

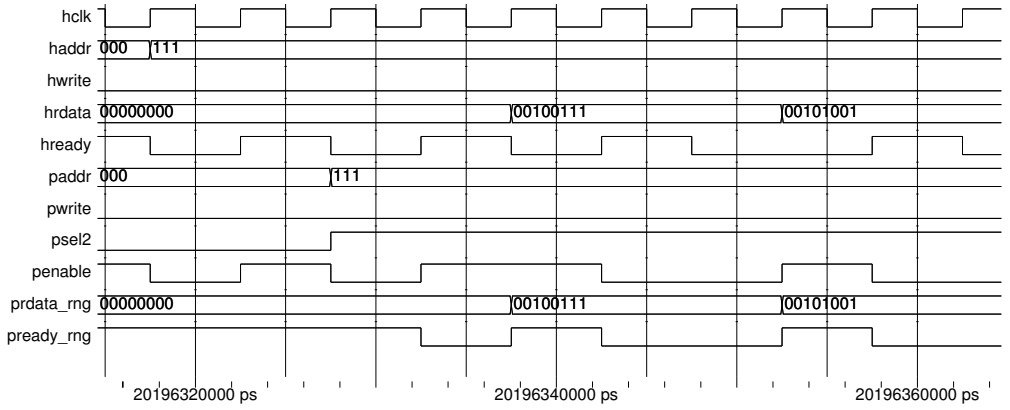


FIGURA 2.21: Simulación de una transferencia de lectura del dispositivo seleccionado con la señal *psel2*.

transferencia previa. Por tal motivo, la transferencia tiene un ciclo adicional. En el flanco positivo, donde *paddr* comienza a valer "111", empieza la fase de setup de la transferencia APB. Luego, en el próximo flanco, la fase de acceso se extiende un ciclo de reloj, debido a que el esclavo coloca en 0 la señal *pready_rng*. En el flanco donde la señal *penable* es 1 finaliza la transferencia leyendo el dato "00100111". Luego se realiza otra lectura en la misma dirección, culminando la transferencia cuando en el flanco de reloj, *hrdata* contiene el valor "00101001". Las señales *pready* o *pready_rng* pueden tomar cualquier valor cuando la señal *penable* es 0.

Capítulo 3

Generador de números aleatorios

Un generador de números aleatorios es un dispositivo físico o computacional diseñado para generar una secuencia de números o símbolos sin un orden aparente.

Los generadores de números aleatorios se utilizan en múltiples aplicaciones como pueden ser simulaciones y modelado, juegos de computación, criptografía, generación de claves y muestreo aleatorio. Debido a esta diversidad, existen dos estrategias para generar números aleatorios: una de ellas es producir bits en forma no determinística, donde cada bit de salida surge de un proceso físico que es impredecible; esta clase de generador se denomina no determinístico o real. La otra estrategia es computar bits de manera determinística, utilizando un algoritmo; esta clase de generador es conocida como pseudo aleatorio o determinístico. Un sistema puede utilizar ambas estrategias funcionando en forma conjunta.

Un generador de números aleatorios real utiliza una fuente no determinística, y un bloque adicional para corregir errores provenientes de la fuente de entropía. Estos errores se ven reflejados en la generación de números que no son completamente aleatorios. En el caso de este tipo de generadores, la entropía se puede obtener a partir de fuentes físicas o fuentes externas. Por otro lado, un generador de números aleatorios determinístico funciona aplicando un algoritmo sobre una semilla inicial.

Al generar una secuencia de números aleatorios, es necesario validarla. La secuencia debe tener una distribución uniforme, es decir, la frecuencia de unos y ceros debe

ser aproximadamente igual. Además, no puede ser inferida a partir de otra, en otras palabras debe existir independencia entre ellas. Existen pruebas definidas para determinar que una secuencia de bits tiene una distribución en particular, pero no existe ninguna específica para asegurar independencia. Por este motivo, se realizan diferentes tipos de pruebas que, en conjunto, aportan una condición para asegurarla.

Intuitivamente se establecería que los números aleatorios obtenidos desde un generador deberían manifestar todos los posibles valores con la misma probabilidad y ser totalmente independientes de sus valores pasados y sus valores futuros. Sin embargo, estos requerimientos son muy restrictivos y corresponderían a características que deberían cumplir los generadores ideales. Por este motivo, dependiendo de la aplicación a la que se dedicará el generador, deberá cumplir alguna o todas de las siguientes propiedades (Killmann y Schindler, 2011):

R1 Debe poseer propiedades estadísticas adecuadas a la aplicación.

R2 El conocimiento de subsecuencias de números aleatorios no debería permitir adivinar predecesores o sucesores de estos números con mayor probabilidad que sin el conocimiento de las mismas.

R3 El conocimiento del estado interno no debería prácticamente permitir computar viejos números aleatorios o un estado interno previo, con mayor probabilidad que sin el conocimiento del mismo.

R4 El conocimiento del estado interno no debería permitir computar o adivinar el siguiente número aleatorio con mayor probabilidad que no teniendo el conocimiento del mismo.

3.1. Clasificación de generadores de números aleatorios

Los generadores de números aleatorios se pueden clasificar en dos clases principales (Koç, 2009). La primera consiste en aquellos que poseen de una semilla inicial y generan números pseudoaleatorios utilizando un algoritmo determinado. Son llamados generadores determinísticos o pseudo aleatorios (DRNG, del inglés *Deterministic Random Number Generator*). La seguridad de los mismos dependerá de la complejidad computacional de posibles atacantes. La segunda clase abarca a los generadores reales, los cuales a su vez se pueden dividir en dos subclases: los físicos (PTRNG, del inglés *Physical True Random Number Generator*) y no físicos (NPTRNG, del inglés *Non Physical True Random Number Generator*). Los primeros, utilizan efectos no determinísticos de los circuitos integrados o fenómenos físicos; mientras que los NPTRNG utilizan eventos no determinísticos como la interacción del usuario. En el caso de los generadores reales, su seguridad dependerá de la impredecibilidad de la salida generada. En ambos casos, se pueden mencionar los generadores híbridos los cuales contienen elementos de un generador determinístico y de uno real. Si únicamente están conformados por elementos de su clase son denominados puros. En la Figura 3.1 se muestra una clasificación general de los tipos de generadores mencionados.

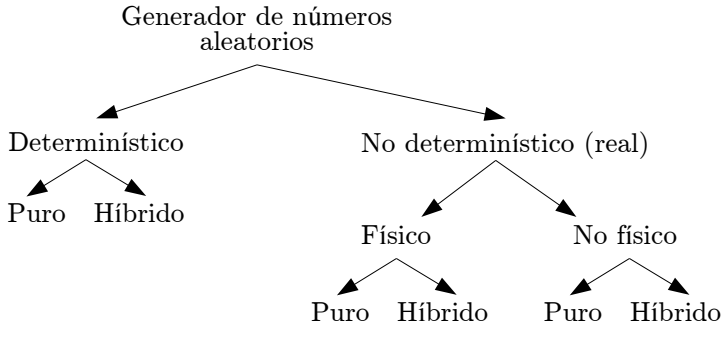


Figura 3.1: Clasificación de generadores de números aleatorios.

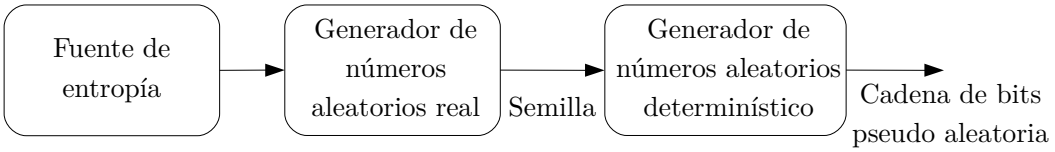


FIGURA 3.2: Esquema de un generador de números aleatorios.

3.1.1. Generador de números aleatorios determinístico

Generan la secuencia de bits a través de un algoritmo determinístico cuya ejecución depende de la entrada inicial, denominada semilla, en general provista por un generador de números aleatorios real. Esta semilla debe ser aleatoria e impredecible.

A continuación se mencionan los requisitos que debería cumplir un generador de números pseudoaleatorios:

- Tener una distribución uniforme de valores.
- No tener un patrón de valores detectables, es decir, generar números sin correlación entre números sucesivos.
- No tener condiciones iniciales débiles, que pueden generar secuencias de longitud cortas.

En la Figura 3.2 se muestra un esquema de un generador de números aleatorios propuesto en el documento desarrollado por el Instituto Nacional de Estándares y Tecnología (Bassham III. y col., 2010).

Generador de números aleatorios determinístico puro

En la Figura 3.3 se presenta un diagrama en bloques de un diseño genérico de un generador determinístico puro. Una vez que se generan $n-1$ números aleatorios, denominados $r_1, r_2, ..., r_{n-1} \in R$, el estado interno del generador es $S_n \in S$.

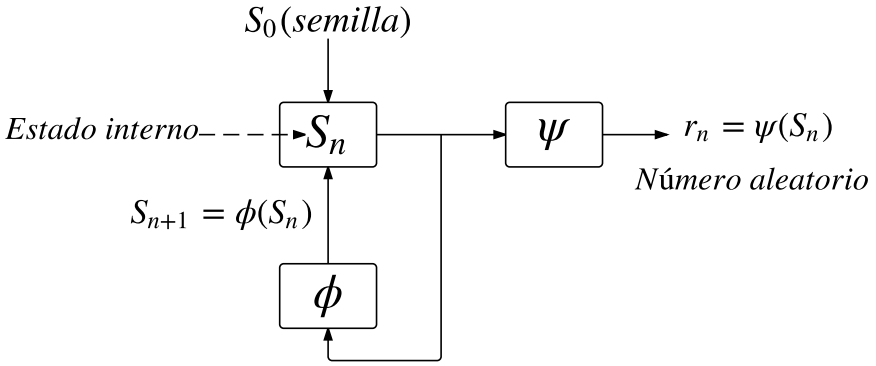


FIGURA 3.3: Diseño genérico de un generador de números aleatorios determinístico puro.

De aquí en adelante, S se denomina el espacio de estados y R el espacio de salida. La función de transición de salida $\psi : S \longrightarrow R$ es la que se encarga de obtener el siguiente número aleatorio r_n a partir del estado interno actual S_n. Luego, S_n se actualiza al siguiente estado S_{n+1} con la función de transición de estado ϕ, es decir, $S_{n+1} = \phi(S_n)$. El primer estado interno S_1 puede ser derivado a partir de la semilla S_0 realizando $S_1 = \phi(S_0)$. De esta manera, un generador determinístico puro se puede describir por el conjunto de las funciones

$$(S, R, \phi, \psi, p_s),$$

donde p_s es la distribución de probabilidad de la semilla.

Existen distintas maneras de generar números pseudoaleatorios por métodos determinísticos. Una de ellas es a partir de un generador denominado lineal congruencial, el cual utiliza una técnica basada en aritmética modular. Su función de transición de estado es:

$$S_{n+1} = (a\ S_n + b) \bmod m, \tag{3.1}$$

y el número aleatorio r_n está dado por

$$r_n = \frac{S_n}{m}, \tag{3.2}$$

donde S_n es el estado interno actual, S_{n+1} es el siguiente estado y las constantes a, b y m son el multiplicador, el incremento y el módulo, respectivamente. La semilla inicial es el valor S_0. Las características de las secuencias generadas por el lineal congruencial dependerán del valor elegido para las constantes a, b y m. Si se elijen los valores de b y m de forma que el único divisor común entre ellos es 1, es decir que b es relativamente primo a m, se genera el mayor ciclo antes que la secuencia se vuelva a repetir. De esta manera se obtiene una secuencia denominada de máxima longitud.

Otras formas de generar números pseudoaleatorios son a partir de los generadores cuadráticos y cúbicos cuyas funciones de transición de estado son:

$$S_{n+1} = (a\ S_n^2 + b\ S_n + c) \bmod m, \tag{3.3}$$

$$S_{n+1} = a\ S_n^3 + b\ S_n^2 + c\ S_n + d) \bmod m, \tag{3.4}$$

respectivamente. Una cuarta forma de generar números pseudoaleatorios es a partir de registros realimentados muy utilizados en aplicaciones de criptografía y cifrado de cadenas. Se encuentran constituidos por un registro de desplazamiento y una función de realimentación. Cada vez que se necesita un bit, se desplazan todos los bits del registro de a uno por vez. La salida del registro es de un bit, y frecuentemente se toma el menos significativo de la cadena. Tiene la ventaja con respecto a otros generadores que son simples de implementar en hardware digital.

El registro realimentado más simple es el lineal o LFSR (del inglés, *Linear Feedback Shift Register*). La función de realimentación que utiliza es la suma binaria que se implementa con una compuerta XOR, de allí proviene su denominación lineal. En este tipo de generadores, dicha función se aplica solo a determinados bits del registro de desplazamiento, dando como resultado un LFSR con cierta cantidad de etapas. En la Figura 3.4 se muestra un ejemplo particular de un LFSR de 4 bits. Los bits del registro de desplazamiento pueden representarse por un vector binario, y el mismo por un polinomio con coeficientes binarios. Un LFSR tiene $2^n - 1$ estados posibles, debido a que el cero no se encuentra incluido, y genera

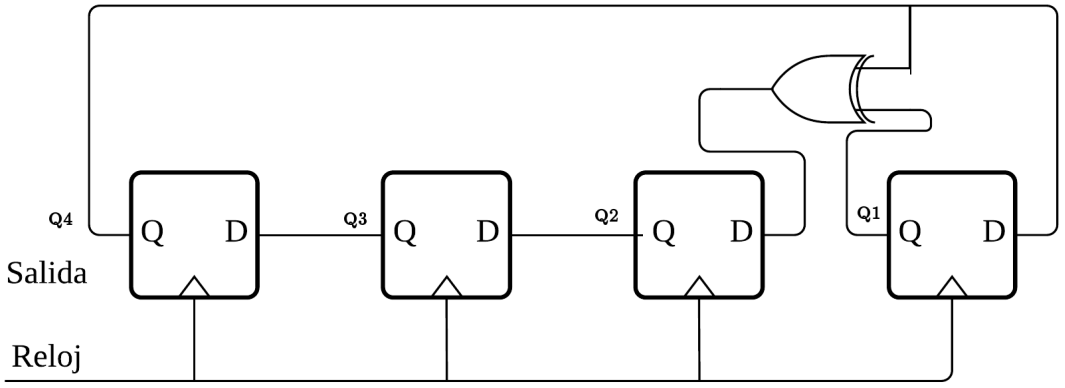


FIGURA 3.4: Registro de desplazamiento de realimentación lineal de 4 bits.

una secuencia pseudoaleatoria de $2^n - 1$ bits. Si se iniciara con ceros, la secuencia quedaría fija en este valor.

Como en los generadores congruenciales lineales, no todas las etapas generan secuencias de máxima longitud. Si el polinomio que representa a las etapas cumple con características especiales, se consiguen este tipo de secuencias denominadas "secuencias m".

El LFSR de la Figura 3.4 se puede representar por el polinomio $x^4 + x + 1$. El término x^4 representa la salida Q4 del último flip flop. x representa la función XOR entre el primer y el segundo flip flop. Por último, el término $x^0 = 1$ corresponde a la realimentación en la entrada D del primer flip flop. Este LFSR generará una secuencia de $2^4 - 1 = 15$ bits.

Otro tipo de generadores pseudo aleatorios son los registros de realimentación con acarreo (FCSR, del inglés *Feedback Carry Shift Register*). Están formados por un registro de desplazamiento, una función de realimentación y un registro de acarreo. La salida del FCSR se obtiene realizando la suma binaria de los bits de las etapas y del contenido del registro de acarreo.

Por último, existe otro tipo de generadores, muy utilizados en aplicaciones de criptografía, cuya seguridad se basa en la complejidad que significa factorizar grandes enteros. De los más conocidos se pueden mencionar Blum-Blum-Shub DRNG que requiere la exponencial modular de un entero de 2^m bits, RSA y Rabin (Schindler, 2009).

Una desventaja de los generadores determinísticos con respecto a los reales, es que su salida se encuentra totalmente determinada por la semilla y que los números aleatorios futuros dependen únicamente del estado interno actual, siendo necesario mantener la protección de este estado. Por el contrario, las ventajas de implementar un generador determinístico son su menor costo en implementación y no necesitan de hardware dedicado. De todas formas, la elección del generador dependerá de la aplicación para la cual se lo utilice. Por ejemplo, en contextos de simulación puede ser necesario generar la misma secuencia de números aleatorios de manera repetitiva.

Generador de números aleatorios determinístico híbrido

Los DRNG computan el número aleatorio de salida $r_n = \psi(S_n)$ y actualizan el estado interno $S_n \rightarrow \phi(S_n)$. Un generador híbrido posee, además de la semilla, una entrada externa adicional, brindando una fuente de entropía adicional en relación al pseudo aleatorio puro. Para describir un generador de este tipo se utilizan 7 variables $(S, R, E, \phi_H, \psi_H, p_s, (q_n)_{n\in N})$, donde E es el conjunto de datos adicional de entrada y $(q_n)_{n\in N}$ las distribuciones de probabilidad del dato adicional. Si la secuencia de entrada adicional $e_1, e_2, ...$ es constante o completamente conocida por un atacante, esto no reduce la seguridad del híbrido por debajo de la seguridad del determinístico puro, en el peor de los casos son iguales. Por otro lado, la entrada adicional aumentará la seguridad del generador dependiendo de su propiedades de aleatoriedad e impredecibilidad. En la Figura 3.5 se muestra un esquema del diseño de un generador con estas características.

Para algunas aplicaciones de generadores determinísticos, puede ser necesario que el mismo cumpla con el requerimiento R4, lo cual no se puede lograr si se conoce el estado interno. Sin embargo, utilizando un híbrido con una entrada adicional aleatoria, R4 podría cumplirse.

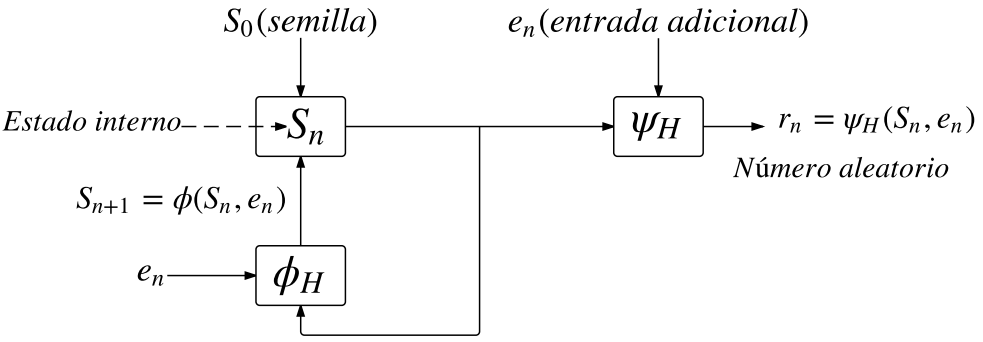


FIGURA 3.5: Diseño genérico de un generador de números aleatorios determinístico híbrido.

3.1.2. Generador de números aleatorios real

A diferencia de los generadores determinísticos, los generadores de números aleatorios reales buscan obtener en su salida secuencias de números aleatorios ideales con la pérdida de velocidad en la generación, comparada con los primeros. Con respecto a la seguridad, la misma recae en la impredecibilidad de los números aleatorios generados.

Los generadores de números aleatorios reales se pueden clasificar por el tipo de fuente de entropía que utilizan. Existen aquellas que generan entropía a través de un proceso físico microscópico como la desintegración atómica radioactiva. También se puede obtener desde fuentes analógicas como la entropía resistiva térmica (ruido blanco), la entropía obtenida desde la ruptura de un diodo o a partir del muestreo de señales provenientes de osciladores. Los generadores que utilizan este tipo de fuentes son llamados físicos.

Por otro lado se encuentran los no físicos (NPTRNG), que utilizan una señal externa como fuente de entropía para generar la secuencia de bits aleatorios. Estas fuentes pueden ser operaciones de entrada salida de CDs, operaciones internas del sistema y datos provistos por el mismo, o la interacción humana como el movimiento de un mouse o el accionamiento de teclas del teclado de una computadora.

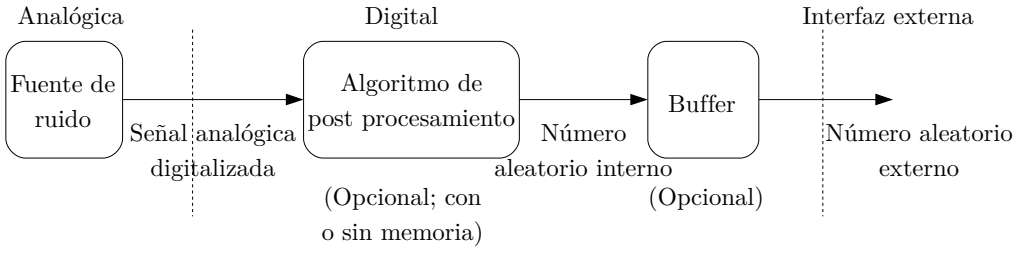


FIGURA 3.6: Configuración básica de un generador de números aleatorios real físico.

Generador de números aleatorios físico

En la Figura 3.6 se muestra un diseño genérico de un TRNG. El núcleo principal es una fuente de ruido. Estas fuentes de ruido generan señales analógicas de tiempo continuo, que son digitalizadas en una de las etapas, convirtiéndose a valores binarios. Estos números aleatorios pueden ser procesados internamente para reducir imperfecciones, obteniendo los números aleatorios internos. El algoritmo de post procesamiento puede ser sin memoria, es decir que sólo depende de los bits actuales, o podrían combinarse con los bits predecesores o con algún otro parámetro secreto.

A pesar de que existen varias implementaciones de TRNG, la mayoría se encuentran constituidos por tres componentes principales:

- Fuente de entropía: es uno de los componentes más importantes de un TRNG. Puede obtenerse a partir de procesos físicos como son ruido térmico o ruido shot, *jitter* y metaestabilidad de circuitos, ruido atmosférico o fenómenos radioactivos. Debe considerarse que no todas las fuentes son adecuadas para cada una de las posibles aplicaciones, y que hay fuentes que contienen una desviación hacia un determinado valor lo cual debe corregirse con un mecanismo de post procesamiento.

- Mecanismo de recolección: la fuente de entropía debe ser recolectada por algún mecanismo que, idealmente, no debería perturbar el proceso físico y brindar la mayor entropía posible.

- Post procesamiento: Este componente no es necesario en todos los diseños, aunque se considera una buena práctica. Su objetivo es proporcionar mayor robustez al diseño. Algunas de las funciones de este bloque son eliminar, si existe, la tendencia de los números generados hacia un determinado valor, la correlación entre muestras, y aumentar la seguridad ante posibles ataques. Un ejemplo puede ser el corrector de Von Neumann (Neumann, 1963).

Generador de números aleatorios real no físico

Un generador de números aleatorios no físico utiliza señales externas como fuente de entropía. Una de estas fuentes puede ser la interacción humana como el movimiento del mouse. Un diseño genérico de este tipo de generadores se ilustra en la Figura 3.7. La entropía de los bits de salida (números aleatorios sin procesamiento) generados por la fuente es, en general, baja con respecto a los generadores físicos. Como consecuencia, en el diseño es necesario adherir complejos algoritmos de post procesamiento. La fuente de entropía no es controlada por el diseñador y depende del sistema utilizado o del propio usuario, implicando diferencias en la evaluación de la seguridad con respecto a un generador físico. En general están destinados a ser una implementación en software en una computadora para que su salida sea utilizada directamente, o ser la semilla para un generador determinístico.

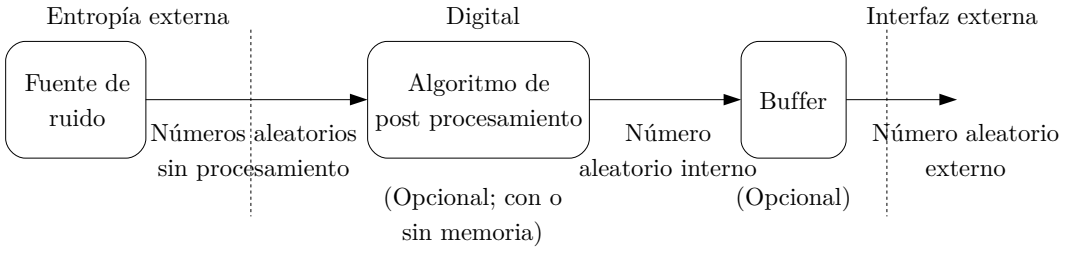


FIGURA 3.7: Diseño genérico de un generador de números aleatorios real no físico.

3.2. Estándares y guías de evaluación

Se encuentran en la bibliografía distintos estándares y guías de evaluación que definen las propiedades que debería tener un generador de números aleatorios robusto. En la guía propuesta por Killmann y Schindler (2011) se definen y explican criterios de evaluación para los generadores de números aleatorios en general, y se detalla cómo deberían verificarse estos criterios. Distintas guías de evaluación y estándares especifican diseños y dan ejemplos para generadores de números aleatorios determinísticos (ISO (2011), NIST (2000), Barker, Kelsey y col. (2012), Schindler (1999)). En el caso de los generadores reales, es difícil especificar un diseño único debido a que el resultado dependerá de la implementación y el resultado de la evaluación se obtendrá de la realización en particular. Existen varias metodologías para evaluar la calidad de un PTRNG. Por ejemplo, la Oficina Federal de Seguridad de la Información (BSI del alemán *Bundesamt für Sicherheit in der Informationstechnik*) propuso un método denominado "Clases funcionales y metodología de evaluación para generadores de números aleatorios físicos" (Killmann y Schindler, 2001).

Como se mencionó, la seguridad de un generador determinístico recae en la complejidad de la función de transición de estado y en la función de salida. Por lo tanto, su seguridad podría variar a lo largo del tiempo si la potencia computacional de un atacante se incrementa o si se detectan debilidades en el algoritmo utilizado. En consecuencia, si se quiere mantener la seguridad a lo largo del tiempo se debería utilizar un generador de números aleatorios real.

La calidad de un generador está muy relacionada con la calidad de la fuente de aleatoriedad y con el método de extracción utilizado. La característica del espectro de la fuente de aleatoriedad y del método de extracción determina los principales parámetros de la cadena de bits de salida: la desviación de los bits de salida hacia un determinado valor y la correlación entre bits subsecuentes. A pesar de que ambas fallas pueden ser remediadas con un buen bloque de post procesamiento, es importante que el generador produzca una cadena de bits de calidad.

Todos los estándares y las guías de evaluación mencionadas se encuentran diseñadas con el fin de detectar características no aleatorias en la salida de un generador, realizando una comparación entre muestras de un generador real contra distribuciones esperadas de una fuente de entropía perfecta.

3.2.1. Pruebas estadísticas desarrolladas por el NIST

El NIST (*National Institute of Standards and Technology*), desarrolló un conjunto de 16 pruebas estadísticas para generadores de números aleatorios y pseudoaleatorios a ser utilizados en aplicaciones criptográficas. Estas pruebas se describen en las distintas versiones del estándar NIST (1994), NIST (2001) y NIST (2009).

En el caso de las pruebas estadísticas propuestas por el NIST se especifican dos hipótesis: una de ellas es que la secuencia sea aleatoria (denominada H_0 o hipótesis nula), y la segunda, denominada H_a o hipótesis alternativa, es que la secuencia no lo sea. Para cada prueba, se selecciona una distribución de probabilidad conocida para determinar si se acepta o rechaza la hipótesis H_0. El procedimiento básico para la prueba de aleatoriedad se describe de la siguiente manera (Kim, Lee y Perrig, 2014):

1. Establecer la hipótesis nula y la hipótesis alternativa. Es decir: en la hipótesis nula H_0, la estadística de la secuencia de salida del generador sigue el patrón de la distribución de referencia, mientras que la hipótesis alternativa no lo hace.

2. Se computa el valor de la prueba estadística para la secuencia dada.

3. Se compara el valor dado por la prueba con un valor crítico. Si la aleatoriedad asumida es real, el valor de la prueba tendrá poca probabilidad de exceder el valor crítico.

El método más popular para comparar el valor de la prueba estadística con el valor crítico se denomina método del valor p. Un valor p es la probabilidad que

un generador de números aleatorios ideal produzca una secuencia menos aleatoria que la secuencia bajo verificación. Si el resultado de una prueba de aleatoriedad arroja un valor-p igual a 1, la secuencia es perfectamente aleatoria. Un valor p de 0 indica que no lo es.

Un buen generador de números aleatorios podría producir una secuencia que no sea aleatoria. Por otro lado, un generador de bajo nivel de seguridad podría producir una secuencia aleatoria. Por estas razones el NIST considera dos tipos de errores:

- Error tipo 1: una prueba estadística rechaza una secuencia que en realidad es aleatoria.
- Error tipo 2: una prueba estadística acepta una secuencia que en realidad no es aleatoria.

En las pruebas estadísticas desarrolladas por el NIST, un error del tipo 1 se denota con la letra α, y un error del tipo 2 con la letra β. Debido a estos errores, una prueba estadística no puede decidir con certeza cuando una secuencia es o no aleatoria. Es por esto que se emplea un nivel de aceptación para la prueba. Este nivel corresponde a la probabilidad de que ocurra un error tipo 1, es decir α. Si el *valor-p* $\geq \alpha$, la hipótesis nula se acepta. Si el *valor-p* $< \alpha$ la hipótesis nula se rechaza. Para aplicaciones criptográficas, α se encuentra entre el intervalo $[0{,}001; 0{,}01]$. Un α de 0,001 indica que se esperaría que de cada 1000 secuencias, una sea rechazada por la prueba, aún siendo aleatoria.

En la Tabla 3.1 se presentan los nombres de cada una de las pruebas estadísticas, junto con la distribución de referencia correspondiente y el tamaño recomendado que tiene que tener la secuencia de bits para aplicar la prueba. El paquete de pruebas estadísticas se encuentra desarrollado en ANSI C (Fischer y col., 2009).

El NIST especifica cinco etapas claves en la verificación de la calidad de un RNG:

1. Selección del tipo de generador a evaluar.

Tabla 3.1: Pruebas estadísticas establecidas por el NIST.

Prueba estadística	Distribución de referencia	Tamaño de bits
Frecuencia (monobits)	media-normal	$n \geq 100$
Frecuencia (bloques)	chi cuadrado	$n \geq 100$
Runs	chi cuadrado	$n \geq 100$
Cadena más larga de unos (bloque)	chi cuadrado	$n \geq 128$ hasta 750000
Rango de la matriz binaria[1]	chi cuadrado	$n \geq 38912$
Transformada de Fourier discreta	normal	$n \geq 1000$
Comparación de plantilla no solapada	chi cuadrado	$n \geq 10^6$
Comparación de plantilla solapada	chi cuadrado	$n \geq 10^6$
Test universal de Maurer	media-normal	$n = 387840$ hasta 10^9
Complejidad lineal	chi cuadrado	$n \geq 10^6$
Serie[2]	chi cuadrado	$m \leq [log_2(n)] - 2$
Entropía aproximada	chi cuadrado	$m \leq [log_2(n)] - 2$
Sumas Acumulativas	normal	$n \geq 100$
Excursión aleatoria	chi cuadrado	$n \geq 10^6$
Variante de excursión aleatoria	media-normal	$n \geq 10^6$

[1] También se encuentra en el conjunto de pruebas estadísticas Diehard.
[2] m corresponde a la longitud del bloque en bits.

2. Generación de la secuencia binaria: para una secuencia fija de longitud n y el generador pre seleccionado, construir un conjunto de m secuencias binarias y almacenarlas en un archivo.

3. Ejecutar el conjunto de pruebas estadísticas.

4. Examinar el valor p: se genera un archivo de salida con estos valores.

5. Interpretación: para un valor de aceptación fijo, un cierto porcentaje de valores p se esperan que fallen, por ejemplo si $\alpha = 0{,}01$, entonces el 1 % de las secuencias se espera que fallen. Una secuencia es correctamente validada por una prueba estadística siempre que el *valor-p* $\geq \alpha$, de otra manera se descarta. Para concluir en un resultado final, se examina la proporción de secuencias que pasan la prueba determinada o se observa la distribución de los valores p.

Como se puede notar de la Tabla 3.1, varias de las pruebas estadísticas tienen como distribución de referencia la normal estándar y la chi cuadrada. Si la secuencia que se encuentra bajo evaluación no es aleatoria, la prueba estadística calculada caerá en regiones extremas de la distribución de referencia. La distribución normal estándar se utiliza para comparar el valor del test estadístico obtenido desde el RNG con el valor esperado de la estadística bajo el supuesto de aleatoriedad. La estadística para la distribución normal estándar es de la forma $z = (x - \mu) \ / \ \sigma$, donde x es el valor de la muestra del test estadístico, y μ y σ^2 son su valor esperado y varianza respectivamente. La distribución chi cuadrado se utiliza para comparar como se adaptan las frecuencias observadas de una muestra medida con las frecuencias esperadas de la distribución de referencia. Su estadística es de la forma $\chi^2 = \sum((o_i - e_i)^2/e_i)$, donde o_i y e_i son respectivamente las frecuencias de ocurrencias observadas y esperadas. Es importante mencionar que la restricción en el tamaño de las secuencias viene dada por el hecho de utilizar distribuciones de referencias que aproximan la realidad; por lo tanto, si se utilizaran menos muestras se deberían además, utilizar distribuciones exactas.

3.2.2. Pruebas estadísticas DIEHARD

Las pruebas estadísticas DIEHARD fueron desarrollados en 1996 por el Profesor George Marsaglia (1996) para la verificación de aleatoriedad de secuencias de números aleatorios.

Consiste en 15 pruebas estadísticas distintas e independientes. Como en el caso de los desarrollados por NIST, los tests DIEHARD dan como resultado valores p, los cuales deberían ser uniformemente distribuidos para obtener el resultado esperado. En relación a los primeros, los DIEHARD utilizan distintos rangos para los valores p.

A continuación se presenta en forma resumida cada una de las pruebas estadísticas:

- Birthday: se seleccionan puntos aleatorios dentro de un intervalo de cierta longitud de bits. La distancia entre los puntos debería tender a una distribución de Poisson.

- Permutaciones superpuestas: se analizan secuencias de cinco números aleatorios consecutivos y se espera que las 120 posibles permutaciones ocurran estadísticamente con igual probabilidad.

- Rango de matrices: se calcula el rango de una matriz construida a partir de bits aleatorios.

- Monkey: se considera un determinado número de bits como una palabra y el test calcula la cantidad de palabras que se solapan.

- Conteo de unos: cuenta palabras de cinco letras determinadas por el número de unos en una cadena de bytes.

- Estacionamiento: consiste en colocar de forma aleatoria círculos en un lugar determinado, y contar la cantidad de veces que no se colocó en el mismo lugar luego de una cierta cantidad de intentos.

- Distancia mínima: aleatoriamente se colocan 8000 puntos en un cuadrado de 10000×10000, luego se busca la mínima distancia entre pares cuyo cuadrado debería estar exponencialmente distribuido con una cierta media.

- Esferas aleatorias: se elijen al azar 4000 puntos en un cubo de lado 1000. En cada punto se centra una esfera del tamaño suficiente para alcanzar el siguiente punto más cercano. El volumen de la esfera más pequeña debería estar exponencialmente distribuido con cierta media.

- Prueba de compresión: se multiplica 2^{31} por números entre $[0, 1)$ hasta alcanzar 1, repitiendo 100000 veces el proceso, el número de números que se necesiten para llegar a 1 debería seguir una distribución uniforme.

- Sumas solapadas: se genera una secuencia de números aleatorios entre $[0, 1)$ y se suman secuencias de 100 números consecutivos. Dicha suma debería tener distribución normal.

- Runs test: se genera una secuencia de números aleatorios entre $[0, 1)$ y se cuenta el número de valores consecutivos.
- Dados: realiza 200000 juegos de dados y cuenta la cantidad de victorias y el número de jugadas para terminar el juego.

La mayor desventaja de las pruebas de aleatoriedad de DIEHARD es que para ejecutarlo de la forma correcta se requieren al menos 80 millones de bits. Por tal motivo, Marsaglia, Tsang y col., 2002 propusieron reducir el número de pruebas del DIEHARD a sólo tres con la ventaja de necesitar enteros aleatorios de 32 bits para llevarlos a cabo. Estas pruebas estadísticas se mencionan a continuación:

- La prueba de aleatoriedad del máximo común divisor. La misma se basa en el algoritmo de Euclides para calcular el máximo común divisor (MCD) de dos enteros aleatorios de 32 bits.
- El test "Gorilla", similar a la prueba de aleatoriedad Monkey correspondiente al DIEHARD.
- La prueba de aleatoriedad "birthday spacing" correspondiente al DIEHARD en una versión mejorada.

3.2.3. Guía de evaluación AIS 31

En el caso de los generadores de números aleatorios reales físicos, el organismo alemán BSI desarrolló una metodología de evaluación denominada AIS 31 (Killmann y Schindler, 2001). La ventaja de esta metodología con respecto a las ya mencionadas es que propone verificar la aleatoriedad de la secuencia a la salida de la fuente de entropía (ver Figura 3.6). Esta característica es interesante por dos motivos: en primer lugar, las señales analógicas digitalizadas tendrán mejores propiedades estadísticas que serán brindadas al algoritmo de post procesamiento; por otro lado si estas señales analógicas digitalizadas tienen buenas propiedades

estadísticas pueden ser utilizadas directamente sin la necesidad de un mecanismo de post procesamiento.

Otra noción que se adiciona en dicha metodología, es que corresponde a una prueba en línea, lo cual permite un control de calidad del generador mientras se encuentra en operación. Una prueba de este tipo se aplica a la señal de ruido digitalizada. Estas pruebas proveen una forma de detener el generador cuando se detecta una falla.

Con el fin de evaluar la calidad de un PTRNG, en el documento AIS 31 se definen dos clases funcionales denominadas P1 y P2. La clase funcional P1 se enfoca en el número aleatorio interno (fuente de ruido digitalizada y post procesamiento) (ver Figura 3.6), y requiere que las propiedades estadísticas del mismo no se vean influenciadas por condiciones externas como por ejemplo la temperatura. Por el otro lado, la clase funcional P2 se enfoca en garantizar que los números aleatorios generados sean prácticamente imposibles de determinar aún si se conocen valores pasados o futuros de los mismos.

Además de las clases funcionales, el estándar describe requerimientos complementarios para ambas: se especifica que se deben pasar distintas pruebas estadísticas; cuatro de ellas son idénticas a las descritas en el documento NIST, 1994 FIPS 140-1 y se agregan 5 adicionales. Entre ellas se pueden mencionar la prueba de discontinuidad la cual verifica que no se encuentren patrones no solapados en una secuencia; el test monobit o de frecuencia que evalúa que el número de unos sea aproximadamente igual al número de ceros; la prueba de aleatoriedad denominada póquer; el test Runs, que cuenta el número de unos o ceros consecutivos; la prueba de autocorrelación, que estudia si existe correlación entre una secuencia y la que surge de realizar un desplazamiento; la prueba de distribución uniforme; la de distribuciones multinomiales y la prueba de entropía. Algunas de estas pruebas de aleatoriedad se aplican directamente a la cadena de bits de salida del PTRNG y otras a la fuente de ruido interna.

3.3. Implementaciones de generadores de números aleatorios reales

En esta sección se realiza un breve comentario de los distintos generadores de números aleatorios reales más citados en la bibliografía.

3.3.1. Diseño de Bagini - Bucci

El diseño se basa en obtener muestras binarias cuantizadas por un comparador a partir de ruido blanco (Bagini y Bucci, 1999). El ruido blanco es una señal aleatoria que se caracteriza porque sus valores de señal en dos tiempos diferentes no guardan correlación estadística, es decir, en el eje del tiempo la señal toma valores que no tienen ninguna relación entre sí. Como consecuencia de ello, su densidad espectral de potencia es una constante.

Un esquema del circuito se puede observar en la Figura 3.8. En el diseño propuesto, la señal cuantizada se envía a un contador binario antes de ser muestreada con la finalidad de eliminar la desviación de los bits de salida hacia un cierto valor. Como fuente de entropía este diseño utiliza ruido blanco gaussiano obtenido a partir de una juntura semiconductora polarizada en forma directa. Este tipo de ruido es controlado por la corriente de polarización, pero su amplitud es baja y debe ser fuertemente amplificada. En la Figura 3.8 el generador de ruido real amplificado es representado por un generador de ruido ideal y un filtro pasa bajos. Se demuestra que la correlación entre muestras está controlada por la frecuencia de muestreo y la relación con el contador.

3.3.2. Diseño de Fischer - Drutarovský

Este diseño se basa en extraer aleatoriedad a partir del jitter de la señal de reloj generada por un PLL (del inglés, *Phase Locked Loop*) (Fischer y Drutarovskỳ, 2003). En este tipo de circuitos, varias fuentes de ruido causan que el VCO (del

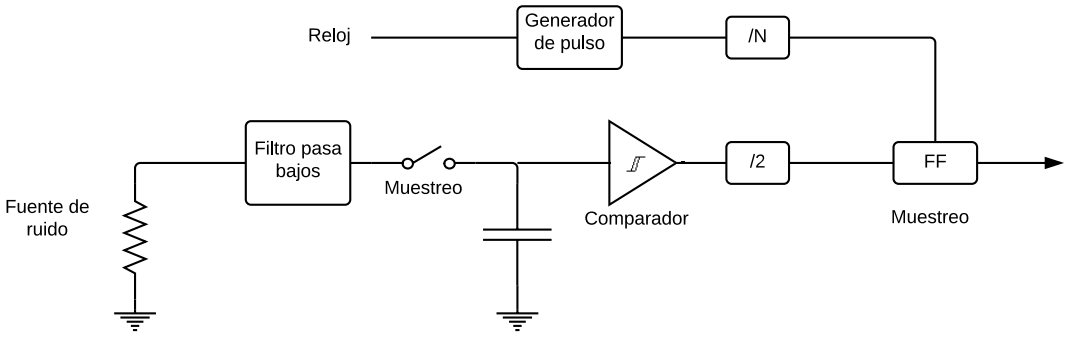


FIGURA 3.8: Diseño propuesto por Bagini y Bucci.

inglés, *Voltage Controlled Oscillator*) fluctúe en frecuencia. Un circuito de control interno, ajusta el VCO para que vuelva a la frecuencia especificada, lo que causa el jitter. En condiciones ideales, el jitter es no determinístico y causado sólo por fuentes de ruido internas, por tal motivo se denomina intrínseco. El esquema del principio de funcionamiento se muestra en la Figura 3.9. El reloj de salida del PLL posee jitter, y es muestreado por un flip flop D controlado por el reloj general del sistema. La señal muestreada contiene valores aleatorios con posibles desviaciones que son eliminadas utilizando un decimador XOR.

3.3.3. Diseño de Tkacik

La fuente de entropía de este diseño es el jitter en un oscilador de anillo (Tkacik, 2003). Un oscilador de anillo consiste en un número impar de inversores conectados en un lazo de realimentación, formando un anillo. Este TRNG se basa en un registro de desplazamiento lineal (LFSR), y un registro de desplazamiento autómata celular (CASR, del inglés *Cellular Automata Shift Register*). Cada registro de

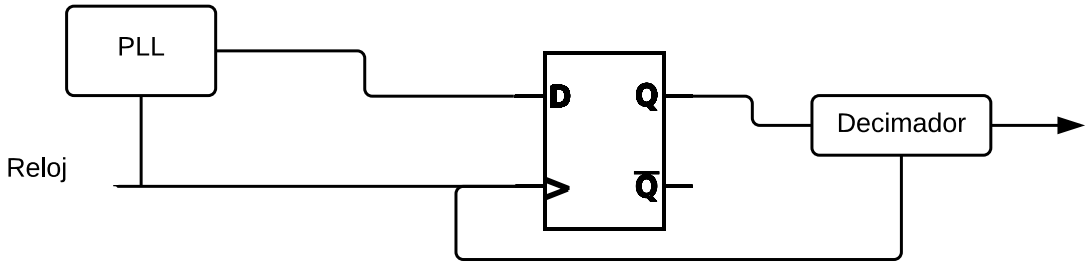


FIGURA 3.9: Diseño propuesto por Fischer y Drutarovský.

desplazamiento se alimenta con un oscilador de anillo independiente. El diseño de ambos registros de desplazamiento es tal que la longitud entre ellos sea relativamente prima. La salida del LFSR y del CASR se ingresan a una compuerta XOR. La salida solo se muestrea cuando se requiere un nuevo número aleatorio. La descripción del diseño se encuentra realizada en Verilog excepto los osciladores de anillos.

3.3.4. Diseño de Golić

La idea propuesta por Golić (2006) es similar a un registro de desplazamiento en donde cada registro es reemplazado por inversores. De la misma manera, las realimentaciones corresponden a un polinomio de cierto grado. Tiene la ventaja de ser un diseño completamente digital. Este esquema utiliza los inversores como elementos de retardo. Las salidas de un oscilador tipo Galois se combina a través de una XOR con la salida de un oscilador Fibonacci.

3.3.5. Diseño de Sunar

Es un diseño propuesto por Sunar, Martin y Stinson (2007) para ser implementado en una FPGA. Está formado por varios osciladores de anillos de igual longitud con un número impar de elementos y una XOR en la salida. Un esquema del diseño se muestra en la Figura 3.10. Las frecuencias de los osciladores de la misma longitud, no son iguales debido a los lugares en donde se implementan los inversores en la FPGA. Este diseño cumple con los test DIEHARD y NIST con un post procesamiento adicional, pero existen implementaciones similares que no lo necesitan.

3.3.6. Diseño propuesto por Intel

El diseño propuesto por Jun y Kocher (1999) se muestra en la Figura 3.11. La fuente de ruido de este RNG son dos resistencias. Este ruido es amplificado y se

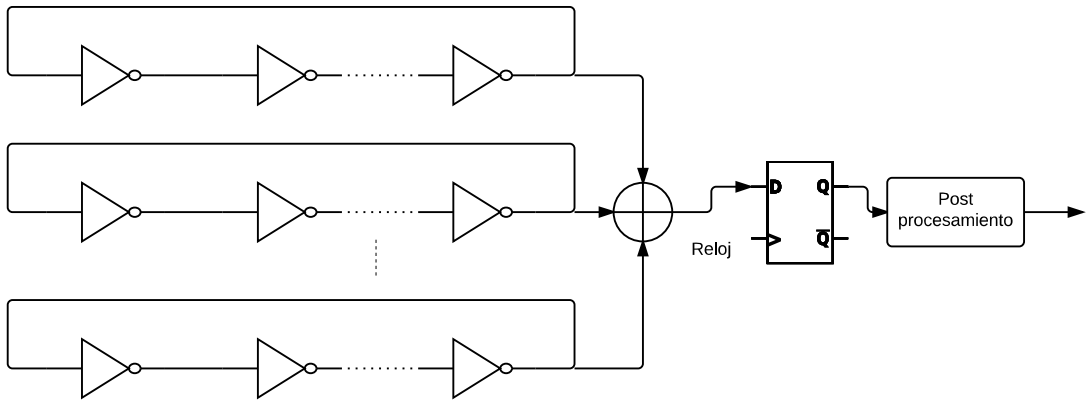


FIGURA 3.10: Diseño propuesto por Sunar.

utiliza como entrada a un oscilador controlado por tensión (VCO). La frecuencia de oscilación nominal del VCO se ve afectada por la tensión de ruido aleatorio, provocando una variación aleatoria en la frecuencia de oscilación de salida. Esta señal se utiliza para muestrear otro oscilador de frecuencia mucho mayor produciendo una secuencia de dígitos binarios aleatorios.

Para corregir posibles desviaciones en la secuencia aleatoria resultante del muestreo, la señal es post procesada por un corrector del tipo Von Neumann. También es post procesada a través de una función del tipo SHA-1. La entrada del circuito es diferencial con el fin de hacer más robusto el diseño contra variaciones externas y contra variaciones en la fuente de alimentación.

El diseño propuesto, como se indica en la publicación, cumple los requerimientos de las pruebas estadísticas de NIST FIPS 140-1.

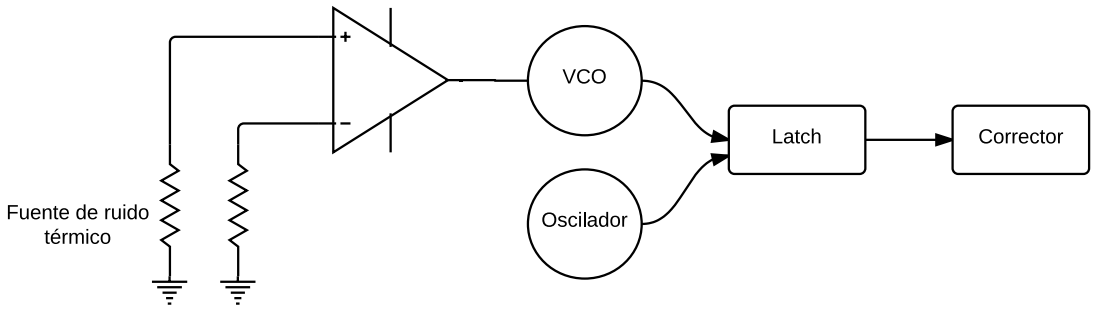


FIGURA 3.11: Diseño propuesto por Intel.

3.3.7. Análisis comparativo

Los diseños presentados en la sección anterior son generadores de números aleatorios para aplicarse en criptografía. Una de las características principales que los diferencia entre unos y otros son los componentes del diseño. Por un lado están aquellos que cuentan con un diseño analógico y obtienen la aleatoriedad desde algún fenómeno físico, en general este tipo cuenta además con algún bloque digital. Por otro, aquellos que se encuentran totalmente diseñados por elementos digitales, destinados a un circuito integrado ó a un FPGA. Y por último aquellos diseños cuyo propósito es únicamente ser implementado en un FPGA.

En el caso del generador propuesto por Bagini y Bucci (1999), su diseño tiene la ventaja de poseer un modelo analítico que describe el comportamiento del sistema. La aleatoriedad es obtenida del ruido shot de una juntura polarizada en directa. Este ruido es controlado directamente por la corriente de polarización, pero su amplitud es demasiado baja y debe ser fuertemente amplificada. Dado que el amplificador debe tener una alta ganancia se debe evitar que el ruido determinístico generado por fuentes externas apantalle el ruido proveniente de la juntura. En esta implementación se garantiza que los bits de salida son aleatorios. Por otro lado, tiene la desventaja que los requerimientos del comparador son grandes debido a que su respuesta debe ser lo suficientemente rápida para que la salida del contador binario sea muestreada en un estado estable. En este caso se propone un modelo con simulaciones en Simulink, sin resultados reales y no se presentan test estadísticos estandarizados para observar confiabilidad de la aleatoriedad.

En el caso del diseño de Intel (Jun y Kocher, 1999), tiene componentes digitales y analógicos, posee la ventaja que todas las estructuras que se utilizan son simples y conocidas. La aleatoriedad generada por este método tiene un respaldo matemático. Fue verificado y cumple los test estadísticos del NIST y los Diehard. A pesar de tener implementado un corrector para reducir la polarización de los bits de salida, sin este corrector el diseño continúa cumpliendo con los requerimientos de aleatoriedad.

La implementación de 2003 es totalmente digital, lo que permite implementaciones más simples y, si se requiere, cuenta con la posibilidad de implementarlo en un FPGA. Cumple con los requerimientos de los test estadísticos y permite reducir el consumo cuando no se necesitan números aleatorios, sin afectar el correcto funcionamiento del generador.

Otro diseño puramente digital es el de Golić (2006). Cuenta con una justificación matemática detallada y fue testeado experimentalmente en un FPGA. Una característica a resaltar es que el diseño es independiente de la tecnología en la que se realice.

Por último el diseño de Fischer y Drutarovskỳ (2003) utiliza bloques que permiten implementar el generador completamente en un FPGA, inclusive un PLL analógico. Como la medida del jitter no es una medida directa, el diseño asume teorías sobre la aleatoriedad del jitter, que luego constata con medidas y las confirma comprobando el valor medio de la salida del generador. El diseño cuenta con una verificación por medio de los test estadísticos estándares del NIST bajo 1 Gb de muestras y se comprobó que cumple con los requerimientos de aleatoriedad. Debido a que se encuentra totalmente en un FPGA, tiene la posibilidad de poder incluirse en un sistema tipo SoPC (*System on Programmable Chip*). Por otro lado tiene la desventaja que el diseño depende exclusivamente de la plataforma programable en la que se implemente, debido a que la aleatoriedad está asociada al jitter del generador. Otra desventaja es que el diseño no puede ser simulado y necesita ser testeado en hardware real.

3.4. Elección del tipo de generador

La tarea de generar números aleatorios es muy importante cuando se implementan sistemas criptográficos. Al diseñar un protocolo de seguridad, existen muchas aplicaciones que requieren números aleatorios, como por ejemplo, los algoritmos usados para llaves de autenticación (RSA, DSA, ECDSA), y para llaves de encriptación (3DES, AES, RSA). Estos tipos de algoritmos necesitan un generador

con una buena calidad, que evite que se puedan inferir los números generados. Por otro lado, las aplicaciones de identificación, y vectores de inicialización, necesitan números aleatorios con menores restricciones, y una salida predecible pero no repetitiva puede ser aceptable.

Otra aplicación en donde se utilizan números con características aleatorias es crear llaves privadas en sistemas de comunicación modernos. Las llaves privadas son números aleatorios únicos para el usuario. En este tipo de sistemas, para establecer una conexión segura, se crea una llave pública que es compartida con los demás usuarios. La llave pública se crea a partir de un número primo y se lo eleva a la potencia del valor de la llave privada del propio usuario, creando un número de gran magnitud que asegura que la llave privada original no pueda ser obtenida con facilidad. La aleatoriedad de los números de la llave privada determina que tan segura es la comunicación contra ataques.

En general, para todas las aplicaciones de criptografía mencionadas, lo más importante es que el atacante, incluyendo aquel que conoce el diseño del generador, sea incapaz de hacer predicciones sobre la salida del generador. Para que esto sea posible, la entropía de una secuencia de salida de n bits idealmente debería ser igual a n.

La entropía de un mensaje se define como:

$$H = -K \sum_{i=1}^{n} p_i \log p_i, \tag{3.5}$$

donde p_i es la probabilidad de que la salida del generador sea un número determinado, n la cantidad de estados posibles y K una constante opcional para proporcionar unidades (1/log 2 bit).

Si el generador produce un resultado binario de $k - bit$, p_i es la probabilidad que la salida sea $0 \leq i \leq 2^k$. Si el generador es ideal $p_i = 2^{-k}$ (cada posible salida tiene la misma probabilidad de ocurrir), con una entropía de k bits.

Una evaluación de seguridad para un generador es cuantificar la entropía generada por bit. La entropía es una propiedad de las variables aleatorias y no de realizaciones observadas. En particular, la entropía no puede ser garantizada por medio de una colección de test de "caja negra" estadísticos. Para estudiar la entropía se tiene que estudiar la distribución de los números aleatorios o la distribución de la variable aleatoria en cuestión.

Una fuente que genera números aleatorios se puede expresar como un sistema que genera secuencias X muestreando y cuantizando un valor S no determinístico. Este valor no determinístico se puede cuantizar de dos maneras posibles: un modo signo o un modo módulo 2,

$$x(i) = \text{sign } (s(i)), \tag{3.6}$$

$$x(i) = s(i) \bmod 2, \tag{3.7}$$

representados respectivamente, donde $x(i)$ es el valor actual de la secuencia y $s(i)$ el valor de la señal no determinística. La cuantización en aritmética modular presenta la ventaja de hacer menos significativos los efectos determinísticos y se lleva a cabo por un flip flop D.

La calidad de la secuencia de salida X depende de la calidad de la fuente S y del procedimiento de cuantización. Un modelo simple de la fuente S se expresa como:

$$S = aR + m + D. \tag{3.8}$$

En la Ecuación (3.8), R es un proceso aleatorio normalizado, y debido a las limitaciones físicas de la fuente, es de ancho de banda limitado. La amplitud intrínseca de la fuente de ruido y una posible amplificación se representa por la letra a, m es un nivel de offset y D representa cualquier proceso determinístico sobre el proceso aleatorio R (ruido externo, ruido de otros componentes, o un atacante). Sin importar la amplitud de D y m, sus efectos se vuelven despreciables si aR es lo suficientemente grande.

Como se presentó en los diseños de generadores, una técnica posible para generar números aleatorios que aplica a las Ecuaciones (3.6) y (3.8), es amplificar directamente el ruido térmico producido por una resistencia, como se muestra en la Figura 3.12. Luego, un comparador sincrónico realiza el muestreo y la cuantización de la fuente de ruido, y finalmente un filtro pasa bajos compensa el offset del amplificador y del comparador. Esta clase de fuente puede tener un alto rendimiento en lo que respecta a los bits generados, siendo una limitación el ancho de banda del amplificador. Esta técnica requiere de una implementación precisa debido al uso de la cuantización del tipo signo; si el offset o alguna señal de interferencia externa son de una amplitud notable, la aleatoriedad de la salida se puede ver altamente afectada.

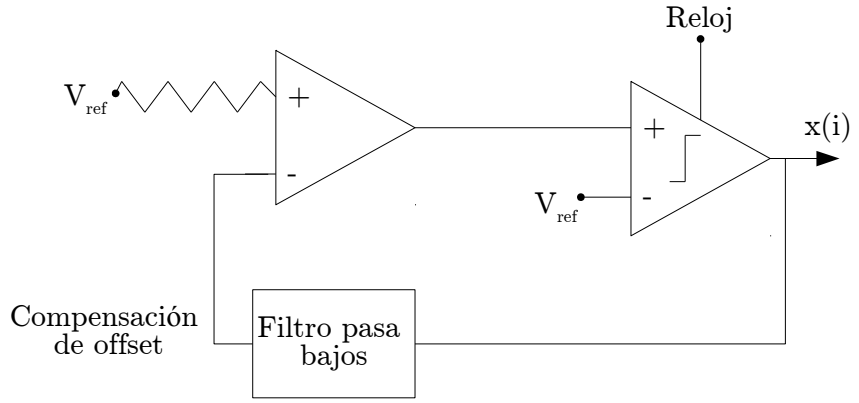


FIGURA 3.12: Método de amplificación directa.

Otra de las técnicas posibles, como también se mostró en los diseños de generadores, es obtener números a partir de los cambios aleatorios producidos en la frecuencia de una señal proveniente de un oscilador de baja frecuencia. Unas de las posibilidades es que un oscilador de baja frecuencia tome muestras de un oscilador con frecuencia natural mucho mayor. En general, este tipo de fuentes provee una implementación más simple y más robusta en relación al caso anterior. La cuantización, como se indica en la Ecuación (3.7), es en aritmética modular y se implementa por medio de un flip flop D (FFD), como se muestra en la Figura 3.13. El ancho y la simetría de las bandas de cuantización dependen de la relación entre las frecuencias de oscilación, y del ciclo de trabajo del oscilador rápido. Una vez

que la frecuencia del oscilador rápido se maximiza, haciendo las bandas de cuantización lo más chicas posibles, el período de muestreo del oscilador lento debe hacerse lo más ancho posible de modo de acumular suficiente ruido de fase entre dos muestras subsecuentes. Cuanto más ruido posean los osciladores, mayor puede ser la frecuencia de muestreo.

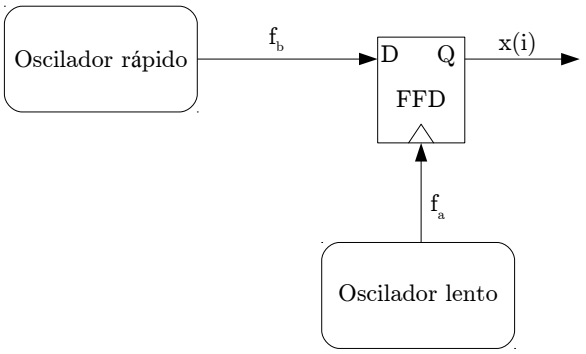


FIGURA 3.13: Método con dos osciladores de distintas frecuencias.

Al utilizar el método del muestreo de una señal rápida con otra lenta, se debe evitar que el siguiente valor se pueda determinar, aún conociendo las frecuencias del dato y del reloj que controlan el flip flop D. Este problema se ilustra en la Figura 3.14, donde $f(x)$ es la función de densidad de probabilidad del jitter, CLK la forma de onda del reloj y D el dato. Para obtener una salida aleatoria, las probabilidades que la salida sea 1 o 0 deben ser iguales. A partir de la función de densidad de probabilidad, se conoce que el área debajo de la curva es igual a 1, lo que corresponde a un 100 % del total de la variación del reloj lento en el flanco. La probabilidad de que el dato D sea igual a 1, en el flanco de la señal CLK, es:

$$P(D = 1) = P(a < Z < b) + P(c < Z < d), \tag{3.9}$$

calculada a partir de la Figura 3.14. Dicha ecuación corresponde a las regiones sombreadas $a - b$ y $c - d$, sobre el área total. De la explicación puede determinarse que la desviación estándar del jitter debe ser suficiente para que la suma combinada de las regiones sombreadas en la función de densidad de probabilidad sea igual a 0.5 o al 50 % de la función de densidad de probabilidad. De esta manera, el valor de D será 1 o 0 con la misma probabilidad (Fairfield, Mortenson y Coulthart, 1985).

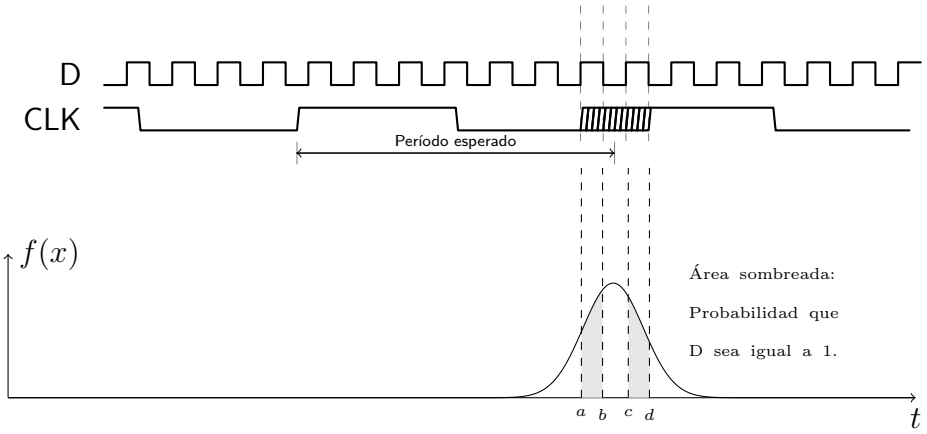


FIGURA 3.14: Formas de ondas de los osciladores y función de densidad de probabilidad del jitter.

Luego de conocer las implementaciones de generadores más comunes y en las que se basan varios diseños, y expuestas las características entre un tipo de método de generación y otro, se considera implementar la arquitectura provista por dos osciladores funcionando a distintas frecuencias. El jitter, en este caso se forma a partir del jitter intrínseco de un VCO y el que se agrega debido a ruido térmico amplificado proveniente de resistencias.

El ruido térmico tiene características aleatorias y posee una distribución Gaussiana, cuya densidad de potencia depende del valor de la resistencia (R) y de la temperatura (T) y está representado por:

$$V_{noise}(\text{RMS}) = V_n = (4kTRB)^{\frac{1}{2}}, \tag{3.10}$$

donde k es la constante de Boltzmann, T es la temperatura absoluta en grados Kelvin (K = C + 273,16), y B es el ancho de banda en Hertz.

La amplitud del voltaje de ruido térmico en cualquier instante de tiempo es impredecible, pero obedece a una distribución de amplitud Gaussiana como se muestra en la Figura 3.15, donde $p(V)dV$ es la probabilidad que la tensión instantánea se encuentre entre V y $V + dV$, y V_n es la tensión de ruido RMS, dada en la Ecuación (3.10).

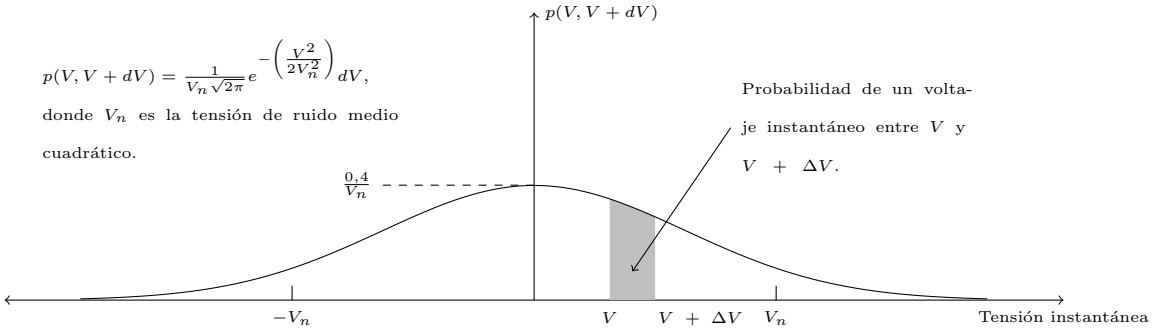


FIGURA 3.15: Distribución de amplitud Gaussiana del ruido térmico.

Una fuente de ruido blanco, con densidad espectral de ruido uniforme, tendrá una distribución del tipo Gaussiana en el valor de amplitud de salida. El valor de la tensión de ruido tendrá en cualquier momento la misma probabilidad de estar por encima o por debajo del valor medio.

3.5. Generador de números aleatorios propuesto

Los generadores de números aleatorios basados en algoritmos matemáticos y circuitos digitales son sencillos de implementar, y suelen presentar especificaciones mínimas de área y consumo. Sin embargo, ciertas aplicaciones (como las criptográficas o los sistemas de seguridad), tienen requerimientos de aleatoriedad que estos generadores no pueden cumplir. Uno de los requerimientos más importantes es que los números aleatorios no puedan predecirse, aún teniendo conocimiento del diseño del generador y de secuencias pasadas de números generados. Para esto es necesario contar con una fuente de ruido no determinístico, que se denomina "semilla".

El generador propuesto en este trabajo está basado en el diseño presentado por Intel (Jun y Kocher, 1999).

3.5.1. Diseño

El diseño del generador de números aleatorios que se propone tiene como aplicación objetivo su utilización en tarjetas inteligentes. Para este fin, una velocidad de generación de 800 Kbps se considera adecuada. El generador se implementará en una tecnología de 180 nm de Tower Jazz por estar disponible desde la Universidad Nacional de Sur.

Tomando ventaja de los fenómenos físicos impredecibles, el diseño obtiene la aleatoriedad del ruido térmico proveniente de resistencias. Este ruido se amplifica por un circuito diseñado de manera tal que no aporte ruido significativo. La señal aleatoria resultante se utiliza para modular la frecuencia de un oscilador controlado por tensión (VCO), cuya frecuencia nominal es de 100 KHz. La topología seleccionada para el VCO cuenta con estructuras simples y conocidas y resulta suficiente para cumplir los requerimientos y asegurar su respuesta lineal en el rango de variación de tensión de entrada.

La salida del VCO se utiliza como reloj de entrada a ocho flip flops D con el fin de muestrear el dato proveniente de osciladores de anillos funcionando a una frecuencia de oscilación natural de 600 MHz, mucho mayor que la frecuencia media del VCO. Por la relación entre las frecuencias de los osciladores y la variación provocada por el ruido térmico, los bits de salida son impredecibles. Esta técnica para generar aleatoriedad es muy utilizada y tiene la ventaja con respecto a otras de limitar o hacer despreciables los efectos determinísticos. Además, el generador cuenta con una interfaz AMBA, muy utilizada para la comunicación entre componentes dentro de un circuito integrado, facilitando el intercambio de información con un microprocesador u otros dispositivos.

La Figura 3.16 presenta el diagrama en bloques del generador propuesto.

El generador cuenta con las siguientes características:

- La semilla de aleatoriedad es la diferencia entre el ruido térmico producido por dos resistencias. Las resistencias son de polisilicio y su valor es tal que

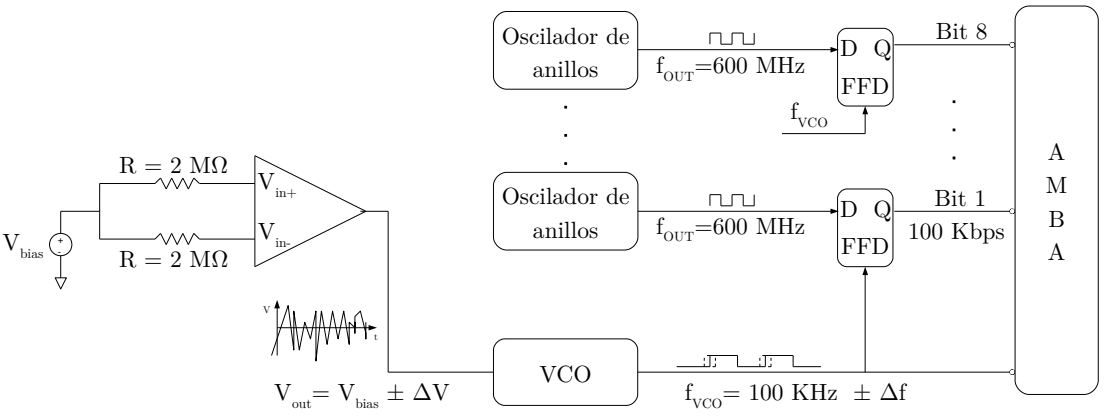


FIGURA 3.16: Diagrama en bloques de la arquitectura seleccionada.

permite generar un nivel suficiente de ruido (que también depende de la temperatura) sin afectar significativamente el área total del circuito.

- El amplificador tiene entrada diferencial y salida simple. El nivel medio de la salida permite polarizar el VCO de manera que presente una frecuencia de oscilación nominal. Los límites máximo y mínimo del ruido de salida tienen que ser tales que permitan un rango mínimo de variación de la frecuencia del VCO, pero sin afectar su polarización.

- La frecuencia de la señal de salida del VCO depende de la señal aleatoria de entrada, por lo tanto, su naturaleza es también aleatoria. Esto significa que es imposible determinar el momento en que se producirá el siguiente flanco de reloj que sincroniza los flip-flops.

- El VCO presenta una frecuencia de oscilación nominal de 100 KHz. Esta frecuencia está determinada por la aplicación, que requiere un número aleatorio de 32 bits en intervalos de aproximadamente 40 μs.

- A frecuencia nominal, un oscilador de anillo produce un bit aleatorio cada 10 μs. La cantidad de osciladores de anillos necesaria depende del ancho de palabra del número aleatorio requerido por la aplicación. En este caso se implementó un arreglo de 8 osciladores de anillo, que permiten obtener 32 bits aleatorios cada 40 μs.

- El ciclo de trabajo de los osciladores de anillo tiene que ser estrictamente del 50 %, de manera de obtener la misma distribución de probabilidad de ceros y unos. La frecuencia natural de los osciladores de anillo debe ser mucho mayor que la del VCO, de manera que su período sea estrictamente menor al jitter de la señal de muestreo.

- La salida de los osciladores de anillo se conectan a la entrada de dato de un flop-flop tipo D, y la salida del VCO se conecta a la entrada del reloj. El flip-flop no tiene restricciones particulares y no formó parte del diseño.

- La aleatoriedad de los números generados se verifica mediante métodos estándar, como el conjunto de pruebas estadísticas del NIST.

- El generador cuenta con una interfaz AMBA APB para permitir la comunicación con otros circuitos como por ejemplo, un microprocesador.

A continuación se detallan cada uno de los componentes del diseño.

3.5.1.1. Amplificador del ruido térmico

La función principal del amplificador es aumentar el nivel de la señal de ruido térmico, sin aportar componentes significativas de ruido determinístico. Para esto se diseñó un amplificador diferencial con entradas tipo N, como puede verse en la Figura 3.17. Las corrientes de polarización Ibias_a e Ibias_b se generan mediante una fuente de corriente estable en temperatura, e insensible a variaciones de proceso, basada en el diseño presentado por Shinde (2014). La resistencia de polarización de la fuente de corriente se implementó en forma externa al chip, permitiendo un ajuste fino de la polarización del amplificador e, indirectamente, de su ganancia.

Dado que el generador debe funcionar en el rango de temperaturas de 0°C a 100°C, y que el ruido térmico de las resistencias depende directamente de la temperatura, el amplificador debe presentar una ganancia variable con la temperatura. Con el objetivo de mantener el diseño simple y ocupando el menor área posible, se decidió

implementar el amplificador como una cascada de 5 etapas diferenciales, y dejar disponible desde el exterior del chip la salida de cada etapa. De esta manera, a bajas temperaturas, cuando el nivel de ruido térmico es muy bajo, se utiliza la salida de la 5^{ta} etapa, que produce la máxima amplificación. Sin embargo, si la temperatura es alta y la salida de la última etapa produce una intensidad de señal tal que podría despolarizar el VCO, entonces se utiliza la salida de una etapa previa.

El circuito esquemático de la Figura 3.17 muestra la primera etapa amplificadora. R3 y R4 son las fuentes de ruido térmico y las salidas Vout_a y Vout_b se conectan a las entradas Vin_a y Vin_b de la etapa siguiente.

Además, se implementó una realimentación de corriente continua, para mantener controlado el nivel medio de la señal de salida, en el valor que genera la frecuencia de oscilación nominal en el VCO.

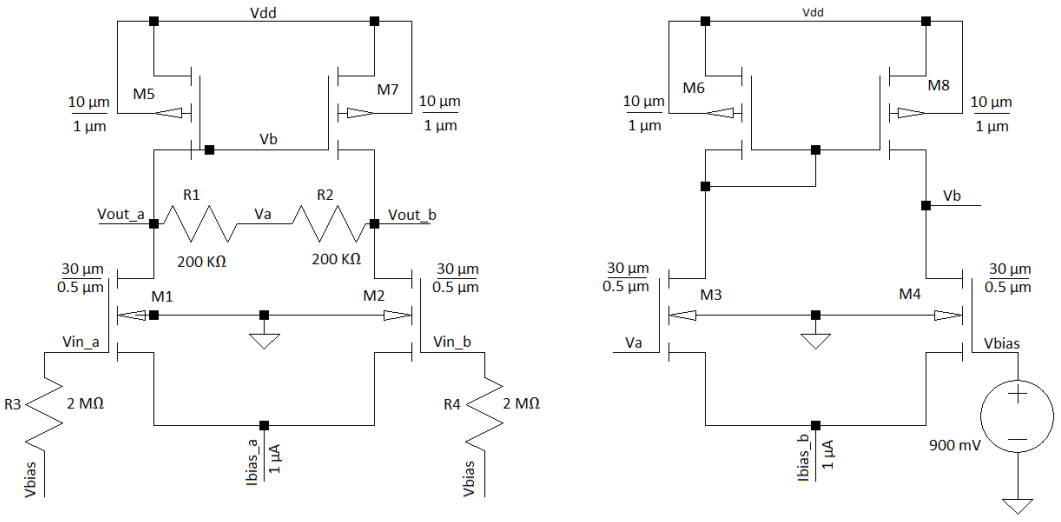


FIGURA 3.17: Una etapa del amplificador de ruido, con la realimentación en corriente continua.

3.5.1.2. Oscilador controlado por tensión

Un oscilador controlado por tensión está conformado por un oscilador y un circuito que controla la frecuencia de la oscilación a través de una tensión. Los osciladores pueden generar una salida sinusoidal, cuadrada o triangular. Los que tienen salida

sinusoidal se realizan utilizando un selector de frecuencias o un circuito de sintonización en una configuración de realimentación, mientras que los osciladores con salida cuadrada se realizan por medio de un circuito con realimentación no lineal. Estos tipos de osciladores no lineales pueden ser de anillos o de relajación.

En el diseño de un VCO se deben tener en cuenta distintos parámetros de desempeño, los cuales se mencionan a continuación:

- Frecuencia central: es la frecuencia nominal de trabajo del VCO.
- Rango de sintonización: este rango se encuentra determinado por la variación de la frecuencia central del VCO debido a variaciones en el proceso y en la temperatura y al rango de frecuencia necesario para cada aplicación.
- Linealidad en el rango de sintonización: la ganancia del VCO presenta no linealidades, por lo cual es necesario disminuir las variaciones de la ganancia a lo largo del rango de sintonización. Las características típicas de los osciladores en general cuentan con una alta ganancia alrededor del valor medio del rango de la variación, y baja en los extremos.
- Amplitud de salida: es deseable conseguir una amplitud grande a la salida con el fin de hacer la oscilación menos sensible al ruido, a expensas de la disipación de potencia, alimentación, y el rango de sintonización.
- Disipación de potencia: como ocurre con otros circuitos analógicos, los osciladores poseen relación de compromiso entre disipación de potencia, velocidad, y ruido.
- Alimentación y rechazo al modo común: los osciladores son sensibles al ruido y en especial a los cambios que se pueden producir en la fuente de alimentación. Dependiendo de la aplicación, para disminuir estos inconvenientes, se suelen utilizar osciladores diferenciales.
- Pureza de la señal de salida: aún con una tensión de control constante, la forma de onda de la señal de salida de un VCO no es perfectamente periódica.

El ruido electrónico propio de los dispositivos y la fuente de alimentación provocan variaciones en la frecuencia de salida del oscilador.

Luego de detallar las características de desempeño, se consideran las ecuaciones involucradas y la topología elegida.

En un oscilador controlado por tensión la frecuencia de salida ω_{VCO} se relaciona con la tensión de entrada

$$\omega_{VCO} = K_{VCO}\ V_{inVCO} + \omega_0, \tag{3.11}$$

donde ω_0 expresada en rad/s es una constante.

La ganancia K_{VCO} de un oscilador controlado por tensión es la pendiente de la curva dada en la Figura 3.18, la cual puede ser representada como:

$$K_{VCO} = 2\pi\ \frac{f_{max} - f_{min}}{V_{max} - V_{min}}\ \left(\frac{\text{rad}}{\text{s V}}\right). \tag{3.12}$$

En general, está relacionada con el tipo de implementación de VCO que se realice.

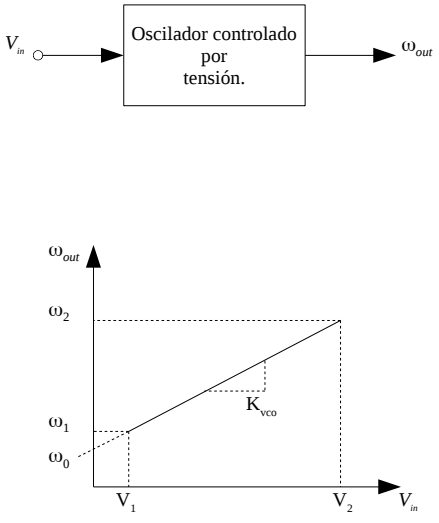


FIGURA 3.18: Ganancia del VCO.

Una de las técnicas más conocidas para realizar un VCO es a partir de un oscilador de anillos con la adición de un control de tensión (Baker, 2011). Controlando el retardo de los inversores, se controla la frecuencia nominal del VCO.

El diseño propuesto se basa en el esquema de la Figura 3.19. Está formado por un inversor (M2 y M3), y dos transistores que funcionan como fuente de corriente (M1 y M4). Estas fuentes de corriente limitan la corriente disponible al inversor a través de las tensiones aplicadas a sus *gates* de entrada. Estas corrientes son iguales y se espejan en cada una de las etapas que conforman el oscilador de anillos. Para determinar las ecuaciones de diseño, se debe hallar la capacidad total en el *drain* de M2 y M3, correspondiente a un inversor. Es decir, la capacidad del *drain* propia (capacidad entre *drain-source*, efecto de la capacidad de entrada en la salida, capacidad *gate-source* y *gate-drain*) y capacidad de la próxima etapa inversora:

$$C_{tot} = C_{out} + C_{in} = C_{ox}(W_p L_p + W_n L_n) + \frac{3}{2} C_{ox}(W_p L_p + W_n L_n). \tag{3.13}$$

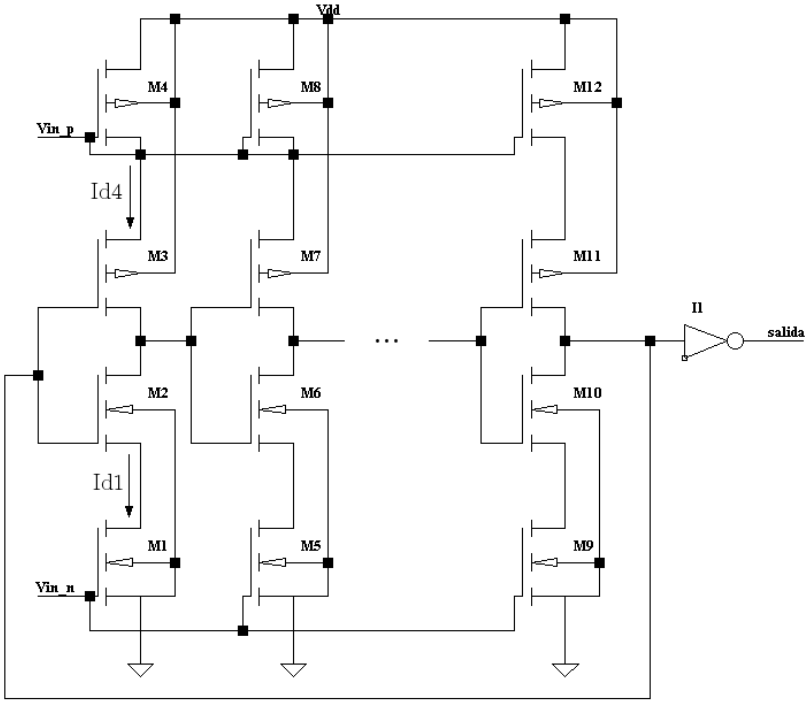


Figura 3.19: Esquemático del VCO.

La corriente que circula por el transistor PMOS del inversor, carga la capacidad total C_{tot} desde cero hasta la tensión media del inversor V_M, donde la tensión de entrada es igual a la de salida. El tiempo de carga de esta capacidad es:

$$t_1 = C_{tot}\frac{V_M}{I_{D4}}. \tag{3.14}$$

El tiempo de descarga desde V_{dd} hasta V_M está dado por:

$$t_2 = C_{tot}\frac{V_{dd} - V_M}{I_{D1}}. \tag{3.15}$$

Como la corriente que circula por ambos transistores es la misma, $I_{D1} = I_{D4} = I_D$, el tiempo total que tarda el inversor en cargar y descargar el nodo de salida es:

$$t_1 + t_2 = \frac{C_{tot}V_{dd}}{I_D}. \tag{3.16}$$

Entonces, el retardo promedio de un inversor simple es:

$$T_D = \frac{t_1 + t_2}{2}. \tag{3.17}$$

Si las fuentes de corriente se colocan en el mismo punto de operación, el período y la frecuencia de oscilación del VCO están dados por:

$$T_{osc} = 2\ N\ T_D. \tag{3.18}$$

$$f_{osc} = \frac{1}{N(t_1 + t_2)} = \frac{I_D}{NC_{tot}V_{dd}}, \tag{3.19}$$

respectivamente. En lo que respecta a las tensiones aplicadas a los *gates* de las fuentes de corriente, cuando la tensión entre el *gate* y el *source* del transistor $M1$ está por debajo de su tensión de umbral, no circula corriente por el inversor y el VCO deja de oscilar. Además, la tensión del *gate* de $M4$ no debe alcanzar $V_{dd} - V_{th}$ y ambos transistores deben permanecer en saturación.

La potencia disipada por el VCO está dada por $P_{avg} = V_{dd}\ I_D$. Por tal motivo,

si se requiere bajo consumo, esta corriente debe ser disminuida, significando una frecuencia de oscilación menor.

En la Figura 3.20 se muestra una etapa de la topología del VCO elegida con los parámetros de diseño. Para lograr la frecuencia de oscilación deseada, esta etapa se replica 11 veces. El control de la corriente por el oscilador está dado por el transistor M1 o M4 según sea la relación de aspecto entre ellos, es decir, según la relación entre el ancho del transistor y el largo del canal donde circulan las portadores de carga. Si el W (ancho del transistor) del M4 es mucho menor que el del M1 para un mismo L (largo del canal), la corriente por las ramas inversoras la controla el transistor M4 a través de su tensión de *gate*. Si el W del M1 es menor que el del M4 la corriente de las ramas inversoras la controla el transistor NMOS M1. El objetivo es encontrar la relación de aspecto entre ellos, de tal forma que variando una de las dos tensiones de *gate* la corriente sea controlada por el transistor NMOS M1 o por el transistor PMOS M4.

La tensión del *gate* del transistor NMOS M1 de la fuente de corriente, Vbias_N, se fija en 500 mV. También se fija la relación de tamaños para que la corriente por los inversores se controle por la tensión de *gate* $Vbias\_P$ aplicada al transistor PMOS M4. Debido al ruido generado por las resistencias y a la ganancia del amplificador, esta tensión tiene un valor nominal de 0,9 V y se considera una variación de $\pm$50 mV lo que produce que la frecuencia de salida del VCO varíe 14,2 KHz en torno a su valor nominal.

En la Tabla 3.2 se resumen los parámetros del diseño del oscilador controlado por tensión.

Por otro lado, los transistores que conforman el oscilador de anillos M2, M3, y los de las demás etapas, se polarizan en un punto de operación en inversión débil, haciéndolos más anchos (W), que largos (L), de forma tal de tener una pequeña tensión entre el *drain* y el *source* en los transistores inversores, y asegurar que los transistores fuentes de corriente M1, M4 y los que copian esta corriente se polaricen en saturación.

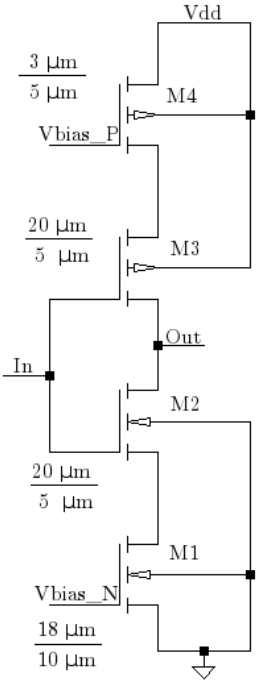


FIGURA 3.20: Una etapa del oscilador controlado por tensión.

TABLA 3.2: Variables del diseño

Componente (Figura 3.20)	Valor
Inversores $\left(\frac{W}{L}\right)_{M2-M3}$	$\frac{20\ \mu\text{m}}{5\ \mu\text{m}}$
Fuente de corriente PMOS $(\frac{W}{L})_{M4}$	$\frac{3\ \mu\text{m}}{5\ \mu\text{m}}$
Fuente de corriente NMOS $(\frac{W}{L})_{M1}$	$\frac{18\ \mu\text{m}}{10\ \mu\text{m}}$
$Vbias\_P$	900 mV
$Vbias\_N$	500 mV

Cuando el VCO oscila a su frecuencia nominal, el valor de corriente que circula por las ramas inversoras es $I_d = 3{,}76\ \mu$A. Para esta topología y conociendo el valor de corriente para una frecuencia de oscilación dada, se puede hallar el número de etapas necesarias para una frecuencia de oscilación particular, por medio de la Ecuación (3.19).

En la Tabla 3.3 se muestran los resultados de la frecuencia de oscilación del VCO para una tensión de entrada de $V_{dd}/2$ y de $V_{dd}/2 \pm 0{,}05$ V y la corriente en cada una de las ramas inversoras.

TABLA 3.3: Frecuencia nominal y rango de variación del VCO.

$Vbias\_P$ [V]	F_{osc} [KHz]	I_D [μA]
0,9	105,2	4,44
0,85	112,5	3,76
0,95	98,10	3,12

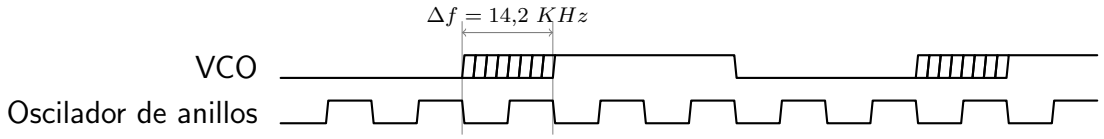


FIGURA 3.21: Señal del VCO muestreando la señal rápida en los flancos positivos.

Con una variación en la tensión de entrada del VCO de ±50 mV en torno al valor de tensión nominal en el *gate* del transistor PMOS, se obtiene una variación de frecuencia de $\Delta f = 14{,}2$ KHz.

Conociendo la variación de frecuencias en la salida del VCO se diseña el oscilador de anillos. El período del mismo puede ser, por un lado igual al delta de frecuencias y tener un ciclo de trabajo de exactamente el 50 % como se indica en la Figura 3.21, o tener un período mucho menor que esta variación. En este caso varios períodos del oscilador rápido entran en el delta de frecuencias del VCO. Cualquiera de los dos diseños aseguran que a la salida del flip flop se obtendrá un valor indeterminado. En el diseño se opta por la segunda opción.

3.5.1.3. Oscilador de anillos

El oscilador de anillos está constituido con un número impar de inversores en un lazo de realimentación, como se muestra en la Figura 3.22. Cada medio período, la señal se propaga a través del lazo con una inversión. Asumiendo que cada inversor tiene un retardo de τ_{inv} y que hay n inversores en el lazo, se tiene que la mitad del período de oscilación está dada por:

$$\frac{T}{2} = n\ \tau_{inv}, \tag{3.20}$$

y la frecuencia de oscilación es:

$$f_{osc} = \frac{1}{T} = \frac{1}{2\ n\ \tau_{inv}}. \tag{3.21}$$

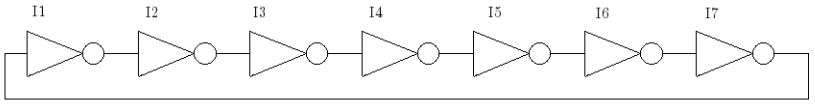


FIGURA 3.22: Oscilador de anillos.

El retardo de propagación de los inversores depende de la capacidad de carga en el nodo de salida del mismo. Esta capacidad de carga está compuesta por la capacidad de salida propia del inversor, y la capacidad de entrada de la siguiente etapa.

En la Figura 3.23 se representa la señal de salida del VCO cuando la tensión de entrada es $V_{dd} = 0{,}9$ V y cuando varía en $\pm 0{,}05$ V ($f_{0,9},\ f_{0,95},\ f_{0,85}$). Asumiendo que ésta es la variación máxima, se calcula el período máximo permitido para el oscilador de anillos:

$$T_{osc} = \frac{T_{0,95}}{2} - \frac{T_{0,85}}{2} = 652{,}4 \text{ ns}. \tag{3.22}$$

Dicho período debe ser exactamente igual a esta variación o un múltiplo entero, caso contrario se obtendrá un valor determinístico no deseado a la salida del flip flop. La frecuencia de oscilación mínima es:

$$F_{osc} = \frac{1}{T_{osc}} = 1{,}532 \text{ MHz}, \tag{3.23}$$

donde $T_{0,95} = 10{,}2\ \mu$s, y $T_{0,85} = 8{,}8\ \mu$s.

Con el fin de obtener la relación de tamaños entre los inversores que conforman el oscilador, utilizando los modelos de los transistores de la tecnología, se obtuvieron sus parámetros a partir de la simulación de sus características de salida en saturación. Con la aproximación, y planteando que las corrientes por ambos transistores son las mismas, que la tensión de entrada es $Vdd/2$ y asumiendo que tienen el largo mínimo, la relación entre el ancho del NMOS W_n y el ancho del PMOS W_p

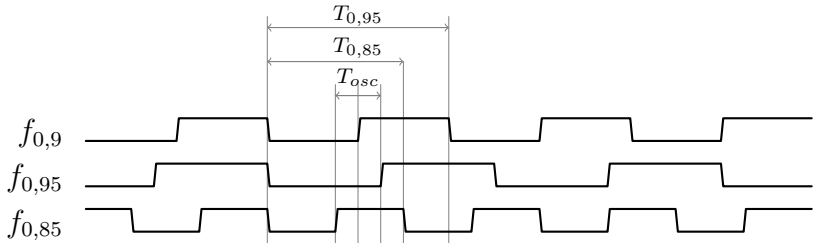


FIGURA 3.23: Máxima variación de frecuencia asumida a la salida del VCO.

es $W_p = 5{,}27\ W_n$. Utilizando el ancho mínimo para el transistor NMOS, los tamaños de los transistores que conforman el oscilador de anillos se muestran en la Tabla 3.4. Con siete etapas como la que se muestra en la Figura 3.24 conectadas

TABLA 3.4: Tamaños de los inversores del oscilador de anillos.

Oscilador de anillos	$\frac{W}{L}[\mu\text{m}]$
PMOS	$\frac{2{,}32}{0{,}36}$
NMOS	$\frac{0{,}44}{0{,}36}$

en cascada, la frecuencia de oscilación resultante obtenida a partir de la simulación transitoria es de 693,5 MHz.

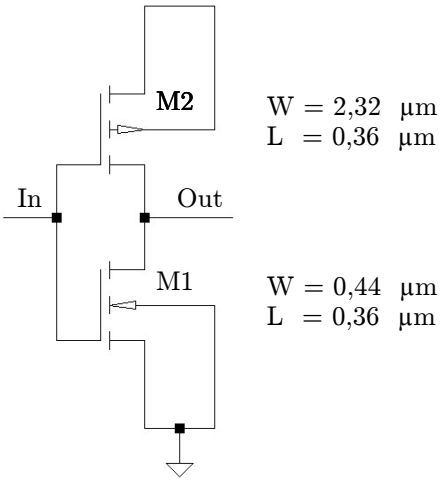


FIGURA 3.24: Una etapa inversora del oscilador de anillos.

Los tiempos de trepada, tp_{lh}, y de caída, tp_{hl}, del inversor se obtuvieron a partir de la simulación utilizando los parámetros de la tecnología:

$$tp_{lh} = 104{,}7 \text{ ps}, \tag{3.24}$$

$$tp_{hl} = 127{,}9 \text{ ps.} \tag{3.25}$$

A partir de éstos tiempos y la cantidad de etapas (N) que en este caso es de 7 inversores, se puede calcular la frecuencia de oscilación resultante con la Ecuación (3.21). Como se puede comprobar, este valor es similar al obtenido por simulación. La corriente que circula por los inversores en el punto de operación es de 32 μA.

$$f_{ring} = \frac{1}{2\ N\ \frac{tp_{lh}+tp_{hl}}{2}} = 614,18 \text{ MHz.} \tag{3.26}$$

Con esta frecuencia de oscilación, en 1 período de la variación producida por el VCO entran 452 períodos del oscilador de anillos, lo cual es lo suficientemente grande para producir un muestreo aleatorio por medio del flip flop. El período del oscilador rápido tiene que ser igual al período de oscilación máximo o un múltiplo del mismo, es decir, T_{osc}/n con n mayor o igual que uno.

$$T_{ring} = \frac{T_{osc}}{n} = 1{,}44 \text{ ns,} \tag{3.27}$$

donde n es igual a $452,44$ y T_{osc} el período de la Ecuación (3.22).

3.5.2. Simulaciones del Generador de Números Aleatorios Real (TRNG)

Las simulaciones que se muestran a continuación se realizaron utilizando parámetros del proceso en tecnología de 0.18 μm de TowerJazz y una tensión de alimentación de 1,8 V con la herramienta Virtuoso de Cadence.

En la Figura 3.25 se muestra una simulación transitoria de la tensión de salida del VCO cuando la entrada de tensión de la fuente de corriente PMOS se encuentra en $Vdd/2$, es decir 0,9 V logrando que el VCO oscile a la frecuencia nominal de 105,2 KHz. Se observa que la tensión de salida se encuentra centrada en 0,9 V. y oscila entre 1,8 V y 0 V. A la salida del VCO se inserta una cadena de 3 inversores con el fin de disminuir el tiempo de trepada de esta señal y cumplir con

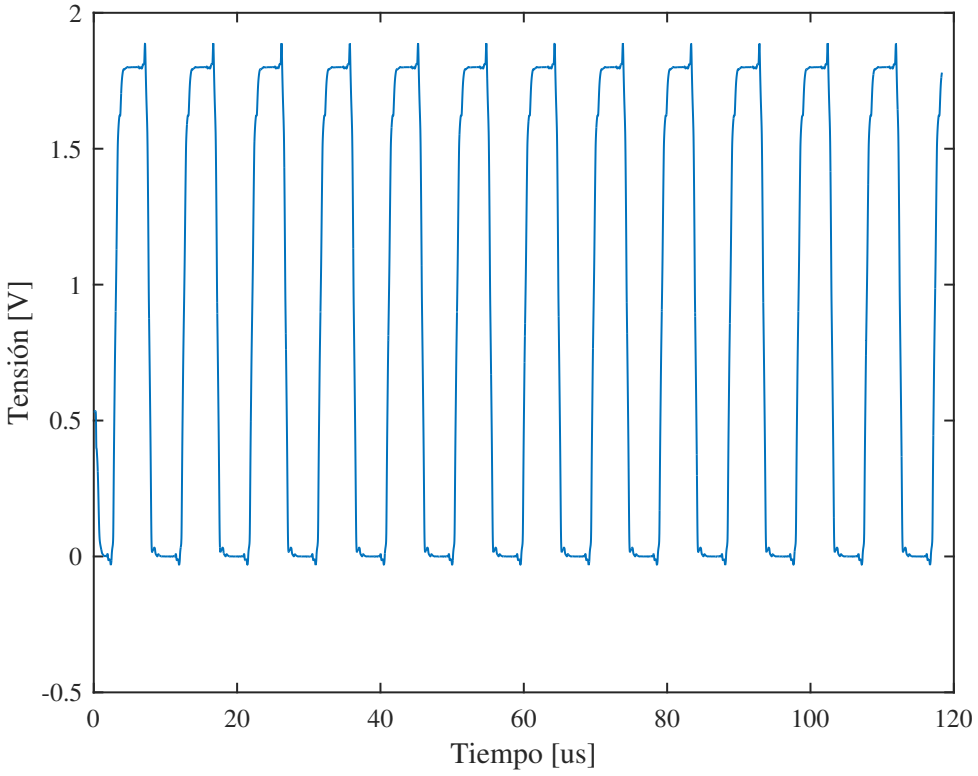


FIGURA 3.25: Simulación temporal de la salida del VCO.

el temporizado para que el flip flop muestree un dato válido. En la Figura 3.26 se indican los tamaños de los transistores en la cadena de inversores.

El diseño se realizó para que la tensión aplicada al transistor PMOS sea la que controle la corriente por las ramas inversoras que conforman el VCO. Con el fin de verificar dicho comportamiento, en la Figura 3.27 se muestra la variación de la corriente sobre las ramas inversoras cuando se varía la tensión del *gate* de este transistor. Como se muestra en la simulación, el diseño contempla que en el rango

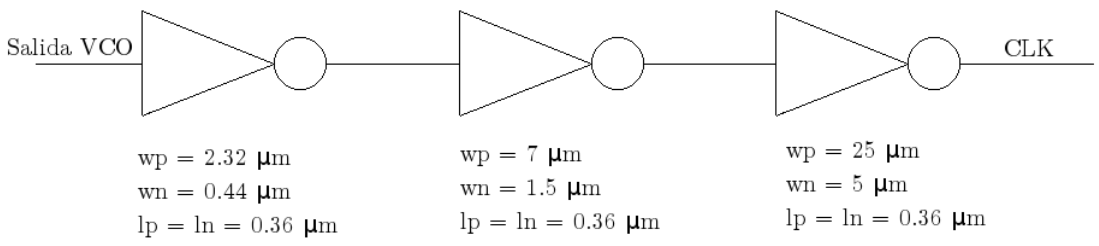


FIGURA 3.26: Dimensiones de la cadena de inversores.

de variación de ±50 mV en torno a 0,9 V, la corriente siga siendo controlada por las fuentes de corriente conformada por los PMOS.

Una característica a tener en cuenta en el desempeño del VCO es su linealidad en el rango de sintonización. En la Figura 3.28 se muestra la relación entre la tensión y la frecuencia de la salida. Como se puede observar, para la variación de tensión de entrada desde 0,85 V a 0,95 V, la frecuencia varía desde 98,10 KHz a 112, 5 KHz con una respuesta lineal dentro de estos rangos.

En la Figura 3.29 se muestra la simulación temporal de la salida del oscilador de anillos a una frecuencia de 695 MHz. La señal de salida del VCO, luego de una cadena de tres inversores, es la misma que ingresa a la entrada de reloj de los 8 flip flops que se utilizan para obtener 8 bits aleatorios de manera simultánea, como se muestra en el esquemático de la Figura 3.30. Para tal fin, cada señal de entrada hacia los flip flops debe ser distinta y por esto se utilizan 8 osciladores

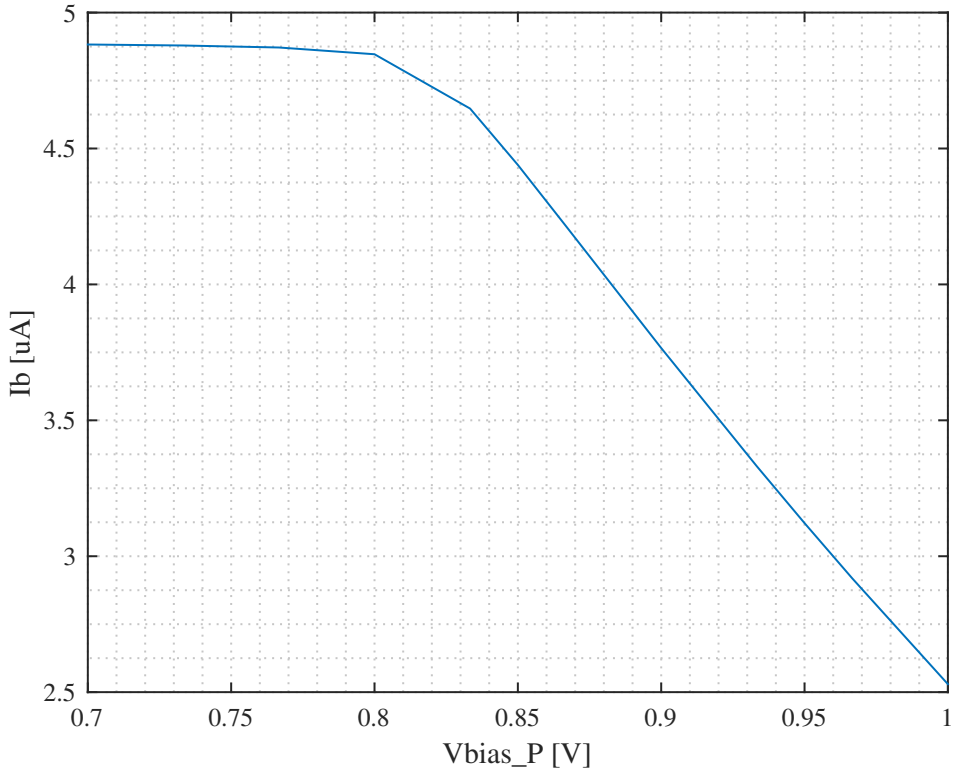


FIGURA 3.27: Corriente por las ramas inversoras en función de la tensión de control del transistor PMOS M14.

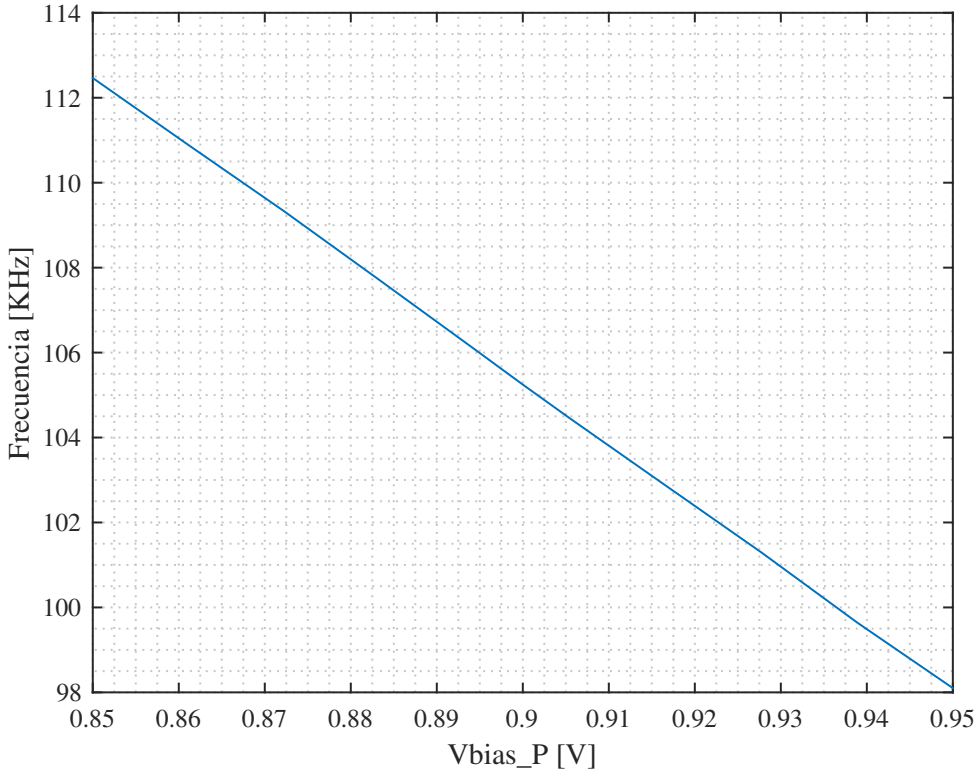


FIGURA 3.28: Característica de tensión vs frecuencia de salida del VCO.

de anillos con 7 etapas inversoras cada uno. En el caso de la simulación, como no se pueden contemplar las diferencias entre los osciladores que puedan ocurrir en el proceso de fabricación, se realizó el último inversor de la cadena con una pequeña desigualdad en el tamaño de los transistores con el fin de generar bits de salida diferentes. Los valores de tensiones referidos en el gráfico bajo los cuales se realizaron las simulaciones son Vbias = 0,9 V, Vbias_P = 0,9 V y Vbias_N =0,5 V.

Los cambios en la tensión de alimentación y en la temperatura, producen variación en la frecuencia del oscilador controlado por tensión y en el oscilador de anillos. Estos cambios pueden afectar la salida del generador si la variación de frecuencias es tal que provoque que el período del oscilador de anillos sea menor o muy cercano al máximo permitido. Por este motivo, es necesario realizar todas las posibles variaciones y analizar cuál es el peor caso. Para tal fin, se realizó una simulación paramétrica asumiendo que se producen cambios de la temperatura de 0°C a 140°C, y de la tensión de alimentación en ±10 %. Se asume que la tensión de control de la fuente de corriente del VCO varía con V_{dd} de la misma forma.

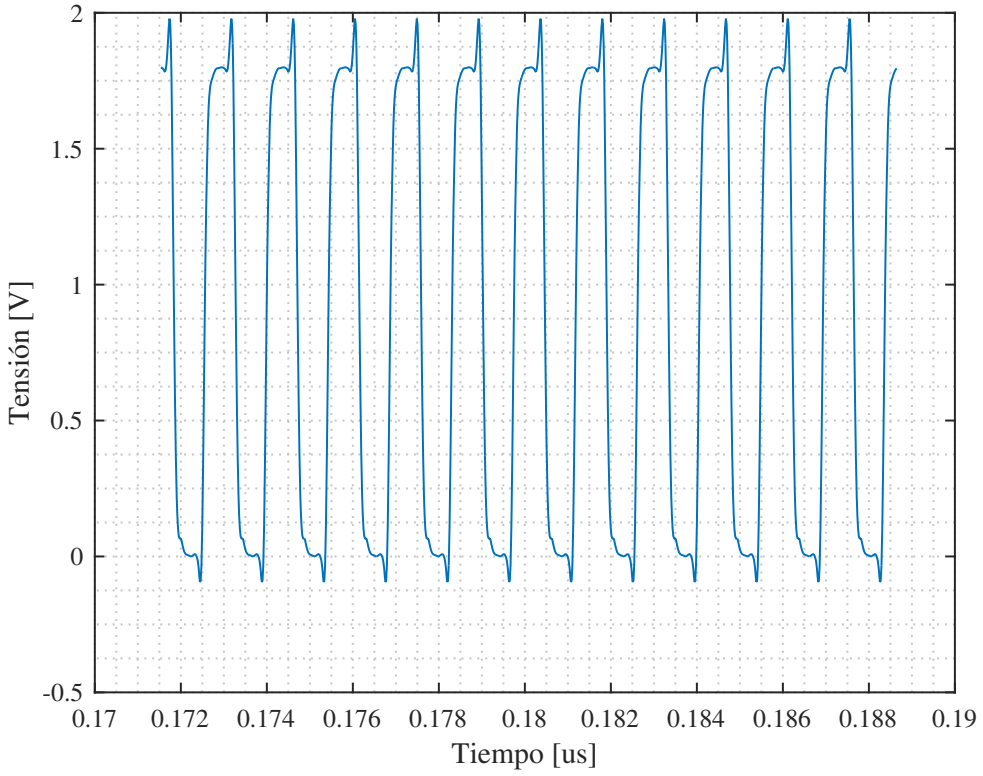


FIGURA 3.29: Oscilador de anillos en condiciones nominales.

En la Figura 3.31 se muestra la variación de la frecuencia del VCO modificando V_{dd} en $\pm 10\,\%$ ($1{,}62V \leq 1{,}8V \leq 1{,}98V$) para un rango de temperatura de 0°C a 140°C con un dispositivo NOMINAL. Luego se almacenan los valores de la frecuencia en los casos extremos de 0°C y 140°C. La misma simulación se realiza teniendo en cuenta las variaciones en el proceso, como puede ser el dopado, provocando dispositivos más lentos o más rápidos que los nominales. Los resultados para los dispositivos *slow*, *fast*, *slow-fast* y *fast-slow* se muestran en las Tablas 3.5 y 3.6.

En la Figura 3.32 se muestra la simulación del oscilador de anillos para una variación de V_{dd} en $\pm 10\,\%$ entre un rango de temperatura de 0°C a 140°C para un dispositivo tipo nominal. El peor de los casos puede llegar a ser un aumento de temperatura máximo cuando la tensión de alimentación está por debajo del valor nominal, en este momento la frecuencia del oscilador puede ser muy baja comparada con la del oscilador controlado por tensión, aumentando la posibilidad que el generador produzca una salida con cierta desviación hacia algún valor. En las

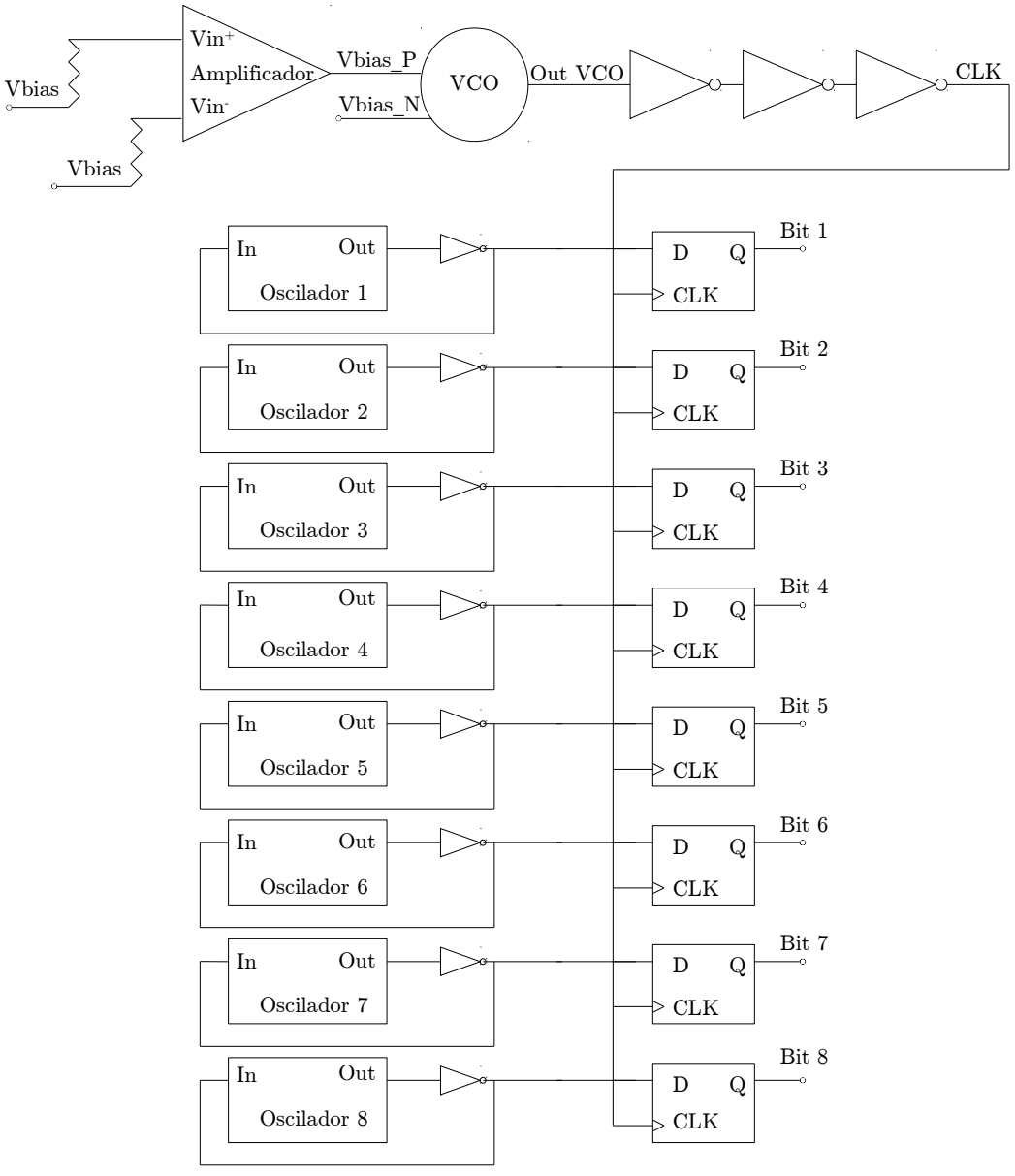


FIGURA 3.30: Esquemático del generador de números aleatorios.

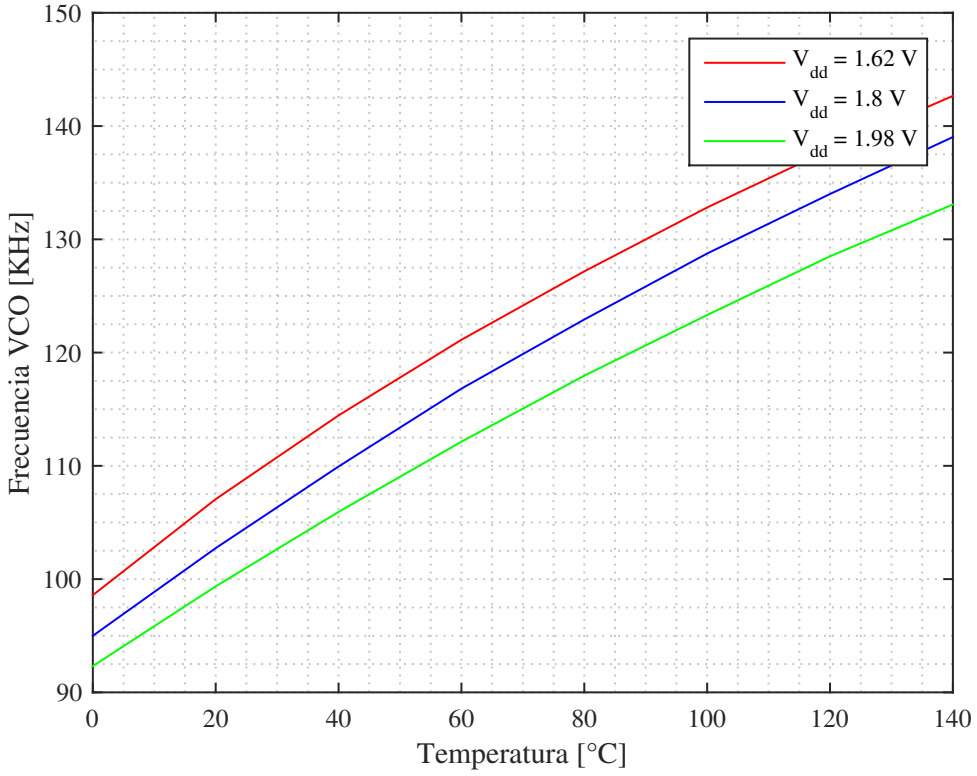


FIGURA 3.31: Frecuencia de oscilación del VCO en función de la temperatura para un dispositivo FET NOMINAL.

Tablas 3.7 y 3.8 se muestran las frecuencias del oscilador de anillos para los peores casos de variación del dispositivo, temperatura y tensión de alimentación.

Por último, para cada uno de los posibles casos extremos de las tablas anteriores, se calcula la variación de frecuencia producida para cada tensión de entrada al VCO, Vbias_P nominal ±50 mV, como se indica en la Ecuación (3.23). Esta variación, corresponde a la mínima frecuencia que puede tener el oscilador rápido. En las Tablas 3.9 y 3.10 se muestran la relación entre la variación temporal producida por el VCO, calculada como en la Ecuación (3.20), y el período del oscilador rápido para los distintos tipos de operación. En dichas tablas se puede ver que aún teniendo en cuenta los casos extremos, el anillo sigue oscilando a una frecuencia muy elevada en relación a la variación dada por el VCO. Cuando la tensión de alimentación se encuentra un 10 % por encima de su valor nominal, a 140°C y un dispositivo fast-slow, la relación entre las dos frecuencias es mínima pero de todas formas sigue siendo lo suficientemente grande para considerar que no se verá afectada la

Tabla 3.5: Frecuencia de oscilación del VCO en función de la tensión de alimentación, la temperatura y el dispositivo (nominal, fast y slow).

	Frecuencia VCO [KHz]					
	Nominal		Fast		Slow	
Vdd [V]	0°C	140°C	0°C	140°C	0°C	140°C
1,62	98,58	142,66	117,86	163,57	82,06	124,97
1,8	95	139,03	113,97	158,56	79,72	122,01
1,98	92,32	133,07	110,70	152,18	77,86	117,05

Tabla 3.6: Frecuencia de oscilación del VCO en función de la tensión de alimentación, la temperatura y el dispositivo (fast-slow y slow-fast).

	Frecuencia VCO [KHz]			
	Fast-Slow		Slow-Fast	
Vdd [V]	0°C	140°C	0°C	140°C
1,62	103,69	144,51	94,18	140,26
1,8	99,33	143,58	92,03	135,43
1,98	96,19	137,10	90,69	130,17

Tabla 3.7: Frecuencia del oscilador de anillos en función de la tensión de alimentación, la temperatura y el dispositivo.

	Fosc Anillo [MHz]					
	Nominal		Fast		Slow	
Vdd [V]	0°C	140°C	0°C	140°C	0°C	140°C
1,62	646,77	501,74	712,05	553,47	595,25	458,79
1,8	732,14	567,46	799,66	622,50	680,69	523,09
1,98	806,27	628,97	874,73	686,69	755,97	583,83

Tabla 3.8: Frecuencia del oscilador de anillos en función de la tensión de alimentación, la temperatura y el dispositivo.

	Fosc Anillo [MHz]			
	Fast-Slow		Slow-Fast	
Vdd [V]	0°C	140°C	0°C	140°C
1,62	643,56	500,43	652,39	503,76
1,8	728,69	566,11	739,21	570,15
1,98	802,95	627,77	814,4	632,27

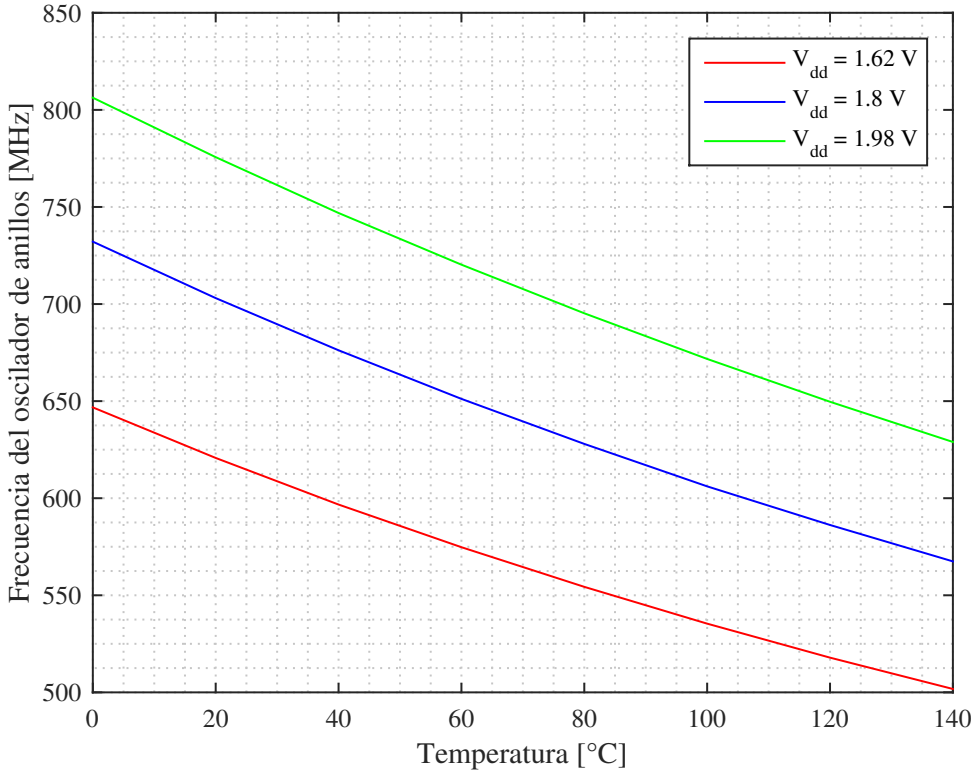


FIGURA 3.32: Frecuencia del oscilador de anillos en función de la temperatura para un dispositivo FET NOMINAL.

salida del generador. Luego de exponer los casos extremos del VCO, a continuación se consideran los resultados obtenidos con respecto a las características del ruido. Una simulación de la señal de ruido generado por las resistencias se ilustra en la Figura 3.33, y en la Figura 3.34 la misma señal en conjunto con la señal amplificada. La variación de tensión resultante de la señal amplificada es mayor a los 50 mV en torno a $V_{dd}/2$ esperados. En el caso de las simulaciones esta variación

TABLA 3.9: Relación entre el período producido por la variación de frecuencias del VCO y frecuencia del oscilador rápido para los dispositivos nominal, fast y slow.

	$\frac{\Delta T_{VCO}}{T_{ring}}$					
	Nominal		Fast		Slow	
Vdd [V]	0°C	140°C	0°C	140°C	0°C	140°C
1,62	560,33	287,64	504,17	258,85	636,65	313,01
1,8	510,49	210,15	459,61	210,32	582,52	229,93
1,98	444,38	203,55	405,80	198,61	527,15	214,00

TABLA 3.10: Relación entre el período producido por la variación de frecuencias del VCO y frecuencia del oscilador rápido para los dispositivos fast-slow y slow-fast.

	$\frac{\Delta T_{VCO}}{T_{ring}}$			
	Fast-slow		Slow-Fast	
Vdd[V]	0°C	140°C	0°C	140°C
1,62	555,63	-673,14	618,54	273,62
1,8	454,83	196,92	542,78	234,49
1,98	415,15	187,78	542,17	212,27

no provocó un efecto considerable en las características de salida del generador, si esto sucede en la validación experimental, el amplificador permite modificar su ganancia en forma externa al circuito integrado.

En las Figuras 3.35 y 3.36 se muestran la distribuciones de la entrada del amplificador y de la salida. Como se ve, la señal de ruido amplificada tiene una variación mayor a la esperada. El histograma de entrada que se forma a partir de 38514 muestras de la señal de ruido generada por las resistencias, presenta una media $\mu = 900$ mV y una desviación estándar $\sigma = 183{,}9$ μV. Con respecto a el histograma de salida de la Figura 3.36, el mismo se forma a partir de la misma cantidad de muestras tomadas de la señal ruidosa amplificada proveniente de las resistencias. Como se observa, la distribución de las muestras se aproxima a una distribución gaussiana con media $\mu = 899{,}59$ mV y desviación estándar $\sigma = 43{,}15$ mV.

En la Figura 3.37 se muestra la simulación de la variación de la frecuencia de salida del VCO en función del tiempo y en la Figura 3.38 la distribución de las muestras obtenidas a partir de la simulación temporal. En este caso, con un total de 20961 muestras, la distribución aproxima a una gaussiana con media 104,81 KHz y desviación estándar de 35,89 KHz. Como el VCO se encuentra dentro de su rango lineal de trabajo, si la distribución de la señal de tensión de entrada aproxima a una gaussiana, se espera que la distribución de la frecuencia de salida siga la misma forma.

En la Figura 3.39 se observa la salida directamente del VCO, y la misma señal luego de pasar por la cadena de tres inversores.

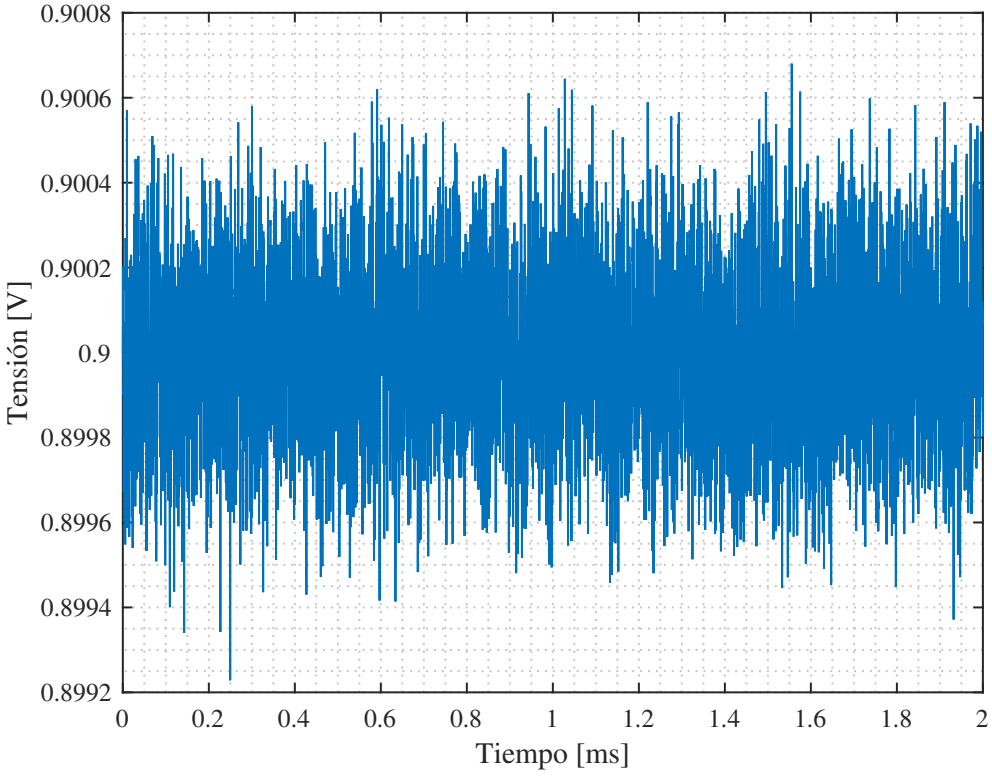


FIGURA 3.33: Entrada al amplificador.

En la Figura 3.40 se muestra la señal de entrada de reloj del flip flop (CLK), la salida del oscilador de anillos ingresando como dato al flip flop D (Dato), y la salida resultante (Bit 1) obteniéndose un bit de salida del generador cada 10 μs.

3.5.3. Resultados de las pruebas de aleatoriedad

Para analizar la salida de los números aleatorios generados, se utilizan las pruebas estadísticas del NIST.

El NIST brinda dos maneras distintas de realizar la interpretación de los resultados: por un lado propone examinar la proporción de secuencias que pasan el test estadístico, por otro examinar la distribución de los valores p con el fin de verificar uniformidad.

- Proporción de las secuencias que pasan el valor del test estadístico: se calcula haciendo la relación entre las secuencias que arrojan un valor p con α mayor

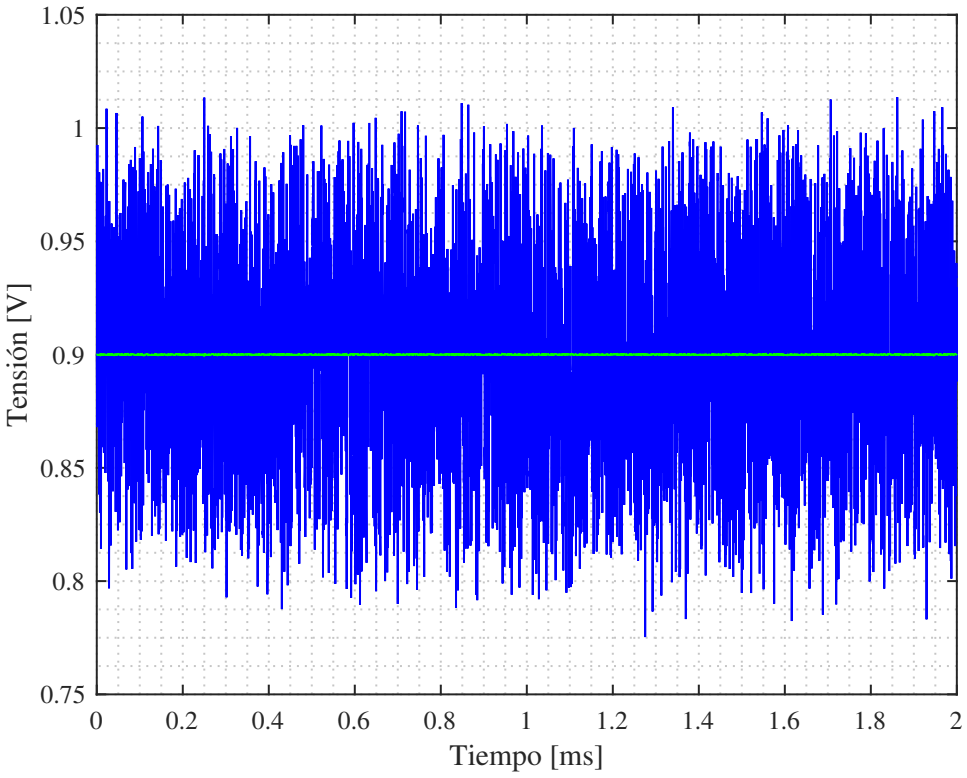


FIGURA 3.34: Entrada y salida del amplificador.

o igual que el valor crítico establecido, sobre el número total de secuencias verificadas. El rango aceptable de esta proporción está determinado utilizando el intervalo de confianza calculado utilizando una distribución normal como una aproximación a la distribución binomial, lo cual es razonable siempre y cuando el tamaño de la muestra sea lo suficientemente grande (≥ 1000).

- Distribución uniforme de los valores p: una forma de ver la distribución de los valores p es realizar un histograma en donde el intervalo entre 0 y 1 se divide en 10 sub-intervalos, y el resultado de cada valor p se ubica en el sub-intervalo que corresponda. Con los valores individuales se calcula un nuevo valor p que se utiliza para determinar si existe uniformidad.

Para ejecutar cada test, es necesario tener en cuenta distintos parámetros como el tamaño de la muestra (cantidad de secuencias), la cantidad de bits en cada secuencia y el tamaño de los bloques. Con respecto a la cantidad de secuencias, este valor se encuentra ligado al nivel crítico α elegido para el test. Si por ejemplo

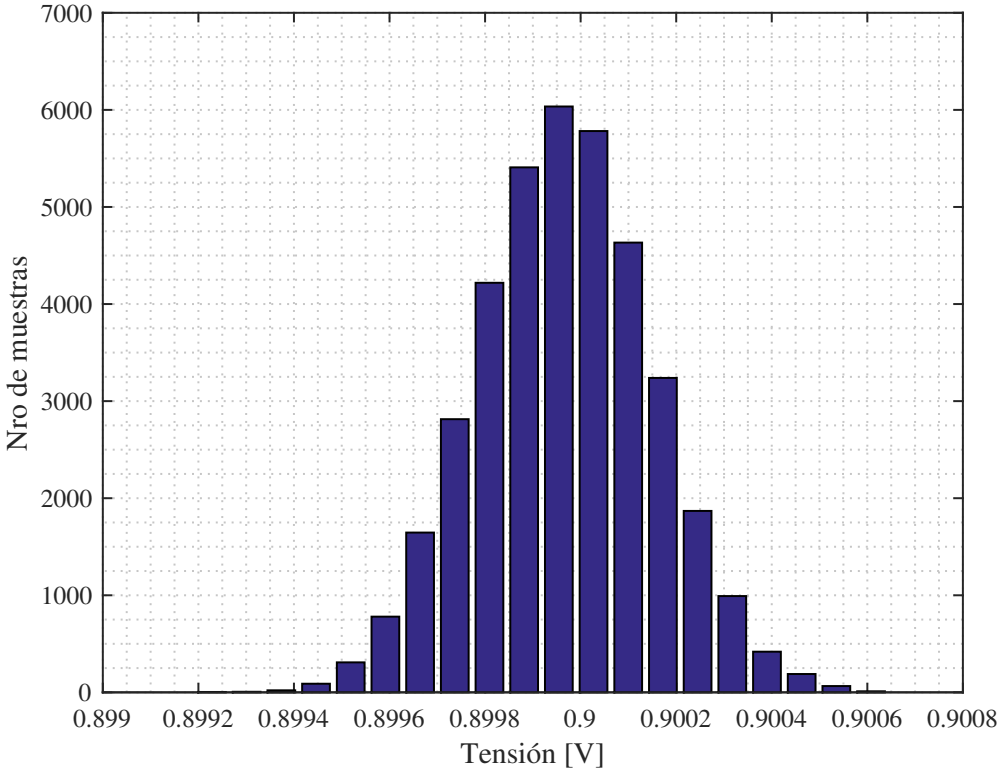


FIGURA 3.35: Distribución de las muestras de la señal de entrada con $\mu =$ 900,002 mV y $\sigma =$ 183,09 μV.

α es 0,001 entonces se espera que 1 de cada 1000 secuencias sea rechazada (que ocurra un error tipo 1 cada 1000 secuencias analizadas). Por lo tanto si se elije una muestra menor de 1000 será raro observar un rechazo y se podría hacer una conclusión errónea. Por este motivo se recomienda que el tamaño de la muestra sea alrededor del inverso del nivel crítico α. Con respecto a la longitud de la secuencia, depende de la prueba que se esté analizando. Por ejemplo, el de Maurer, requiere longitudes extremadamente grandes.

Cada test debe ser independiente uno de otro y se recomienda que el de frecuencias se aplique en primer lugar. La mayoría tienen el objetivo de (1) examinar la distribución de unos y ceros, (2) estudiar los armónicos de la cadena de bits utilizando métodos espectrales, o (3) detectar patrones.

Los bits de salida de cada flip flop se almacenan en un archivo de texto con ceros y unos. El programa del NIST se ejecuta indicando la longitud de las secuencias y luego la cantidad de muestras. Es decir si el archivo con el contenido de la salida

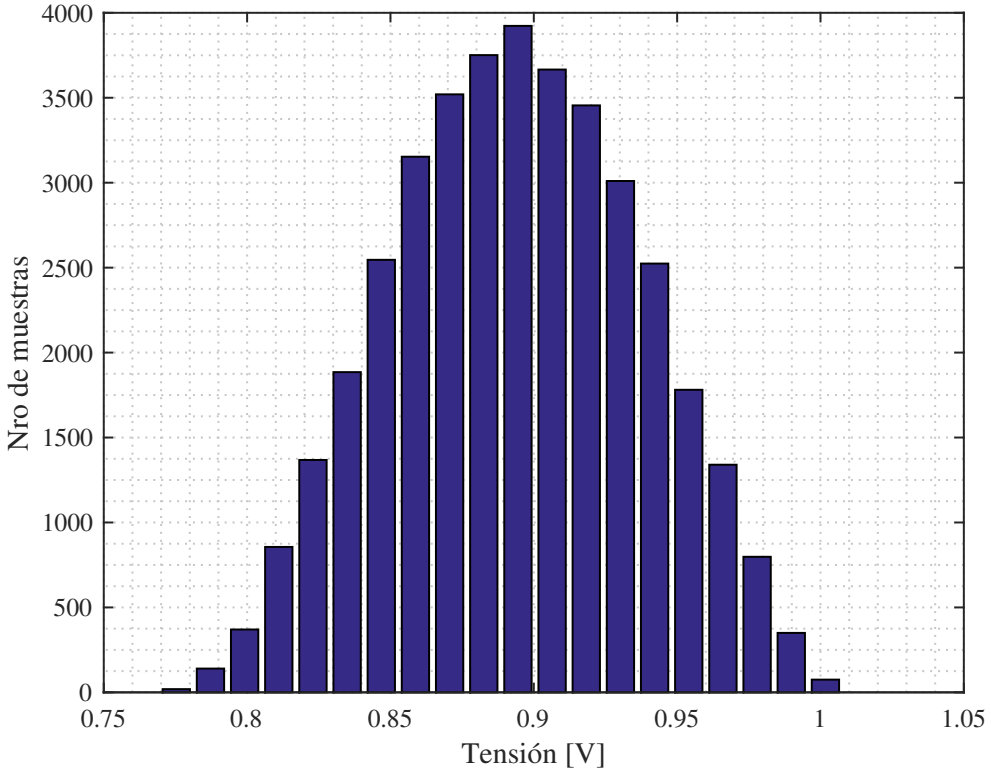


FIGURA 3.36: Distribución de las muestras de la señal de salida con $\mu =$ 899,59 mV y $\sigma =$ 43,15 mV.

del generador es de 10000 bits, y cada secuencia es de 1000 bits, se puede tener un tamaño de muestra de como máximo 10. Es posible elegir si se realizarán todos los test o alguno en particular. Las pruebas se realizan bajo una muestra de 2320 bits luego de simular 2900 μs, se analizan 23 secuencias de 100 bits cada una y se ejecuta en primer lugar el test de frecuencias. Su propósito es analizar la uniformidad de 0 y 1. El algoritmo convierte los bits 0 en -1 y realiza una sumatoria. Con este resultado, para cada secuencia se calcula un valor-p en función de la función de error complementario. Luego se analiza la proporción de secuencias que pasan la prueba, verificando que todas arrojan un valor-p mayor que el valor crítico establecido.

Luego se realiza la prueba de frecuencias en bloques que analiza la proporción de unos en un bloque de M cantidad de bits. En el supuesto de aleatoriedad, se espera que la proporción de unos sea $M/2$. Cada secuencia se divide en bloques de longitud M. El código tiene establecido un valor $M = 20$. Los bits de salida se

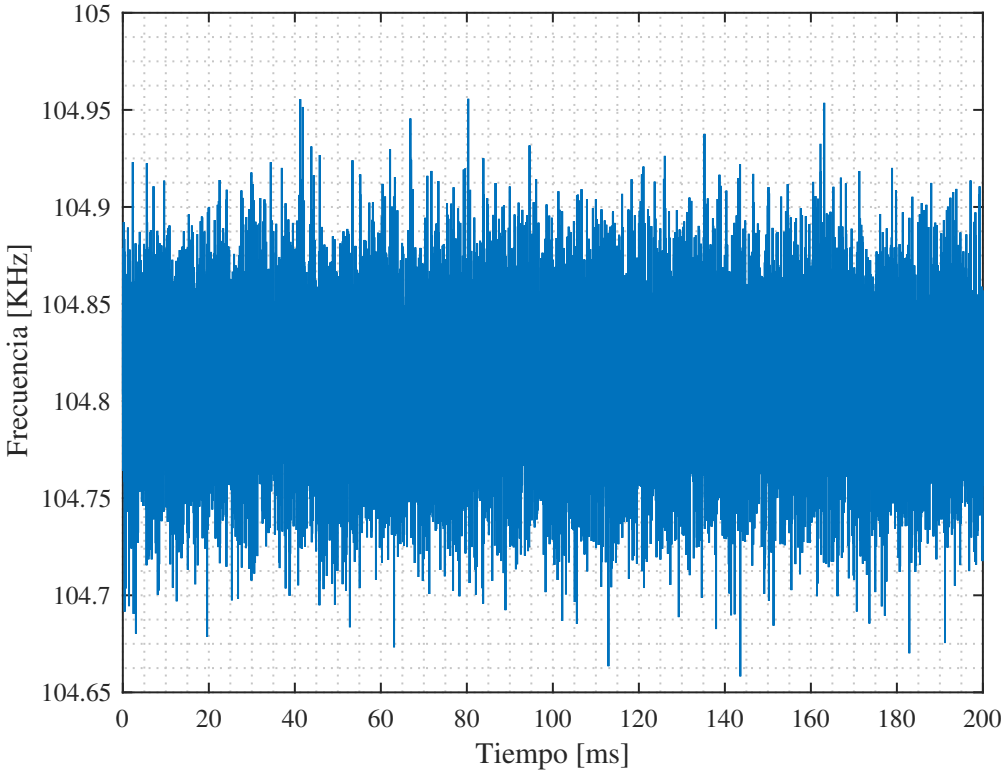


FIGURA 3.37: Variación de la frecuencia de salida del VCO en función del tiempo.

dividen en 23 secuencias de 100 bits y con el M seleccionado cada secuencia en 100/20 sub-bloques.

El run test cuenta el número de valores iguales sucesivos. Como requisito debe pasar el test de frecuencias. Con 23 secuencias de 100 bits cada una la prueba es satisfactoria siguiendo el criterio de proporción de secuencias.

La prueba de cadena de unos en un bloque se ejecuta con 18 secuencias de 128 bits cada una y pasa el test con el criterio de la proporción de secuencias.

El test de la transformada de Fourier discreta se ejecuta con los parámetros indicados en la Tabla 3.11 y tiene como objetivo identificar patrones repetitivos en la secuencia.

El test de entropía aproximada tiene como objetivo detectar la frecuencia de bloques solapados. Por último, se realiza el test de sumas acumulativas, en el que se

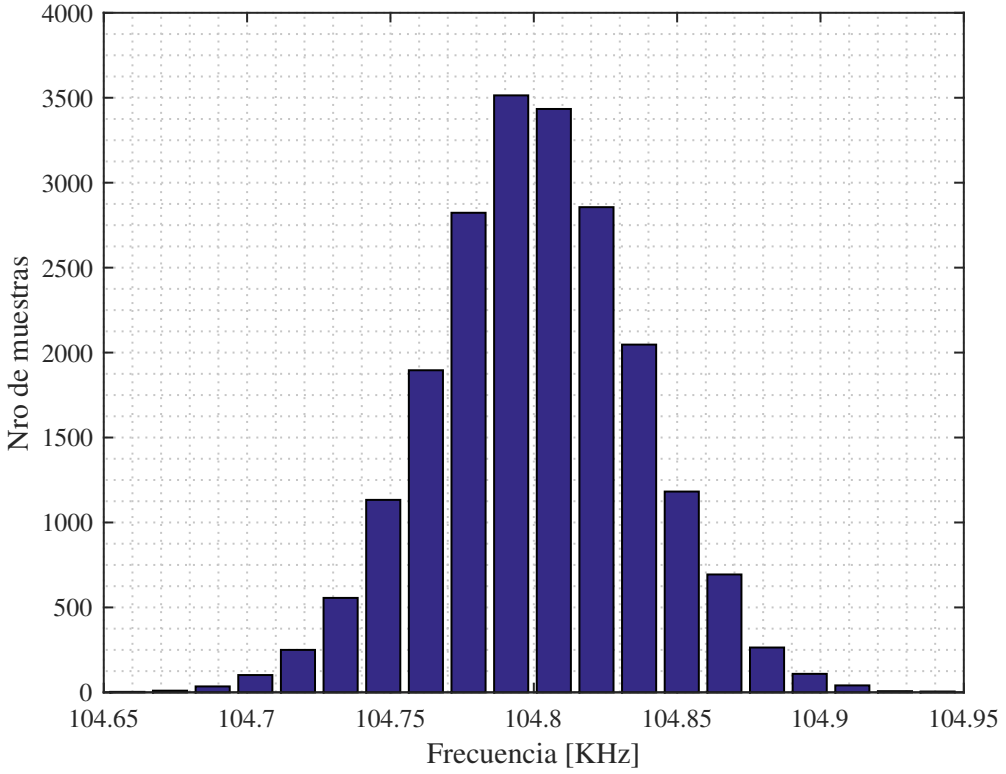


FIGURA 3.38: Distribución de la frecuencia de salida del VCO con μ = 104,81 KHz y σ = 35,89 KHz.

analiza si el resultado de estas sumas es similar al que se esperaría de una secuencia aleatoria.

En la Tabla 3.11 se muestran los resultados obtenidos. La primer columna indica el nombre del test, la segunda la cantidad de bits de cada secuencia, la siguiente el número total de secuencias y por último el resultado de la proporción de ellas que cumplen con los requisitos de la prueba estadística. Los test que no se muestran en la tabla no pudieron ser verificados por falta de cantidad de bits. El nivel de aceptación seleccionado es de α = 0,01 por lo tanto, para obtener resultados significativos se deberían verificar al menos 100 secuencias de longitud mínima de 100 bits. Debido a que los tiempos de simulación para obtener esta cantidad son elevados, se decide realizar la verificación de una mayor cantidad de muestras en el momento que se pruebe el circuito integrado, y continuar con el diseño para llegar a los tiempos impuestos por el fabricante del chip. En el caso que los bits de salida posean una desviación, el resultado puede ser corregido con la implementación de

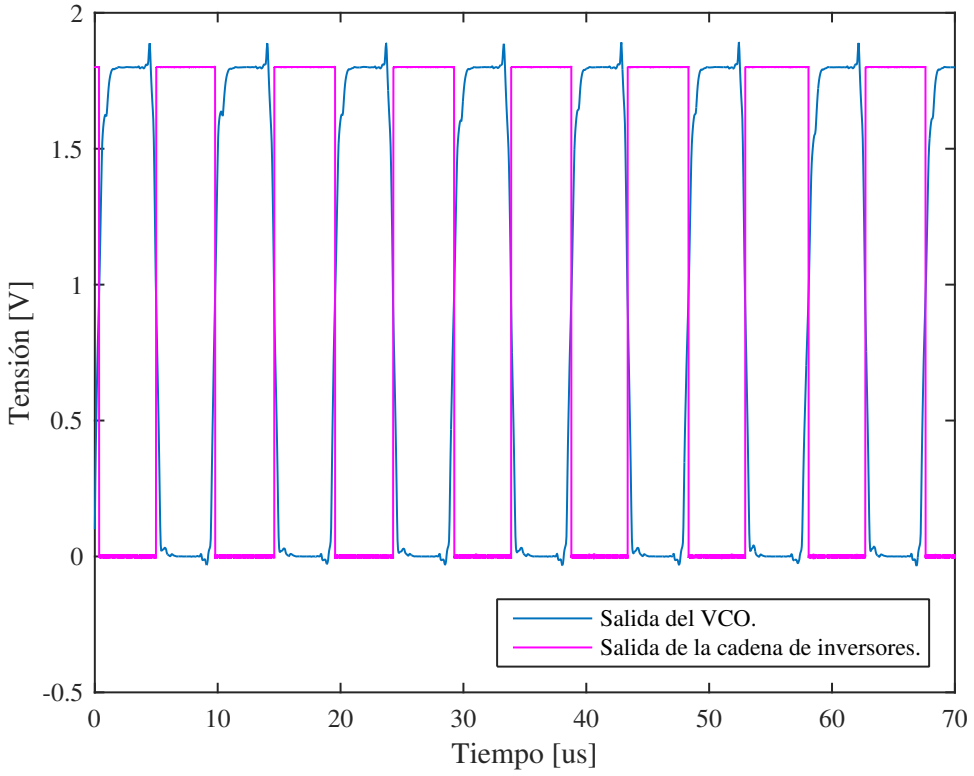


FIGURA 3.39: Salida del VCO y la misma señal después de la cadena de inversores.

un corrector (Neumann, 1963). Este corrector puede ser realizado en forma externa al chip, dentro de un FPGA, al momento de procesar la salida.

3.5.4. Interfaz AMBA APB del RNG

Con el fin de lograr incorporar el generador de números aleatorios a un sistema actual, se realizó la interfaz con el bus AMBA APB. Este bloque contiene las señales del protocolo, la salida del VCO y los 8 bits de datos del generador. Es un diseño completamente digital cuya descripción se realizó en VHDL.

Los bits de salida de cada flip flop y la señal de salida del VCO ingresan a un bloque digital que genera las señales necesarias para el bus APB, como se muestra en la Figura 3.41. En el caso del diseño, el generador no admite una operación de escritura, sólo se permiten lecturas de los datos de salida. El bus APB puede leer un dato siempre que el generador tenga uno nuevo, esta acción se controla por

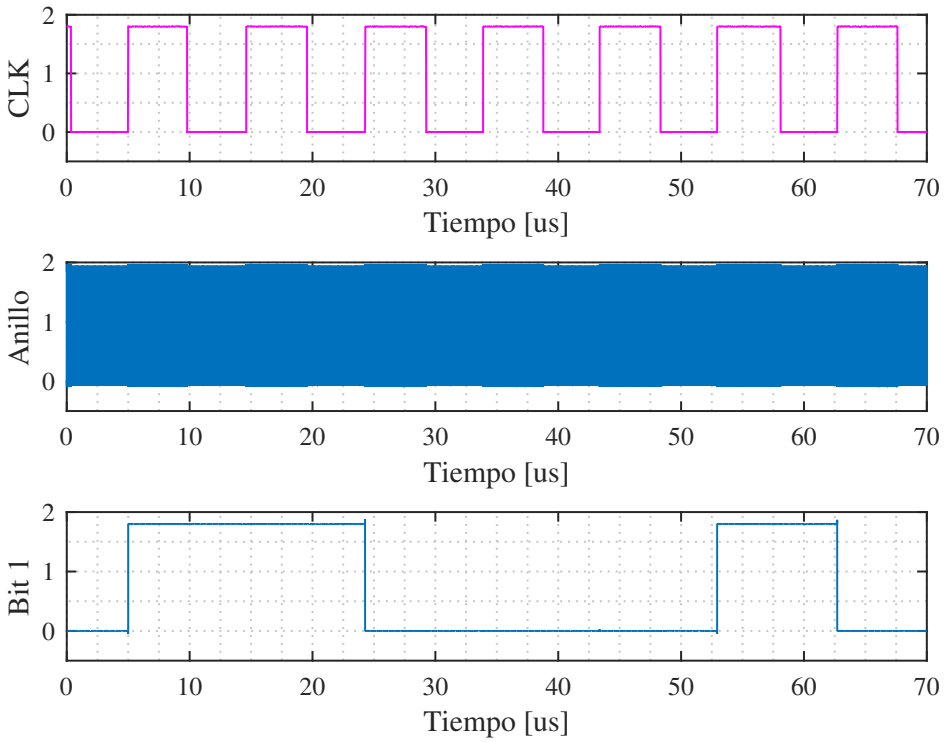


FIGURA 3.40: Muestreo de la señal aleatoria.

medio de una señal en particular del protocolo APB que agrega estados de espera en el caso que no se haya generado un nuevo dato aleatorio (*pready*).

El diseño de la interfaz se basa en los diagramas de estados que se muestran en las Figuras 3.42 y 3.43. El cambio de estados de la Figura 3.42 se controla por medio de la señal de salida del VCO, mientras que el de la Figura 3.43 se controla por medio del reloj del bus.

En referencia a ambas figuras, la señal de selección se establece en 1 cuando hay

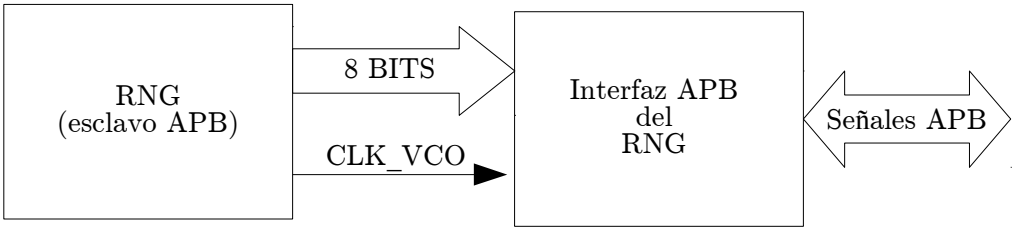


FIGURA 3.41: Diagrama en bloques.

Tabla 3.11: Resultado de las pruebas estadísticas.

Prueba estadística	n [2]	m [3]	Proporción de secuencias
Frecuencia (monobits)	100	23	23/23
Frecuencia (bloques) $M = 20$ [1]	100	23	22/23
Runs	100	23	23/23
Cadena más larga de unos (bloque) $M = 8$	128	18	16/18
Transformada de Fourier discreta	1000	2	2/2
Entropía aproximada	100	23	23/23
Sumas Acumulativas	100	23	23/23

[1] Bloques de bits en una secuencia.
[2] Cantidad de bits en cada secuencia.
[3] Cantidad de secuencias.

una transferencia. Entendiéndose como tal cuando el maestro selecciona en el bus al esclavo (en este caso el RNG) con la señal *psel*, coloca la dirección correcta en *paddr*, y coloca la señal de control de escritura y lectura *pwrite*.

La máquina de estados controlada por el VCO cuenta con tres estados: El primero denominado IDLE, el segundo GENERO_1, y el último GENERO_2. En el estado IDLE, si el maestro del bus selecciona este esclavo se hará una transición al nuevo estado, denominado GENERO_1. En el próximo flanco de reloj del VCO, independiente de cualquier otra señal, el siguiente estado es GENERO_2. La máquina de estados se mantiene alternando entre estos dos, hasta que la interfaz se reinicie, ingresando nuevamente al estado IDLE. En lo que respecta a la salidas de estos estados, se define una señal denominada *genero* que luego de un *reset* se inicializa en 0. En el estado GENERO_1 pasa a valer 1 y en el estado GENERO_2 pasa a valer 0, alternando en cada flanco de reloj entre estos dos valores.

La máquina de estados controlada por el reloj del sistema, cuya frecuencia teórica es de 1000 veces mayor que la del reloj generado por el VCO, realizará una transición del estado IDLE hacia el estado ESTADO_1 cuando el esclavo se selecciona. Las transiciones se llevan a cabo por las señales que se indican en la Figura 3.43. Con respecto a las salidas de los estados en ESTADO_1 y en ESTADO_2 la señal *pready* se coloca en 0 indicando que el generador todavía no tiene un nuevo dato

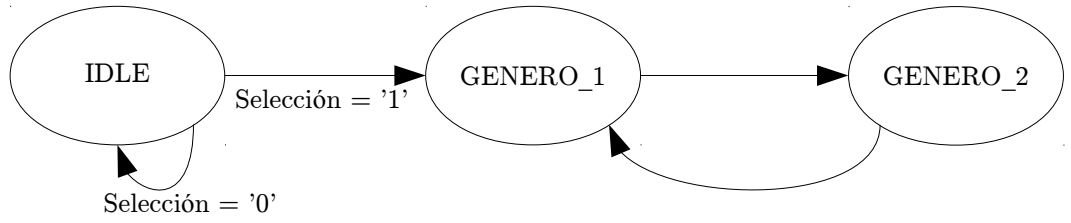


FIGURA 3.42: Almacenamiento de los bits aleatorios.

disponible y no puede ser leído por el maestro. En el estado DATO_DISPONIBLE esta señal se establece en 1 indicando que el maestro puede leer el número aleatorio.

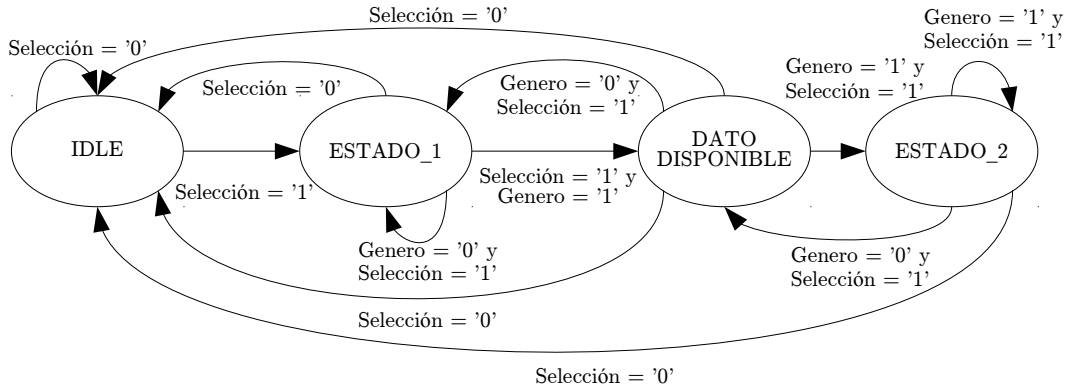


FIGURA 3.43: Diagrama de estados de la interfaz APB.

Como se indica en la simulación de la Figura 3.44, el esclavo se selecciona con la señal *psel* en 1, la dirección de selección del generador *paddr* = "111", y la señal *pwrite* en 0. El maestro selecciona el esclavo en la fase o estado actual denominado SETUP, momento que la señal de selección se establece en 1. En el próximo flanco de reloj *pclk* no ocurrió, hasta el momento, un nuevo flanco de la señal del VCO, por lo que la señal *pready* se coloca en 0 y se mantiene en el estado ESTADO_1 hasta el flanco positivo de *clk_vco* momento en el que se genera un nuevo dato aleatorio. La máquina de estados controlada por el reloj del sistema pasa al estado DATO_DISPONIBLE y el dato aparece en *prdata* terminando la transferencia en el próximo flanco de reloj del bus. En el estado denominado ESTADO_2 se espera hasta un nuevo flanco positivo de la señal del VCO y el proceso se repite de la misma forma que con el estado ESTADO_1.

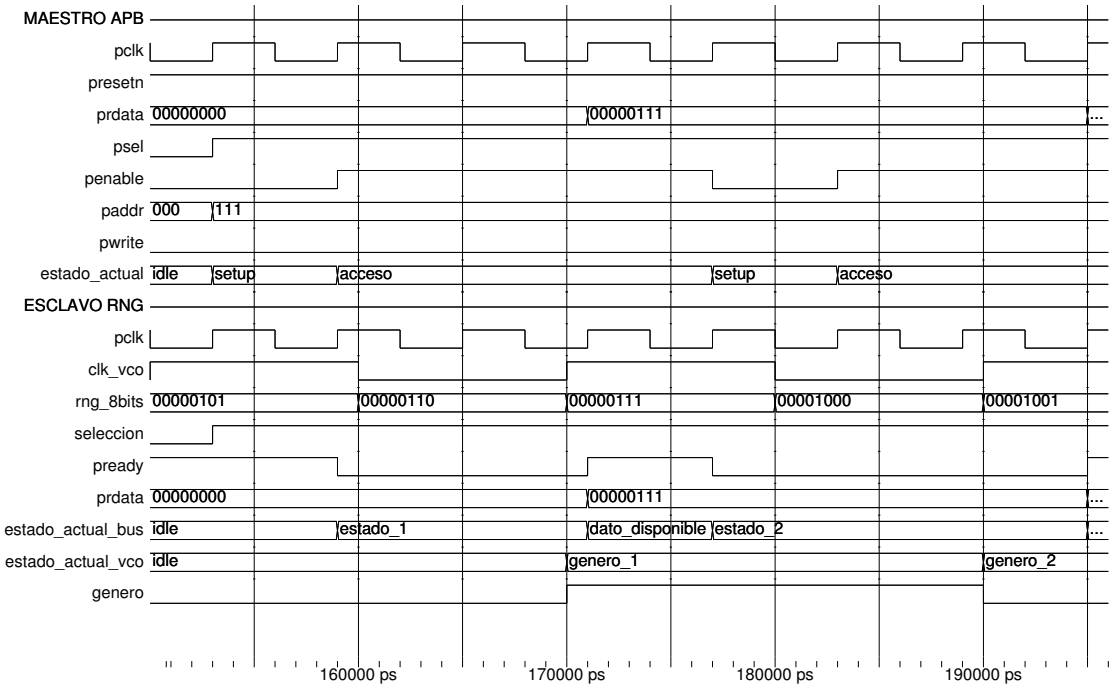


FIGURA 3.44: Simulación de la interfaz AMBA APB del generador.

3.5.5. Implementación física del generador

Una vez verificado el correcto funcionamiento a través de simulaciones, se realizó la síntesis lógica del diseño por medio de la herramienta Design Vision de Synopsys. Esta herramienta necesita como entrada los archivos de la tecnología, los archivos con la descripción VHDL y restricciones de temporizado. En estas restricciones se indica cuáles son los relojes en el diseño, en este caso el reloj del bus PCLK (con un período de 10 ns) y el de salida del VCO, CLK_VCO (con un período de 10000 ns). Por otro lado se indica que estos relojes corresponden a caminos independientes entre los bloques controlados por uno y otro, y comandos para establecer el temporizado de las transiciones de los relojes y sus incertezas. Luego que la herramienta realiza la optimización según el diseño, las restricciones dadas y opciones de compilación, se obtiene como salida un *netlist* del diseño con las compuertas de la tecnología. Como resultado de la optimización, el diseño cumple con las especificaciones dadas.

La síntesis física de la interfaz AMBA APB del generador se realizó con la herramienta de implementación digital de Cadence, Encounter. Como entrada, dicha

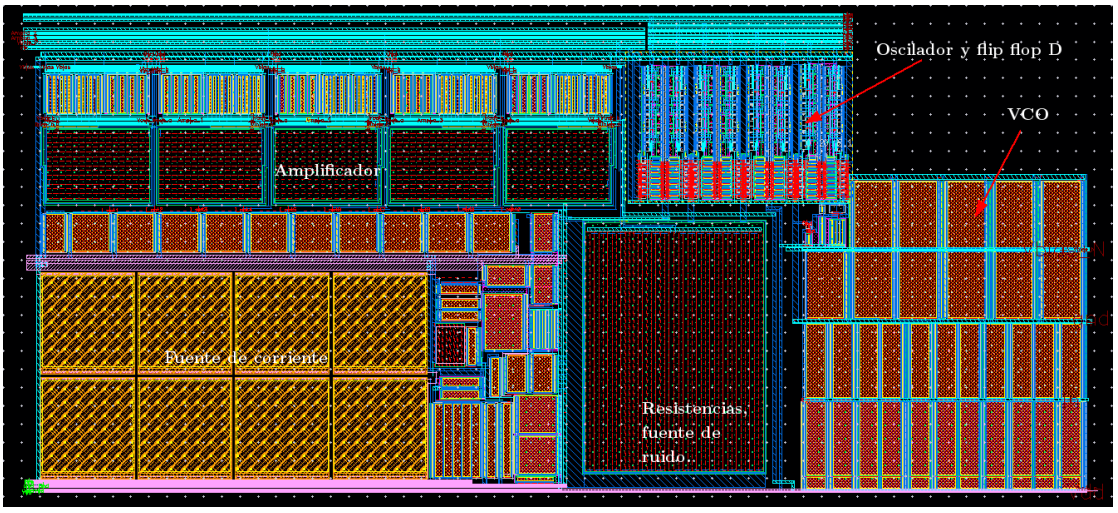


FIGURA 3.45: *Layout* del generador de números aleatorios.

herramienta necesita el archivo Verilog que contiene el netlist de salida del Design Vision. Además, necesita de la definición de un módulo de mayor jerarquía en el que se instancia el bloque diseñado y los pads de entrada y salida. La síntesis física de esta interfaz se realizó en conjunto con la interfaz de comunicación serie.

La disposición de las máscaras (*layout*) para la fabricación del generador se muestra se muestra en la Figura 3.45, se realizó de tal forma que las conexiones entre bloques sean sencillas de realizar y manteniendo la forma lo más rectangular posible. Con respecto a las entradas y salidas, se puede ver en la figura, sobre la parte inferior las líneas por donde ingresan las señales de alimentación V_{dd} y Gnd, la tensión Vbias_N de 0,5 V para polarizar los transistores NMOS de la fuente de corriente del VCO y la señal Rm en donde se conecta la resistencia externa de polarización de la fuente de corriente. En la parte superior derecha, se encuentran los bits de salida de cada uno de los 8 flip flop D. Por último en la parte superior izquierda, de abajo hacia arriba, se encuentran las líneas de conexión Vbias, CLK como salida del VCO luego de pasar por los tres inversores, Vbias_P, Amp_a_3, Amp_a_2, estas tres últimas se utilizan para monitorear las salidas de las tres etapas finales del amplificador.

En la Figura 3.46 se muestra el *layout* del VCO, en la parte superior se ubican los transistores PMOS de la fuente de corriente PMOS del VCO. Como se puede observar, estos transistores son de un ancho considerablemente menor que los

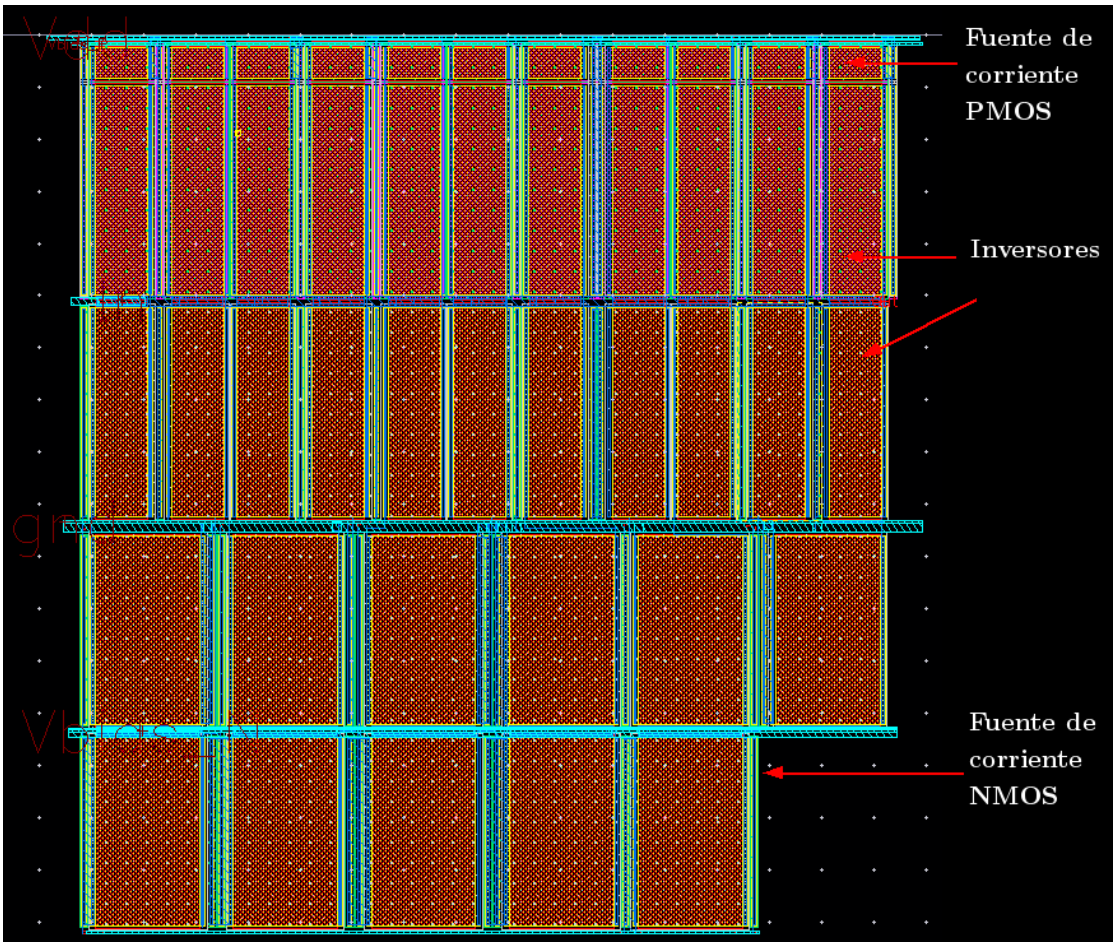


FIGURA 3.46: *Layout* del oscilador controlado por tensión.

transistores NMOS de las dos filas inferiores, correspondientes a los transistores de la fuente de corriente del VCO tipo NMOS. Por último, los transistores PMOS y NMOS de las mismas dimensiones corresponden a los transistores inversores del oscilador de anillos que conforman el VCO. Este diseño ocupa un área de 78 μm $\times$ 86 μm.

A continuación, en la Figura 3.47, se ilustra el *layout* del oscilador de anillos. En este nivel, se pueden ver sólo 6 inversores, dado que se mantuvo la diferencia mínima en el tamaño del séptimo inversor que se tuvo en cuenta en un bloque de mayor jerarquía para observar aleatoriedad en la simulación. Tampoco se observa en este caso la conexión entre la entrada y la salida para completar el oscilador, de la misma forma, esta conexión se realiza en un nivel de mayor jerarquía. Este bloque ocupa un área dentro del chip de 10 μm $\times$ 7 μm.

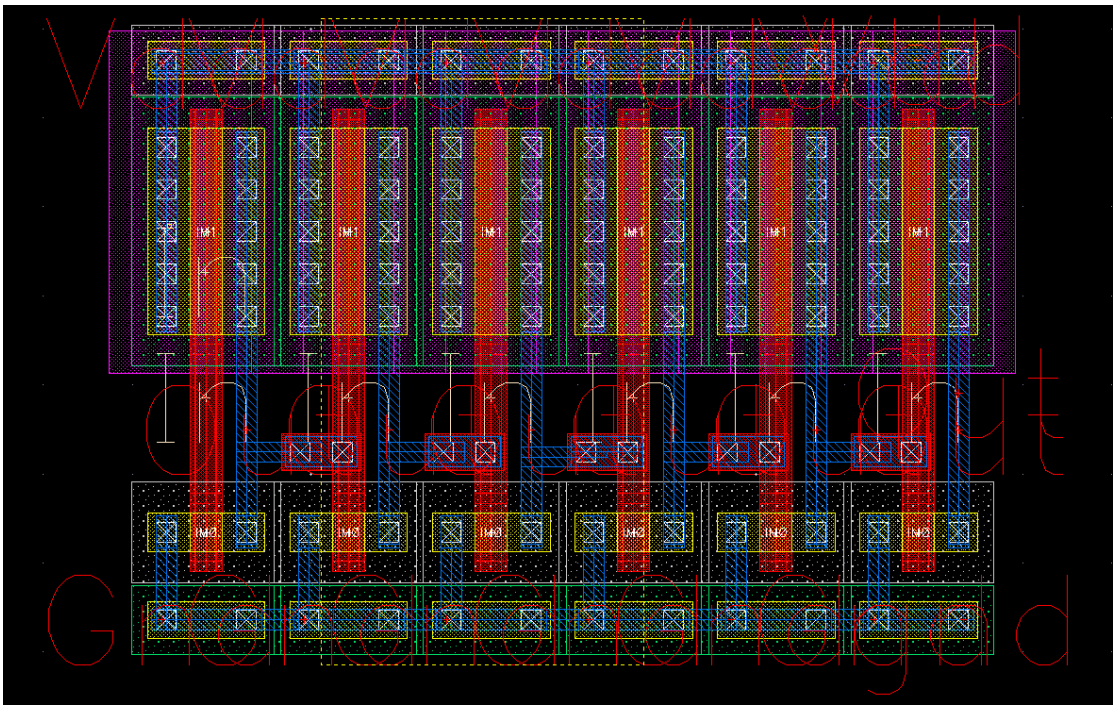


FIGURA 3.47: *Layout* del oscilador de anillos.

En la Figura 3.48 se muestra el *layout* del amplificador. En la parte derecha se encuentran las resistencias de las que se obtiene el ruido térmico para generar aleatoriedad. En la parte superior se observan las cinco etapas del amplificador, y lo que resta corresponde a la fuente de corriente. Este diseño ocupa un área de 200 μm $\times$ 121 μm.

Por último, en la Figura 3.49 se observa una única etapa del amplificador. El par diferencial de entrada son los transistores NMOS centrales, y la carga activa los transistores PMOS de la izquierda. La entrada de corriente para la polarización se encuentra en las líneas de metal centrales e ingresan a los *sources* de los transistores del par. Este bloque ocupa un área dentro del chip de 30 μm $\times$ 42 μm.

El *layout* completo del generador de números aleatorios sin considerar su interfaz APB ocupa un área total de 296 μm $\times$ 132 μm.

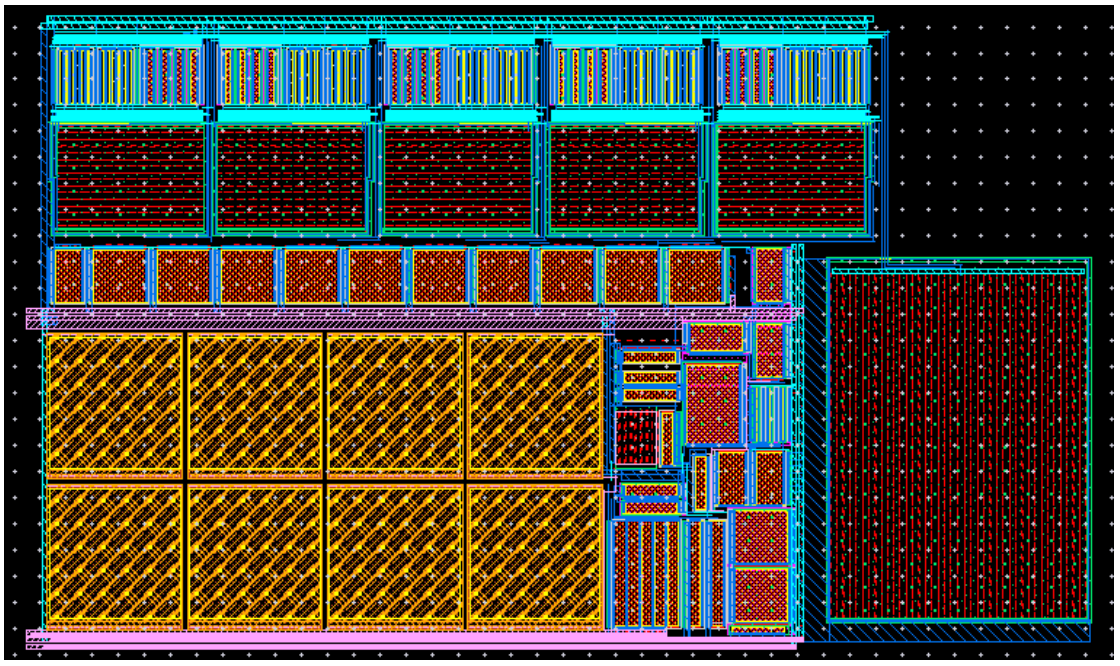


FIGURA 3.48: *Layout* del amplificador y fuente de corriente.

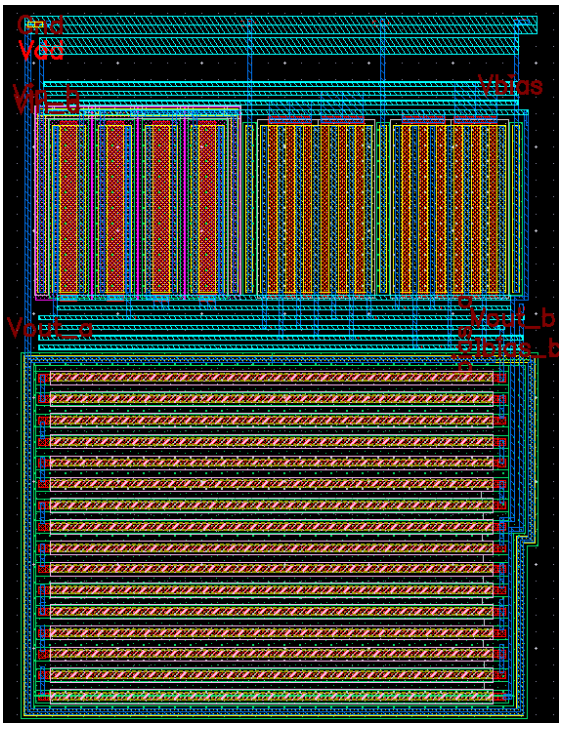


FIGURA 3.49: *Layout* de una etapa del amplificador.

Capítulo 4

Receptor-Transmisor Asíncrono Universal (UART)

4.1. Descripción

Un Receptor-Transmisor Asincrónico Universal (UART) es un dispositivo que convierte transferencias de información entre la forma paralela y serie. Funciona como interfaz entre un dispositivo maestro, que se comunica en paralelo a alta velocidad, y dispositivos esclavos, que realizan transferencias en serie a velocidades más bajas. Existen diferentes versiones de UARTs en el mercado, pero todas se basan en el mismo principio. Por lo general, son usadas en conjunto con estándares de comunicación como EIA, RS-232, RS-422 ó RS-485.

Tanto el transmisor como el receptor utilizan una señal de sincronismo, cuya frecuencia debe ser 16 veces mayor a la tasa de transferencia requerida. Esta señal se utiliza como base de tiempo para generar la salida en el transmisor y reconocer los bits de entrada en el receptor.

Se pueden mencionar dos tipos de comunicación serie: por un lado la tipo *full duplex*, en la que se puede enviar y recibir un dato en forma simultánea, y por otro, la *half duplex* en la que el dato puede ser enviado o recibido, pero no al

mismo tiempo. La comunicación serie llevada a cabo entre dos UARTs es *full-duplex* y con el fin de sincronizar la comunicación, el transmisor incorpora un bit de sincronismo.

El término asincrónico implica que el transmisor envía datos utilizando su propia fuente de reloj, en vez de compartir la fuente con el receptor. Este tipo de transmisión provee gran autonomía y utiliza pocas líneas de conexión, con la desventaja que el receptor debe conocer la velocidad de transmisión del dato y cada dispositivo debe trabajar a una frecuencia de reloj similar.

Con la finalidad de sincronizar la transmisión, el protocolo de comunicación serie más utilizado se basa, como indica la Figura 4.1, en un bit de inicio, 8 bits de datos, 1 bit de fin y 1 bit de paridad. Estos dos últimos no forman parte del dato, sino que proveen al receptor una indicación de que se está enviando información. Luego del bit de inicio, el transmisor envía en primer lugar el bit menos significativo (LSB), y por último el bit más significativo (MSB).

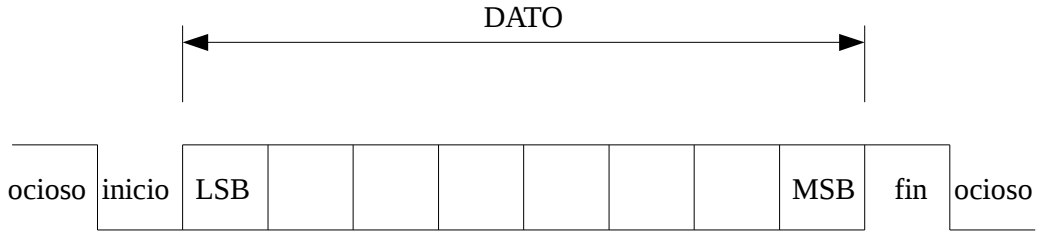


Figura 4.1: Protocolo de comunicación serie.

4.2. Diseño

El objetivo del diseño es obtener un esquema de comunicación serie configurable que pueda ser incorporado en un sistema actual, razón por la cual la interfaz con el maestro se realiza a través del protocolo AMBA APB. Además, se busca agregar versatilidad al sistema, mediante la incorporación de registros de control que permiten determinar características como frecuencia del reloj interno, utilización o no

de una FIFO (por sus siglas en inglés, *First Input First Output*), o cola circular, en el receptor y emisor, verificación de integridad de señal a través del bit de paridad, etc. A continuación se resumen las especificaciones del diseño.

- Transmisión de 8 bits de datos.
- Bit de paridad y tipo, opcional.
- FIFO programable de 16 bytes (8 bits de ancho de palabra).
- Velocidad de transmisión programable.
- Interfaz AMBA APB.

En la Figura 4.2 se muestran los principales bloques que componen el Receptor-Transmisor Asíncrono Universal diseñado: generador de baudios, registros de estado y control, receptor y transmisor, verificación y generación de paridad. Tanto el emisor como el receptor cuentan con la opción de activar o desactivar una FIFO de 16 bytes (16 posiciones de memoria con un ancho de palabra de 8 bits). Ésta permite reducir la pérdida de datos debido a la diferencia de velocidad entre la comunicación serie y el maestro que controla el UART. El emisor y receptor también incorporan bloques, de uso opcional, que permiten detectar errores de transmisión mediante bits de paridad. Tanto las FIFOs, como los registros de transmisión/recepción y los bloques encargados de manejar la paridad, están conectados a los registros de control y estado, que son los que permiten la configuración de los distintos bloques y la detección de condiciones de error, respectivamente.

Como puede observarse en la Figura 4.2, el diseño cuenta con una interfaz AMBA APB para la comunicación entre la UART y el dispositivo maestro que controla sus operaciones. Por otro lado, cuenta con entradas y salidas de señales que intervienen en la comunicación serie entre el transmisor y el receptor. Estos puertos se describen en la Tabla 4.1.

A través de la interfaz APB, un maestro del bus está habilitado para realizar las siguientes operaciones:

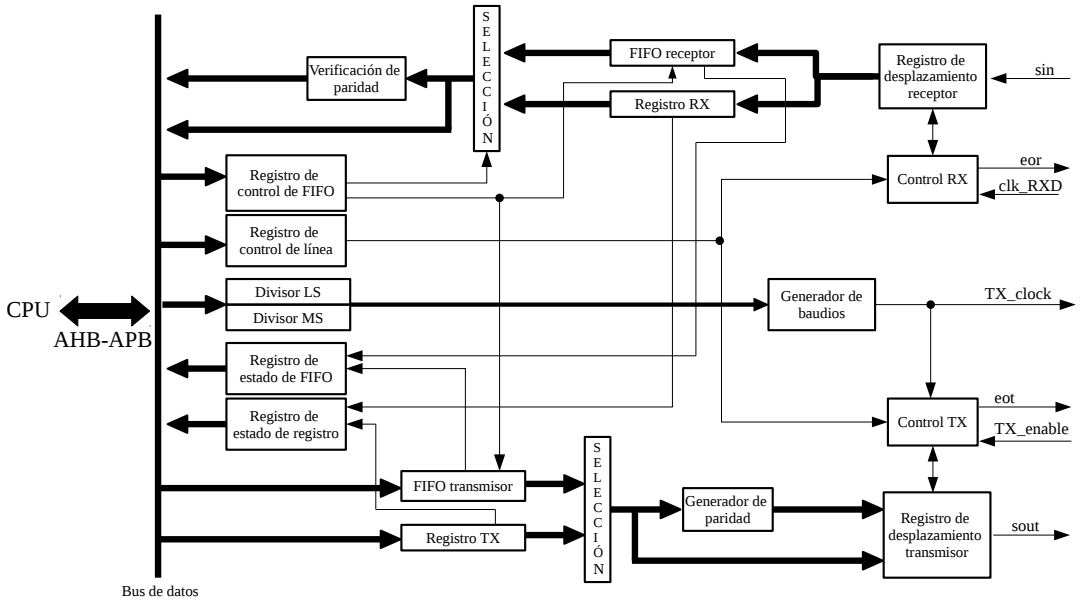


FIGURA 4.2: Diagrama en bloques de la UART diseñada.

- Escribir el dato que se desea transmitir.
- Leer el dato recibido.
- Establecer la configuración deseada.
- Consultar por el estado de la transmisión.

Dichas operaciones son llevadas a cabo seleccionando los bits de dirección (*paddr(0)*, *paddr(1)* y *paddr(2)*), las señales de control del bus APB (*psel*, *penable* y *pwrite*) y la señal de habilitación de la FIFO (*en_fifo*), tal como se muestra en la Tabla 4.2.

A continuación se describen cada uno de los bloques que conforman el sistema.

4.2.1. Generador de baudios

La UART incluye un generador de baudios programable, que permite dividir la frecuencia del reloj de referencia de entrada en factores de 1 a 65536, produciendo

Tabla 4.1: Descripción de los puertos de entrada y salida.

Puerto	Tipo	Tamaño [bits]	Descripción
clk	entrada	1	Reloj APB.
pwdata	entrada	32	Dato escrito por APB.
prdata	salida	32	Dato leído por APB.
paddr	entrada	32	Dirección APB.
presetn	entrada	1	Reset APB, activo en nivel lógico bajo.
psel	entrada	1	Selección de esclavo APB.
penable	entrada	1	Habilitación APB.
pwrite	entrada	1	Control de escritura APB.
pready	salida	1	Esclavo disponible.
pslverr	salida	1	Error en una transferencia APB.
pprot	entrada	3	Tipo de transacción APB.
pstrb	entrada	4	Bits de datos válido.
clk_RXD	entrada	1	Reloj del receptor. Opera a una velocidad de 16 veces los baudios deseados.
sout	salida	1	Salida de datos en forma serie.
sin	entrada	1	Entrada de datos en forma serie.
eot	salida	1	Activa en nivel lógico alto. Indica que el transmisor finalizó una transferencia.
eor	salida	1	Activa en nivel lógico alto. Indica que el receptor finalizó la recepción de un dato.
TX_enable	entrada	1	Activa en nivel lógico alto. Habilita el transmisor para inicializar una transacción serie.
TX_clock	salida	1	Reloj del transmisor controlado por el generador de baudios programable. Opera a 16 veces los baudios programados.

un reloj cuya frecuencia es 16 veces mayor que la velocidad de transmisión requerida. Este reloj también se puede utilizar para controlar la velocidad del receptor. El módulo de comunicación serie es controlado por la CPU a la velocidad de la frecuencia del bus APB. El bloque generador de baudios posee como entrada dicho reloj y el divisor, que es un valor de 16 bits. Este divisor se establece en el registro de control y está formado por dos registros independientes de 8 bits. El reloj de salida de este bloque es utilizado por el transmisor para generar la cadena de caracteres antes de ser enviada en forma serie. El reloj del transmisor se reinicia

Tabla 4.2: Configuración, control y accesos.

pwrite	en_fifo	paddr(2)	paddr(1)	paddr(0)	Acción
1	0	0	0	0	FIFO del transmisor (sólo escritura)
1	1	0	0	0	Registro transmisor (sólo escritura)
1	X	0	0	1	Registro control de línea (sólo escritura)
1	X	0	1	0	Registro control de FIFO (sólo escritura)
1	X	0	1	1	Byte menos significativo del divisor de baudios (sólo escritura)
1	X	1	0	0	Byte más significativo del divisor de baudios (sólo escritura)
0	0	0	0	1	Registro de estado de FIFO del transmisor y receptor (sólo lectura)
0	1	0	0	1	Registro de estado del registro transmisor y receptor (sólo lectura)
0	0	0	0	0	Leer contenido de la FIFO rx (sólo lectura)
0	1	0	0	0	Leer contenido del registro rx (sólo lectura)

en forma asincrónica cada vez que se realiza un reset del sistema.

El valor del divisor de 16 bits se calcula, de acuerdo a la velocidad de transmisión deseada, como:

$$divisor = \frac{f_{clk}}{16 * baudios}. \tag{4.1}$$

En la Tabla 4.3 se observa el valor del divisor para distintas velocidades de transmisión. El divisor debe ser cargado cuando se inicia el sistema con el fin de asegurar su correcto funcionamiento.

Tabla 4.3: Valores del divisor.

Baudios	Reloj = 100 MHz
	Divisor (baudios × 16)
110	56818
1200	5208
1800	3472
2000	3125
2400	2604
3600	1736
7200	868
9600	651
19200	326

4.2.2. Registros de control y de estado

El diseño cuenta con registros de control y estado. Sus funciones son: por un lado, configurar el modo de operación del UART y, por otro lado, conocer el motivo de falla en caso de presentarse una condición de error. En la Tabla 4.4 se muestra el contenido del registro de control de las FIFOs con una descripción del estado de los bits. Este registro es de sólo lectura y se selecciona cuando los bits del puerto de entrada paddr son iguales a "010". En la Tabla 4.5 se muestra el contenido de cada bit del registro de control de línea, que se utiliza para habilitar y establecer el tipo de paridad. Por último, en lo que respecta al control de la UART, se muestra en la Tabla 4.6 el contenido de cada bit de los registros del divisor. Para todos estos tipos de registros de control se tiene acceso desde el bus AMBA y son de sólo escritura.

El otro tipo de registro con los que cuenta la UART son los de estado, los cuales permiten conocer la condición en la que se encuentra la UART. El contenido de los mismos se observa en las Tablas 4.7 y 4.8, e indican estados de las banderas de la FIFO y estados de los registros respectivamente. Estos registros se acceden con los mismos bits de dirección y son de sólo lectura. Cuál de ellos se lee queda

determinado por el bit del registro de control de FIFO, que indica si está habilitada la FIFO o el registro.

4.2.3. Transmisor

El transmisor es el bloque encargado de tomar la información enviada por el maestro en forma paralela, y retransmitirla al receptor en forma serie de acuerdo al protocolo de la Figura 4.1. La estructura del transmisor consta de una máquina de estados, cada uno de los cuales corresponde al envío de un bit. La duración de cada estado es de 16 ciclos del reloj de salida del bloque generador de baudios. Este reloj también es una salida que puede ser conectada con el puerto de entrada de reloj

Tabla 4.4: Registro de control de FIFOs de sólo escritura (paddr = 010).

Bit	Descripción
0	Cuando se establece en nivel lógico bajo se habilita la FIFO transmisora y receptora. Por el contrario, se habilita el registro transmisor y receptor.

Tabla 4.5: Registro de control de línea de sólo escritura (paddr = 001).

Bit	Descripción
0	En estado lógico alto se habilita la paridad.
1	Si se establece en 1 la paridad es de tipo impar, por el contrario es par.

Tabla 4.6: Contenido de los registros del divisor.

Bit	Divisor_LSB (paddr = 011)	Divisor_MSB (paddr = 100)
0	divisor(0)	divisor(8)
1	divisor(1)	divisor(9)
2	divisor(2)	divisor(10)
3	divisor(3)	divisor(11)
4	divisor(4)	divisor(12)
5	divisor(5)	divisor(13)
6	divisor(6)	divisor(14)
7	divisor(7)	divisor(15)

clk_RXD de un receptor. Este criterio de diseño se basa en la posibilidad de transmitir y recibir datos a diferentes velocidades en un mismo Receptor-Transmisor Asíncrono Universal.

En la Figura 4.3 se muestra el diagrama en bloques de la máquina de estados implementada en el transmisor. Durante el estado IDLE, el transmisor se encuentra ocioso y el nivel lógico de la salida serie sout se mantiene alto. La transmisión se inicia con la orden de las señales de entrada *TX_enable* y *dato_leido* o *registro_vacio* que indican si hay un dato en la FIFO transmisora o en el registro transmisor respectivamente. Los estados siguientes corresponden al envío de ocho bits. Una vez enviado el dato, dependiendo del valor del bit menos significativo del registro de control de línea, correspondiente a la señal *hab_paridad*, se añade o no un bit de paridad. Para indicar el fin de la transmisión, se envía un nivel lógico alto durante la duración de un bit por el puerto de salida serie (bit de stop).

Además de la habilitación de paridad, otro aspecto que puede ser configurado en el transmisor es la cantidad de caracteres que pueden ser almacenados antes

Tabla 4.7: Registro de estado de las FIFOs del receptor y del transmisor de sólo lectura (paddr = 001).

Bit	Descripción
0	En 1 indica que la FIFO del transmisor está llena.
1	En 1 indica que la FIFO del receptor está llena.
2	En 1 indica que hubo un error de paridad en la FIFO del receptor.
3	Cuando es 1 la FIFO del receptor está vacía.
4	Cuando es 1 la FIFO del transmisor está vacía.

Tabla 4.8: Registro de estado del transmisor y del receptor de sólo lectura (paddr = 001).

Bit	Descripción
0	En 0 indica que el registro transmisor se encuentra vacío.
1	En 1 indica que el registro receptor se reescribe.
2	En 1 indica que hubo un error de paridad.
3	En 1 indica que el registro receptor está vacío.

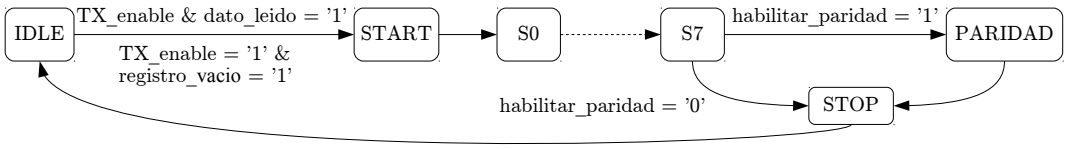


FIGURA 4.3: Máquina de estados del transmisor.

de ser transmitidos. En caso de habilitarse la FIFO, se cuenta con una cola de 16 caracteres que esperan por ser transmitidos. Si la FIFO está llena y se intenta escribirla, la operación es ignorada por la FIFO y se señala esta condición mediante el registro de estado de FIFO, indicando la pérdida de un dato.

Cuando la FIFO no está habilitada, los datos se almacenan en un registro antes de ser enviados en forma serie. Si el registro está ocupado y el maestro intenta escribirlo porque desea enviar ese dato en forma serie, la operación se ignora y se da aviso de esta condición en el bit 0 del registro de estado del registro. El transmisor indica la finalización de transmisión de un caracter mediante el puerto de salida eot (por sus siglas en inglés, *end of transmission*).

La FIFO cuenta con dos relojes, uno para realizar la escritura de la misma, y otro para realizar la lectura. En el caso del bloque transmisor, la escritura de la FIFO se controla por el reloj del bus y se lee a la velocidad de transmisión de la UART.

4.2.4. Receptor

En forma análoga al transmisor, el receptor está realizado con una máquina de estados cuyo esquema se muestra en la Figura 4.4. El reloj usado como base de tiempo en este bloque es la señal que ingresa al sistema por el puerto de entrada clk_RXD. El cambio de estados se produce de manera tal que cada uno de los bits que ingresan en forma serie se muestrean por el receptor en la mitad de su duración, con el fin de determinar su valor en forma precisa. La recepción de un dato comienza cuando se detecta por el puerto de entrada *sin* un bit de comienzo (es decir *sin* = “0”). Dependiendo del valor del bit menos significativo del registro de control de línea, el receptor realiza o no una verificación de paridad. Durante

el último estado de cada recepción, se lee el bit de stop. En caso de que el valor de este bit no sea correcto (nivel bajo), el receptor descarta el caracter recibido.

En caso de habilitarse la FIFO, se cuenta con una cola de 16 caracteres. Si la misma está llena y el receptor intenta escribirla, la operación se ignora y se avisa de esta condición mediante el registro de estado de FIFO. Cuando la FIFO no está habilitada, los datos se almacenan en un registro antes de ser leídos por el maestro. Si el registro está ocupado y se intenta escribirlo, condición que se indica en el bit 1 del registro de estado provisto en la Tabla 4.8, la operación se ignora indicando en el bit de dicho registro la pérdida de un dato.

El receptor indica la finalización de recepción de un caracter mediante la señal de salida eor (por sus siglas en inglés de *end of reception*). Esta señal puede ser conectada al puerto de entrada TX_ENABLE de un transmisor, de manera que las transmisiones ocurran sólo en los momentos en los que el receptor se encuentra en estado ocioso (IDLE).

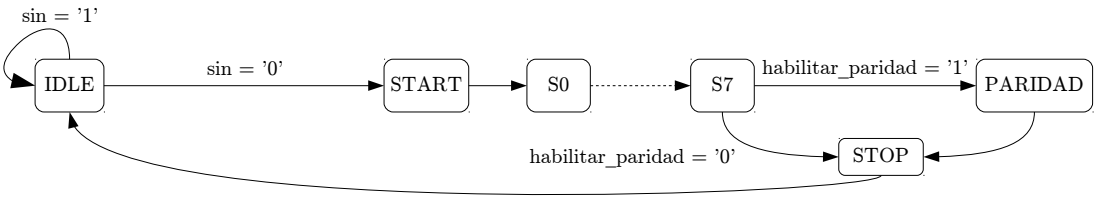


FIGURA 4.4: Máquina de estados del receptor.

4.3. Simulaciones de la UART

En esta sección se presentan las simulaciones de cada uno de los bloques que conforman el sistema, de acuerdo a la Figura 4.2.

4.3.1. Generador de baudios

En la Figura 4.5 se muestra una simulación del bloque encargado de generar el reloj que controla la transferencia de datos en forma serie. El valor del divisor

se inicializa en 326 que en relación a la Tabla 4.3 corresponde a una velocidad de transmisión de 19200 baudios. La señal *clk* es el reloj del sistema, la señal de salida de reloj es *clk_txd*, *rst* la señal de reinicio, *brd* es el divisor, y las restantes son señales internas. La velocidad de transmisión es válida luego de realizar un reset del sistema, si se desean configurar los baudios a valores menores, se debe efectuar un reset. Cada flanco positivo del reloj del sistema incrementa un contador que, al alcanzar el valor almacenado en el registro del divisor *brd*, cambia la señal de salida *clk_txd*.

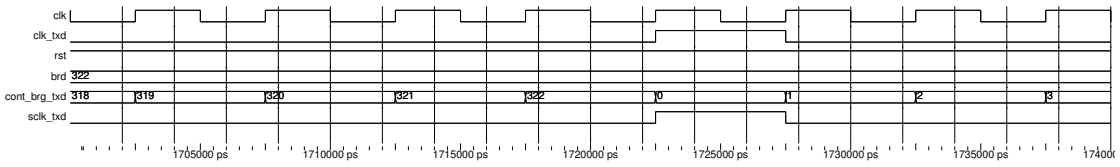


FIGURA 4.5: Generador de baudios.

4.3.2. Transmisor

El bloque encargado de la transmisión, envía un dato siempre que la señal *tx_enable* se encuentre habilitada y se haya leído un dato desde la FIFO, acción indicada por la señal *dato_leido*, o se haya leído un dato desde el registro transmisor indicado en la señal *registro_vacío*. Para la simulación, la señal *tx_enable* se controla por un receptor externo que la coloca en nivel lógico alto cuando finaliza la recepción de un dato.

La señal *dato* es el dato que se desea transmitir. Según el valor de la señal *en_fifo*, corresponde a un dato almacenado en la cola transmisora o en el registro transmisor. En la Figura 4.6a se observa que, en el momento inicial, dicha señal se encuentra en 0, por lo tanto la cola del transmisor se encuentra habilitada. La duración de cada bit transmitido corresponde a 16 períodos de la señal de reloj *clk_txd*. Como la FIFO se encuentra habilitada, se lee un dato desde la misma que en este caso es el 00000001, acción indicada en la señal *dato_leido*. En este momento comienza la transmisión del dato que continúa hasta finalizar, independientemente del valor de la señal *en_fifo*. En el estado START, la salida *sout* es,

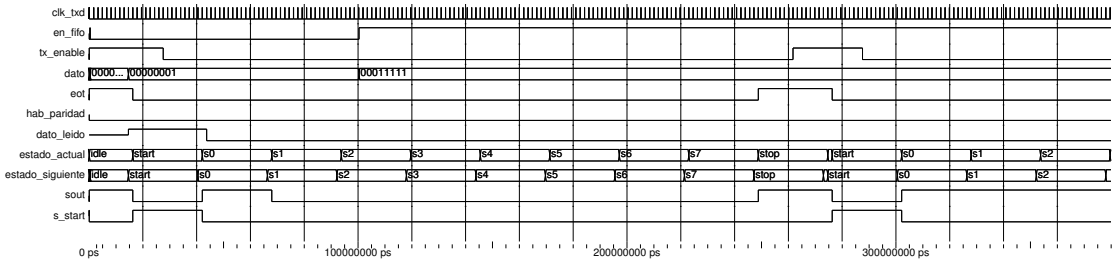
respetando el protocolo de comunicación, un nivel lógico bajo. Luego se transmite el bit menos significativo hasta llegar al estado STOP donde se coloca en nivel lógico alto y finaliza la transmisión, regresando nuevamente al estado IDLE.

En la Figura 4.6b, a diferencia del caso anterior, se transmite un dato desde el registro del transmisor, que en este caso es 00011111. Nuevamente el dato a transmitir se encuentra en la señal *dato*. Al comenzar la transmisión se muestra que la señal *registro_vacio* pasa a estar en 1, es decir, que el registro queda vacío y disponible para almacenar un nuevo dato desde el bus.

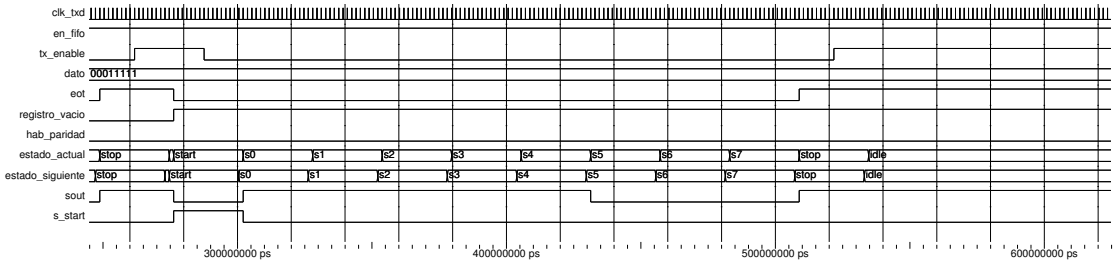
Para ambos casos, la señal *s_start* es una salida hacia un bloque de mayor jerarquía, que se utiliza para indicar que comenzó una transferencia.

Si se encuentra habilitada la verificación de paridad, el transmisor genera un bit adicional cuyo valor corresponde a la paridad par o impar, según esté establecido en el registro control de línea. En el caso de la simulación de la Figura 4.6c se envía el número 00000011 con la paridad establecida par, por lo que el transmisor genera la paridad par y en este caso particular envía un 0 por *sout*.

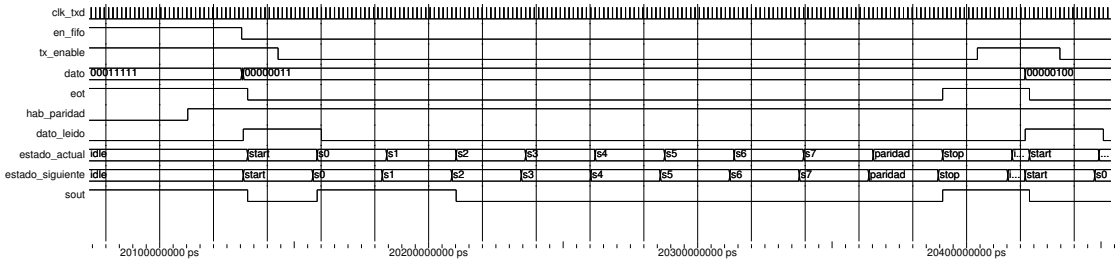
En la Figura 4.7 se muestran las simulaciones correspondientes a la FIFO del transmisor. La misma cuenta con dos relojes, la señal *clkin* corresponde al reloj de escritura (que en este caso es idéntico al del sistema) y la señal *clkout* al reloj de lectura (que en este caso lee a la velocidad del reloj del transmisor). La escritura en la FIFO del transmisor se controla por el bus APB. Siempre que se intente escribir y se encuentre llena, no se realizará ninguna acción. La lectura de la FIFO se habilita siempre que la señal *TX_enable* y la señal *seot* se encuentren en 1, indicando que se finalizó la transferencia de un dato. En el caso que se muestra en la Figura 4.7a la FIFO se escribe hasta que se llena, momento en que el nuevo estado es LLENA indicado en la señal *estado_actual_w*, y la señal *full_flag_n* se establece en nivel lógico bajo. En la Figura 4.7b se muestra el instante en que se lee la FIFO del transmisor con un flanco del reloj *clkout* y deja de estar llena indicando dicho comportamiento en la bandera de FIFO llena, *full_flag_n*, colocándose en 1. En la Figura 4.7c se realiza la lectura de la FIFO hasta que se vacía, estado que se indica como VACIA en la señal *estado_actual_r*, y estableciendo la bandera *empty_flag_n*



(A) Transmisión de un dato almacenado en la cola del transmisor.



(B) Transmisión de un dato almacenado en el registro del transmisor.



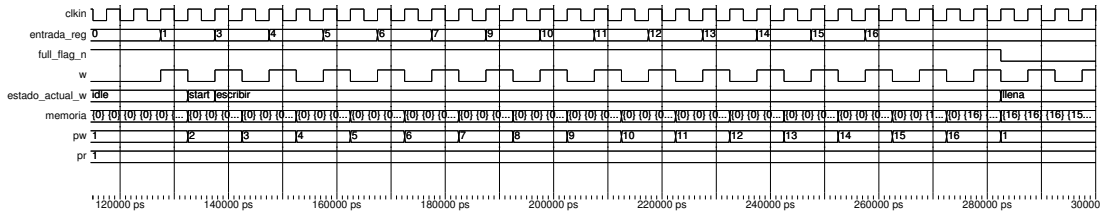
(C) Transmisión de un dato almacenado en el registro del transmisor con la paridad habilitada.

FIGURA 4.6: Transmisión de un dato del registro o de la cola del transmisor.

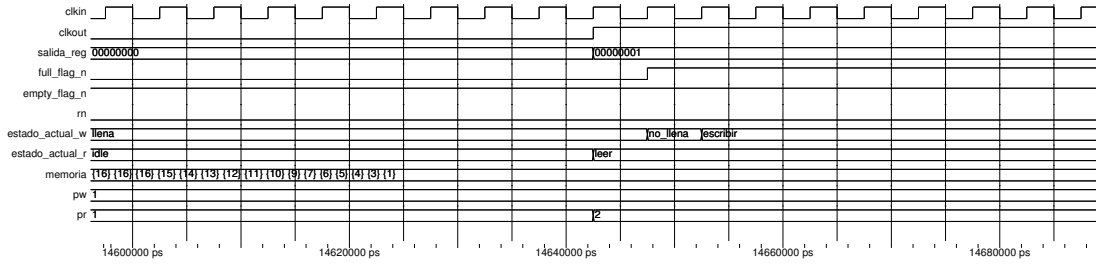
en 0. Las banderas que advierten que la cola se encuentra vacía son sincrónicas con el reloj de lectura *clkout*, mientras que las que indican que la FIFO se llena son sincrónicas con el reloj de escritura *clkin*, en el caso del transmisor.

En la Figura 4.8 se muestra la generación de paridad en el transmisor. Este es un bloque asincrónico que genera un bit de paridad siempre que la señal *hab_paridad* se encuentre en 1 y se lea un dato desde la FIFO o del registro del transmisor. Si se calcula paridad par, el bit de paridad se obtiene realizando la función XOR de los bits de la señal *dato_in*. En cambio, el bit de paridad impar se obtiene de la misma función negada. El resultado se encuentra en la señal *bit_paridad*.

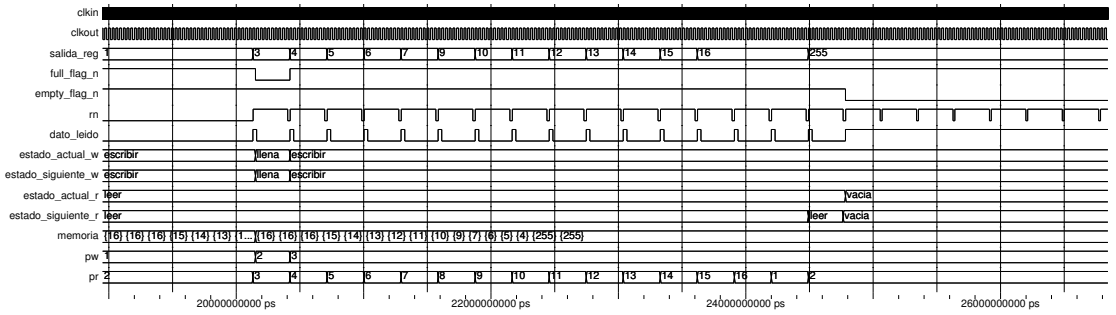
El registro del transmisor se vacía cuando el transmisor lee el dato desde el registro, en el estado START. El reloj de lectura de la FIFO *clkout* del transmisor se genera



(A) Escritura en la FIFO del transmisor.



(B) Lectura en la FIFO del transmisor.



(C) Lectura de la FIFO del transmisor hasta vaciarla.

FIGURA 4.7: Escrituras y lecturas sobre la FIFO del transmisor.

con *clk_txd*. La escritura del registro de control, es dirigida por el reloj del sistema.

Como se observa en la simulación de la Figura 4.9, la escritura del registro transmisor se controla desde el bus APB, y es posible escribirlo siempre y cuando se encuentre vacío, lo cual se indica por la señal *s_registroTX_vacio* cuando la misma está en 1.

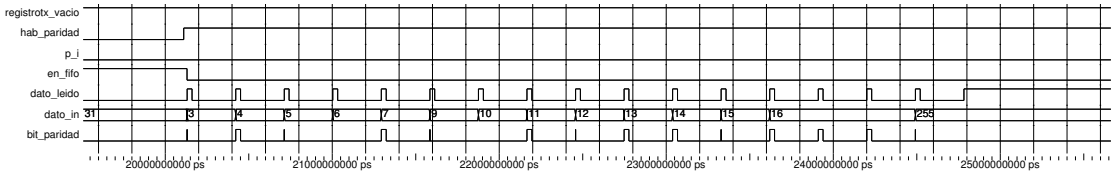


FIGURA 4.8: Generador de paridad.

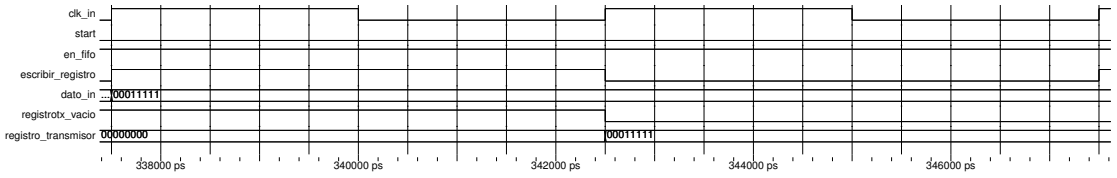
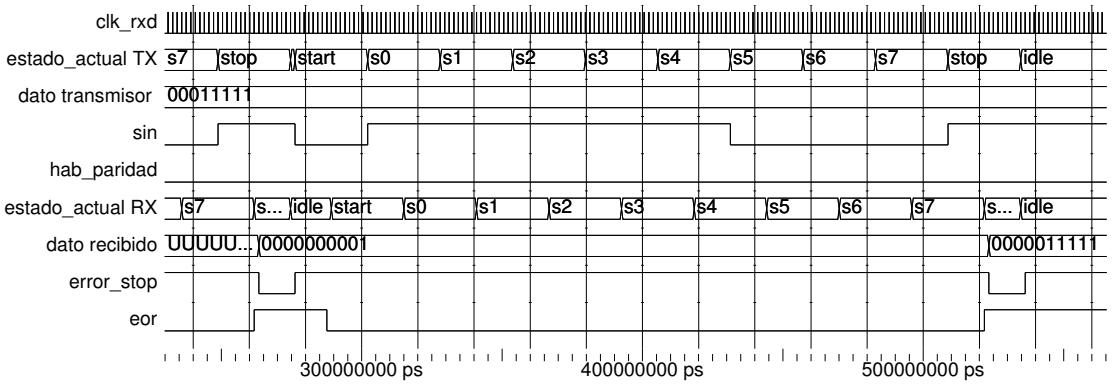


FIGURA 4.9: Escritura en el registro transmisor.

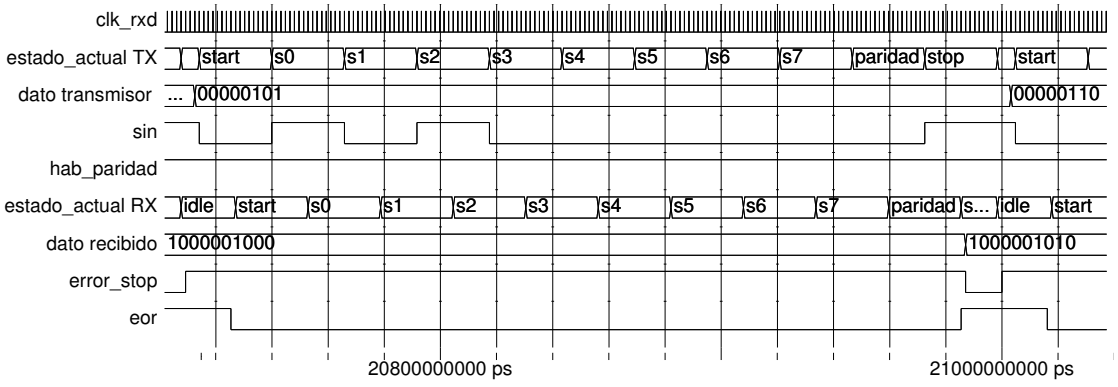
4.3.3. Receptor

Las simulaciones del bloque receptor se muestran en le Figura 4.10. La recepción de un dato serie sin paridad se muestra en la Figura 4.10a. La recepción se inicia siempre que se detecte por el puerto *sin* un bit de inicio. Este bloque controla las transiciones de las señales por el reloj de la transmisión clk_RXD. Su objetivo principal es muestrear el dato recibido a la mitad del bit con el fin de asegurar un dato válido. La recepción finaliza en el estado STOP de la señal *estado_actualRX*, indicando si hubo o no error en la transmisión verificando el bit de fin recibido, requerido por el protocolo. En caso de error, se indica mediante la señal *error_stop*, y el dato no se escribe ni en la FIFO ni en el registro transmisor. La señal *error_stop* se encuentra en 1 y si no ocurre un error en la transmisión, se pone en 0 durante el estado STOP. Como en el caso del transmisor, si la paridad se encuentra habilitada, se agrega un estado adicional denominado PARIDAD en el que se recibe el bit de paridad correspondiente para luego ser verificada, como se muestra en la simulación de la Figura 4.10b. De esta manera, como se observa en ambas simulaciones, la palabra transmitida es siempre de 10 bits, compuesta por los 8 bits de datos, el bit de paridad, y el bit que indica si la paridad está o no habilitada. El bit de paridad se ignora si la misma no se encuentra habilitada.

En la Figura 4.11 se muestra el bloque encargado de verificar paridad. Toma los 8 bits de datos y el bit de paridad en la señal *dato_in*, realiza la función XOR del dato de entrada y lo compara con el bit de paridad. Si son distintos se indica la condición de error estableciendo la señal *error_paridad* en 1 si se verifica paridad par, o en 0 si lo que se verifica es paridad impar. Este tipo de error se refleja en las señales *error_paridad_f* o *error_paridad_r*, de acuerdo al estado de la señal *en_fifo*, que indica si la FIFO o el registro están habilitados. En el caso de la simulación, se



(A) Recepción de un dato serie cuando el bit de paridad se encuentra deshabilitado.



(B) Recepción de un dato serie cuando el bit de paridad se encuentra habilitado.

FIGURA 4.10: Recepción de un dato.

encuentra habilitada la FIFO y se verifica paridad par. Por tal motivo en el caso que el dato de entrada es 000000100 donde el bit más significativo es 0, se genera un error de paridad par.

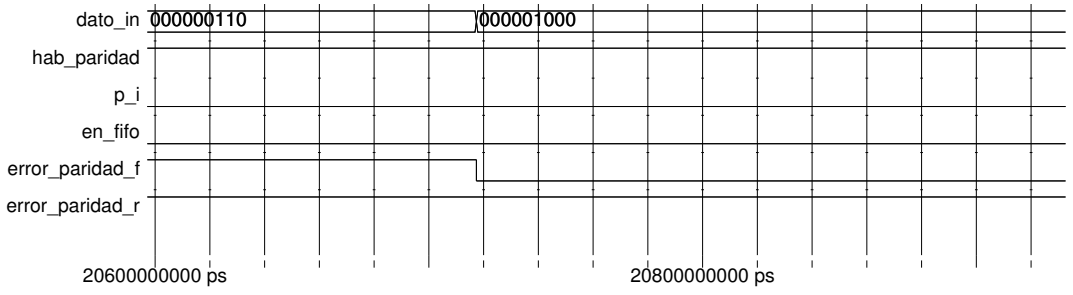


FIGURA 4.11: Verificación de paridad en el receptor.

En el caso de la FIFO del receptor, la escritura se controla por el reloj que dirige la transmisión serie, mientras que la lectura se encuentra sincronizada por el reloj del sistema, que es de mayor velocidad. En la simulación de la Figura 4.12 se muestra

la primer escritura en la FIFO seguida inmediatamente de una lectura. El dato de entrada a la memoria de la FIFO del receptor se encuentra en *entrada_reg* y el de salida en *salida_reg*. Se puede observar como deja de estar vacía (reflejado en la bandera *empty_flag_n*) y se vuelve a vaciar al leer el dato escrito. Cada vez que la memoria de la FIFO se llena o se vacía los punteros que controlan las posiciones de memoria que se leen o se escriben mantienen su valor hasta que se salga de esa condición. Estos punteros se notan en la simulación como *pw* (puntero de escritura) y *pr* (lectura).

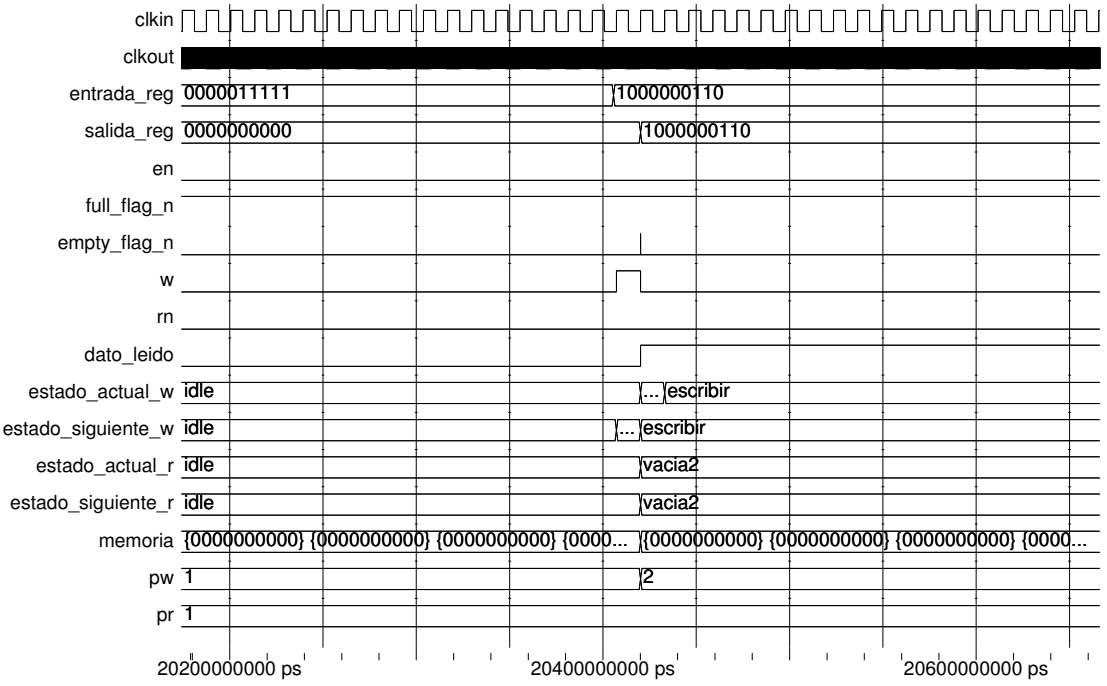


FIGURA 4.12: Escritura y lectura en la FIFO del receptor.

4.3.4. Registros de control y de estado

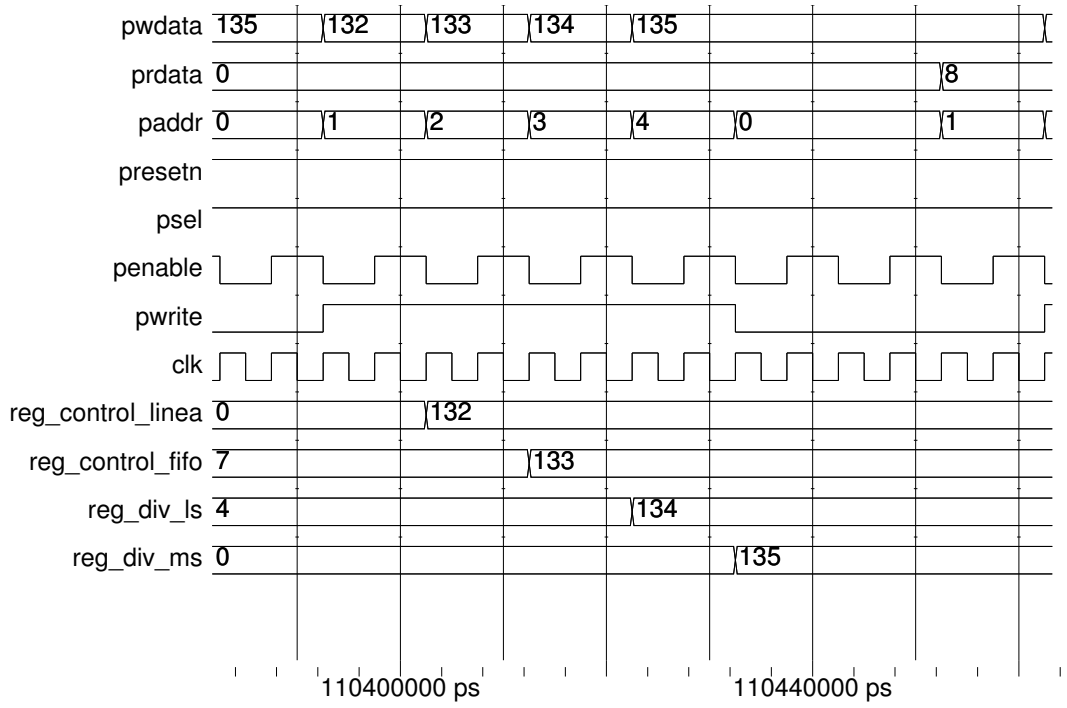
Todas las señales que indican una condición de error o un aviso hacia el sistema se almacenan en estos registros, sincronizados con el reloj del sistema. En la Figura 4.13 se muestran simulaciones de la escritura y lectura de estos registros. La CPU los accede desde el bus APB para configurar la transmisión serie o para conocer su estado. En el caso de la Figura 4.13a se escriben los registros de control. Las señales *pwdata*, *prdata*, *paddr*, *presetn*, *psel*, *penable* y *pwrite* corresponden a

las señales del bus APB. La dirección (*paddr*), el dato (*pwdata*) y la señal *pwrite* deben ser colocadas por el maestro cuando se inicia la transmisión y no deben cambiarse hasta terminarla. Este proceso tiene una duración de al menos dos ciclos de reloj. En el segundo ciclo, es decir en la fase de acceso, el maestro coloca en 1 la señal *penable*, indicando el fin de una transmisión. Como se muestra en la simulación, se escriben los valores 132, 133, 134 y 135 en el registro de control de línea, en el registro de control de la FIFO, en el registro que contiene los 8 bytes menos significativos del registro del divisor y en el que contiene los 8 bytes más significativos del registro del divisor, respectivamente. En la Figura 4.13b se observa la lectura del número 9 desde el registro de estado del transmisor y del receptor, dado que se encuentra habilitado por la señal *en_fifo*.

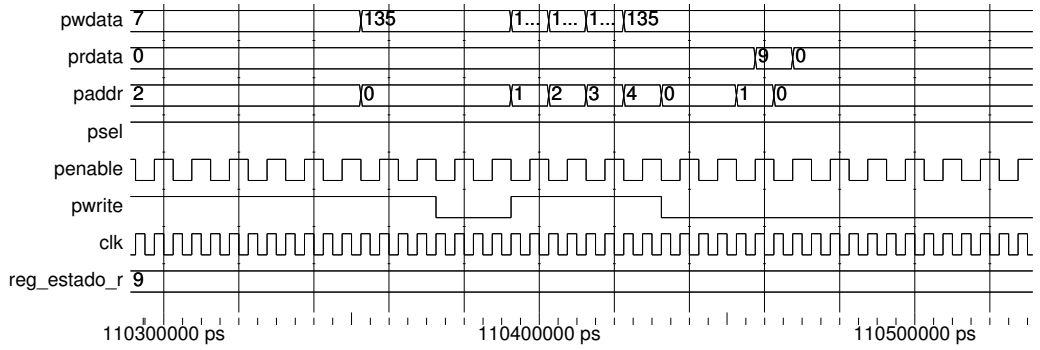
4.3.5. Transferencias del bus APB

En la Figura 4.14 se muestran simulaciones de transferencias de escritura y de lectura simples entre un maestro APB y la UART diseñada. En la Figura 4.14a se puede observar la escritura desde el maestro APB hacia los registros del divisor de la UART. La transferencia APB comienza cuando se detecta un nivel lógico alto en la señal *psel*, al mismo tiempo que se establece la dirección en la señal *paddr*, el dato en *pwdata*, y la señal *pwrite* en 1 debido a que es una transferencia de escritura. En este caso particular, se escribe en las direcciones 3 y 4 los valores 134 y 135, respectivamente. Como se puede observar, el esclavo completa la transferencia en el tercer flanco de reloj, cuando reconoce la fase de acceso y la señal *penable* se encuentra en 1. Siempre que finaliza la transferencia, la señal *penable* debe volver a 0. Este proceso corresponde a un tipo de transferencia APB simple, en la cual el esclavo no necesita insertar estados de espera mediante la señal *pready* para efectuar la operación.

En la Figura 4.14b se muestra una transferencia de lectura simple APB. Cuando la dirección del bus *paddr* es 1 y la señal *pwrite* es 0, se selecciona la lectura del registro de estado del receptor y del transmisor . En el caso de la lectura, el dato es válido en el bus en el tercer flanco positivo del reloj luego de iniciada la



(A) Escritura de los registros de control.



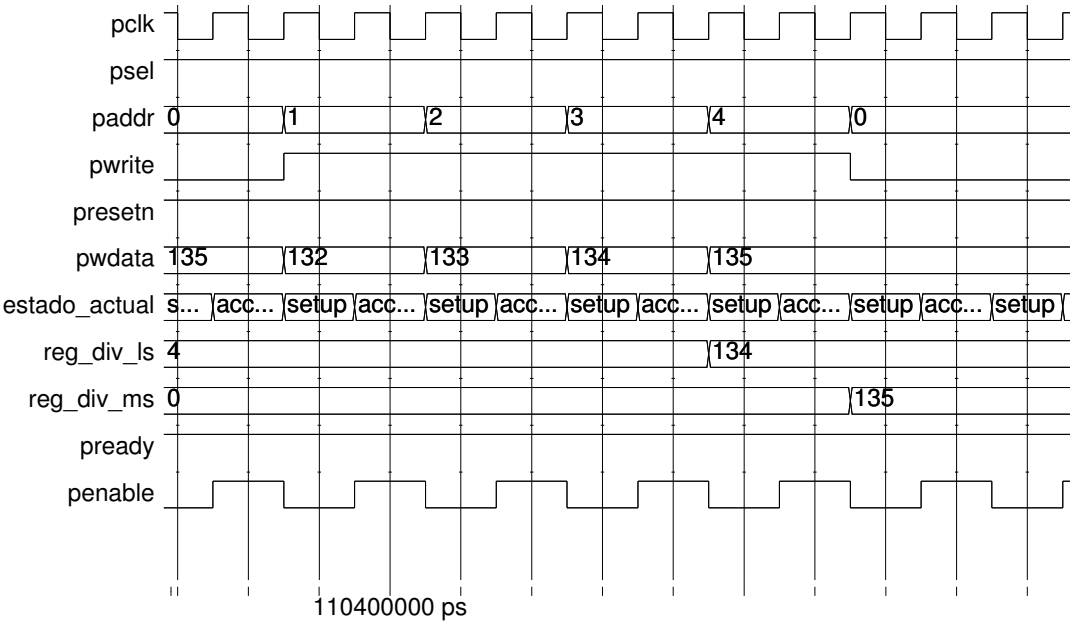
(B) Lectura del registro de estado.

FIGURA 4.13: Registros de control y de estado.

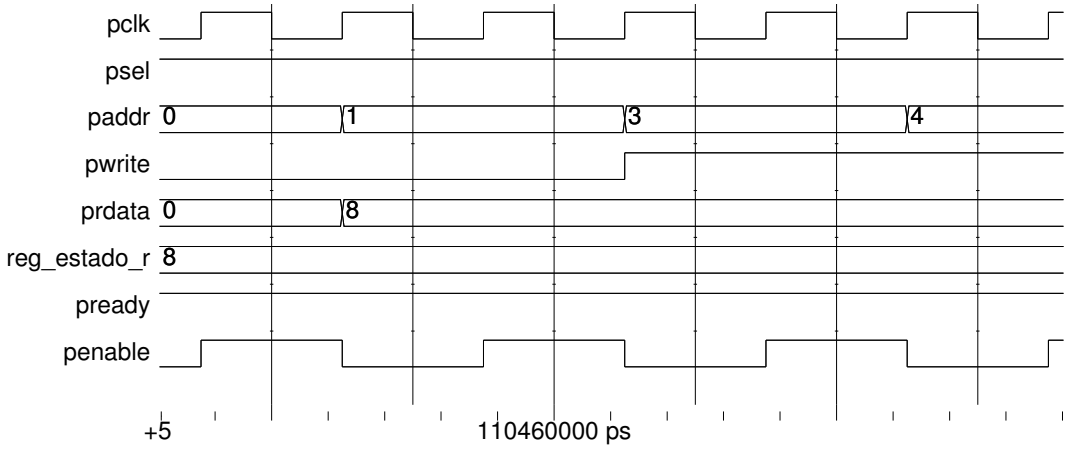
transferencia, donde la señal *penable* es 1. El dato debe ser colocado por el esclavo antes de este flanco de reloj.

4.4. Implementación física de la UART

Luego del diseño, se realizó la síntesis lógica con la herramienta Design Vision de Synopsys, la cual tiene como entrada la descripción en VHDL del sistema y la



(A) Transferencia de escritura.



(B) Transferencia de lectura.

FIGURA 4.14: Transferencias con el bus AMBA APB.

traduce a compuertas de la tecnología. Con el fin que la herramienta realice una buena optimización, la descripción en VHDL se debe realizar de forma tal que pueda identificar las estructuras del código, las pueda mapear correctamente con las características de la tecnología y pueda realizar la optimización.

El diseño se realizó utilizando librerías de la tecnología de Tower Jazz de 180 nm. Para la síntesis se obtuvieron resultados que cumplen con las restricciones con un reloj del sistema de 100 MHz y una velocidad de transmisión máxima de 19200

baudios.

Una vez realizada la optimización, se cumplen con las restricciones de temporizado establecidas y se observa que existe una violación en las restricciones de área, debido a que se le indicó a la herramienta que realice una optimización con área máxima igual a cero, lo cual resulta imposible. El período de reloj se determina desde el sintetizador, considerando el mínimo período para el cual no existan violaciones de las restricciones.

En las Figuras 4.15 y 4.16 se muestran simulaciones del resultado de la síntesis lógica realizada. Estas simulaciones se realizaron en Modelsim, con archivos de entrada del resultado de la síntesis en Verilog y archivos con las compuertas propias de la librería.

En la Figura 4.15 la señal *clk* es el reloj del bus, la señal *clk_RXD* es igual a este reloj mientras el *reset* se encuentre activo, luego depende del valor cargado en el divisor del generador de baudios. En este momento no se realiza ninguna transmisión serie por lo que las señales involucradas en esta transmisión, *sin*, *TX_enable*, *sout*, *eot* y *eor* se encuentran en el estado que indica que no hay transmisión. La señal *hab_escritura* en 1 determina que se realizará una escritura. Como la señal *reg_control_fifo* en 0 indica que se encuentran habilitadas tanto la FIFO receptora como transmisora, se escribe la FIFO transmisora porque es el bus APB el que está escribiendo en la interfaz. Luego de 16 escrituras la FIFO del transmisor se llena y cambia el valor de sus banderas *full_flag_n_t* y *empty_flag_n_t*.

En la Figura 4.16 se transmite un dato almacenado en el registro transmisor. En este caso, como se indica en la señal *dato*, se transmite y se recibe en el registro receptor (*en_fifo* en 1) el 00000011. La transmisión finaliza en el momento indicado por la señal *eot* en 1. El dato es recibido por el receptor cuando la señal *eor* se establece en 1 y se muestra el dato recibido en la señal *dato_p*. En esta señal se muestra que se recibe el 00000011 con un bit adicional en 1 que indica que la paridad se encuentra habilitada.

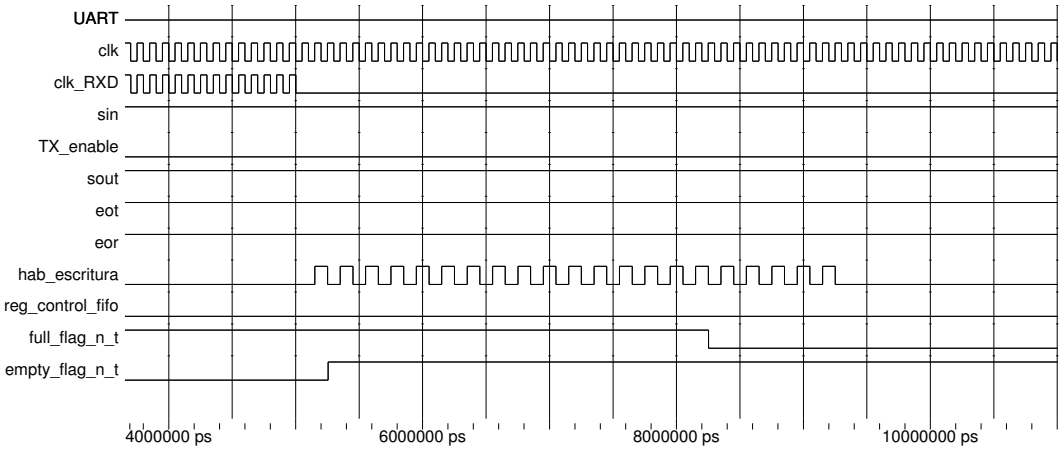


FIGURA 4.15: Simulación de escritura en la FIFO sintetizada del transmisor.

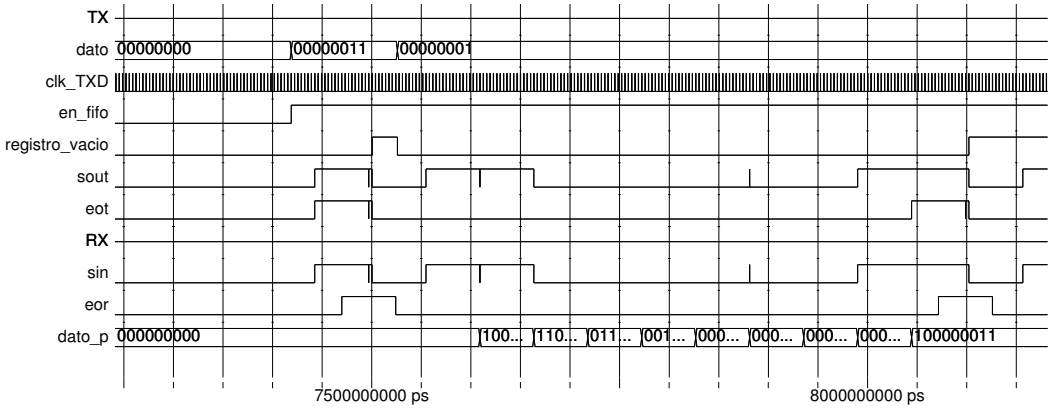


FIGURA 4.16: Simulación de transmisión y recepción de un dato serie luego de la síntesis lógica.

En la Figura 4.17 se muestran las máscaras del circuito con el generador de números aleatorios, la interfaz de comunicación serie, la interfaz AMBA APB y los pads de entrada y salida. Con respecto a estos últimos, se utilizaron 6 de las opciones ofrecidas por la tecnología. Los que corresponden a la alimentación son pv0a (p_gnd_IO), pvda(p_Vdd_IO), pvdi(p_Vdd), pvdi(p_Vdd_A), pv0i(p_gnd_A). Los dos primeros son para la alimentación del anillo de pads. Luego, tanto la parte analógica como digital comparten ground y tienen Vdd independiente, donde el pad p_Vdd_A es para el Vdd de la parte analógica y el otro p_Vdd para la alimentación de la parte digital. Los demás son de entrada y salida digitales, pdid1ecdv y pdt4chv respectivamente y las entradas y salidas analógicas pc3d00.

Al realizar la síntesis física, se incluye el anillo de pads con los pads de entrada

y salida digitales y los analógicos. A pesar que la parte digital no utilice éstos últimos, se evita colocarlos manualmente en Virtuoso. Una vez que se instancia el diseño digital y el anillo de pads en Virtuoso, se realiza el conexionado entre las entradas y salidas del generador y los pads analógicos.

La síntesis física de la interfaz AMBA APB y la UART se realiza con la herramienta de implementación digital de Cadence, Encounter. Como entrada, dicha herramienta necesita el archivo Verilog que contiene el netlist de salida del Design Vision. Además, necesita de la definición de un módulo de mayor jerarquía en el que se instancia el bloque digital diseñado y los pads de entrada y salida.

Luego de tener la síntesis física del diseño digital con la instanciación de los pads, se modifica el *layout* con la herramienta Virtuoso de Cadence en donde se instancia la parte del *layout* analógico. El área total ocupada por el circuito completo es de 759 μm $\times$ 406 μm. El generador de números aleatorios ocupa un área de 296 μm $\times$ 132 μm y la parte digital un área de 426 μm $\times$ 406 μm.

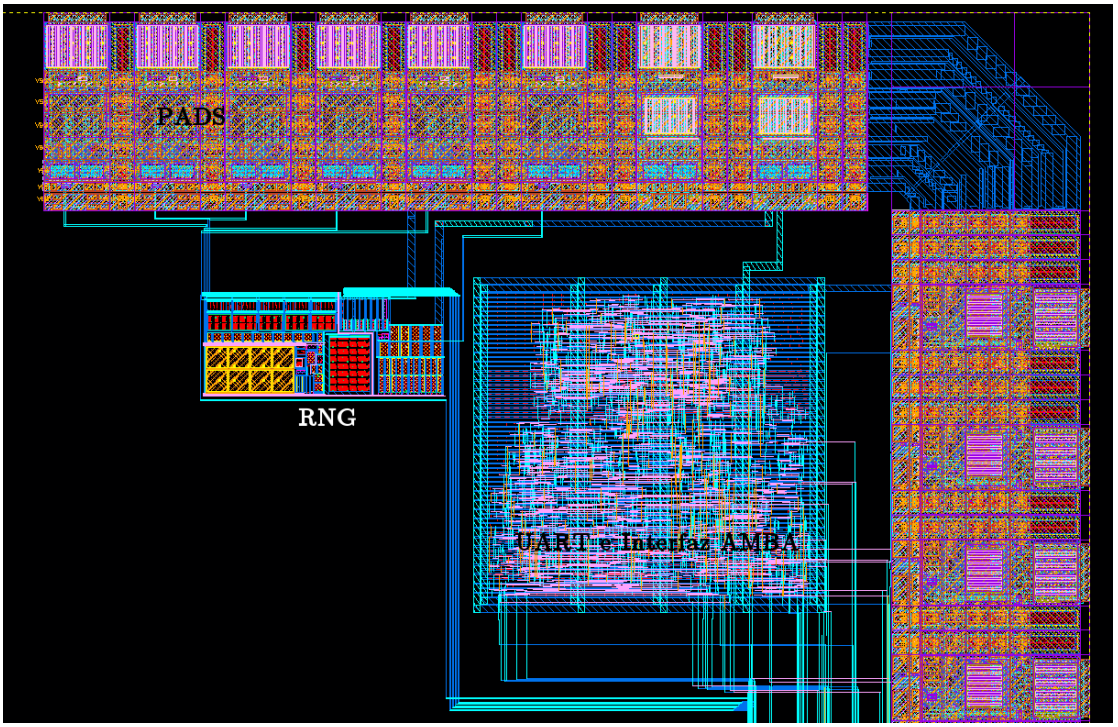


FIGURA 4.17: Máscaras del generador de números aleatorios y UART con interfaz AMBA APB.

Capítulo 5

Aspectos de la validación experimental

Al momento de presentar esta tesis, el circuito integrado se encontraba en etapa de fabricación en el proceso de 180 nm de TowerJazz, al que la Universidad Nacional del Sur accede a través de MOSIS.

Al recibir el circuito integrado, se realizarán una serie de mediciones con el objetivo de validar los resultados obtenidos por simulación. Las mediciones se realizarán en el laboratorio de microelectrónica del Instituto de Investigaciones en Ingeniería Eléctrica Alfredo Desages (IIIE), que cuenta con el equipamiento de ultra-bajo ruido y alta precisión necesario para estas aplicaciones.

El procedimiento de validación experimental que se propone es el siguiente:

5.1. Validación del generador de números aleatorios

5.1.1. Amplificador de ruido

Luego de polarizar el circuito, y antes de conectar el amplificador al VCO, se medirá la señal de salida de las últimas tres etapas del amplificador. Se evaluará la variación de esta señal de salida en función de la polarización (variando el valor de Rm) y en función de la temperatura. Se realizará el análisis estadístico de las señales muestreadas para asegurar que su distribución es la esperada.

5.1.2. Oscilador controlador por tensión

Para un punto de polarización del amplificador, y una temperatura fija, seleccionados de manera que permitan evaluar el desempeño del VCO correctamente, se conectará la salida del amplificador a la entrada del VCO y se medirá la señal de reloj obtenida. En este caso no se puede desconectar el arreglo de osciladores de anillo, que funcionarán como carga del VCO, al igual que la capacidad parásita del pad de salida. Se realizará el análisis estadístico de la variación de frecuencia de salida del VCO, para verificar que tenga las mismas propiedades que la salida del amplificador. Luego se evaluará su desempeño frente a variaciones de la corriente de polarización y de la temperatura, y se identificarán los casos extremos en los que el circuito deja de funcionar correctamente.

5.1.3. Osciladores de anillo

Se realizará la adquisición de una serie de bits de cada uno de los osciladores, para verificar su correcto funcionamiento. La cantidad de bits a adquirir será determinada por las herramientas de análisis estadístico seleccionadas, con el fin de verificar que ningún oscilador tenga una tendencia marcada a generar más

1 que 0, o viceversa. Finalmente, formando palabras de 8 bits con la salida de cada oscilador, se validará la aleatoriedad de los números generados, mediante el estándar de evaluación propuesto por el NIST.

5.2. Validación de la UART

La validación de la UART se llevará a cabo utilizando un FPGA Artix 7 XC7A200T y su kit de evaluación AC701. La descripción de los test de validación se realizaran en VHDL y se utilizará el software Vivado Design Suite: Design Edition de Xilinx.

5.2.1. Transmisión serie

Los puertos de entrada y salida serie del FPGA se conectarán a los de la UART diseñada. El puerto de reloj de la UART se conectará al reloj del FPGA y el reset, al reset de la placa de desarrollo. Los demás puertos de entrada y salida de la UART se conectarán a los pines de expansión del FPGA.

Estableciendo la velocidad serie que corresponda, se enviarán datos a la UART desde el puerto serie RS-232 de una PC y se realizará un eco para comprobar que el dato que se envía es el que se recibe.

5.2.2. Interfaz APB, registros de configuración y registros de estado

La descripción en VHDL de un maestro APB se sintetizará en un FPGA. El maestro APB contará con las siguientes señales:

- Una señal que habilita la transferencia que es posible controlar externamente.
- Una señal para establecer la dirección la cual se controlará por medio de una entrada externa del FPGA.

- Una señal para controlar si se realiza una escritura o una lectura. De la misma forma se controlará por una señal externa al FPGA.
- Señal del dato a escribir en la UART. Este dato se generará internamente desde el FPGA según la señal de dirección.
- El maestro además, cuenta con las señales que se conectarán directamente con pines de la UART a través de los puertos de expansión de la placa de desarrollo del FPGA. Estas señales son: PRDATA, PSEL, PENABLE, PADDR, PWRITE y PWDATA. Estas señales controlan la transferencia APB.

Por otro lado, en cuanto a la transmisión serie, el pin sout de salida serie se conectará con el pin sin de entrada. El reloj de salida del generador de baudios TX_clock se conectará con el reloj de entrada del receptor CLK_RXD. La señal TX_enable para habilitar la transmisión serie será una entrada de la placa de desarrollo del FPGA y los pines eot y eor se conectarán a los pines de expansión provistos por la placa de desarrollo del FPGA. Estas señales serán controladas internamente para comprobar el estado de la transmisión.

Junto con la descripción en VHDL del maestro APB se realizará un bloque en donde se establezcan las acciones a realizar cada vez que se realice una transferencia a una dirección determinada.

En primer lugar se escribirán los registros de control estableciendo externamente las señales de dirección que correspondan, habilitando la escritura y la transferencia. En condiciones iniciales las FIFOs se encuentran habilitadas, y la paridad deshabilitada. Lo que se modificará será el valor de los registros del divisor de baudios de manera de dejar establecida la velocidad de transferencia. Con la transmisión serie deshabilitada se escribirá la FIFO transmisora indicando la dirección que corresponda. Los datos a escribir estarán preestablecido al momento de hacer la síntesis en el FPGA. Luego de 16 escrituras se realizará una lectura del registro de estado de FIFOs para verificar el estado.

Continuando con la lectura del registro de estados, se habilitará la transmisión serie, y se enviará por dicho puerto los datos de la FIFO transmisora hasta que el registro de estado indique que se encuentra vacía, mientras que la FIFO receptora debería llenarse. El estado de los registros se verá en los leds de la placa de desarrollo del FPGA.

Luego se realizarán lecturas a la FIFO del receptor, seleccionando la dirección que corresponda y se verificará que los datos recibidos son los enviados por el transmisor.

A continuación se desactiva la transmisión serie por medio de la señal TX_enable y se escribe en el registro de control de forma de deshabilitar las FIFOs y habilitar la escritura en el registro transmisor y receptor. Con los pines de transmisión serie conectados para realizar una transferencia hacia la PC, se habilita la transmisión, se realiza un eco y se controla que ambos registros se escribieron correctamente. Por último se validará la habilitación, generación y verificación del bit de paridad. Para esto se escribirá en el registro de control de línea y se escribirá un dato con la paridad par o impar. Desde el transmisor se enviará el dato hacia el receptor y se controlará que la paridad es correcta.

Capítulo 6

Conclusiones

En esta tesis se presentó el desarrollo de un sistema de comunicación dentro de un circuito integrado basado en un bus de comunicación AMBA, compuesto por dos circuitos periféricos. Uno de ellos es un receptor transmisor asíncrono universal, y el otro, un generador de números aleatorios real. La aleatoriedad de este último se obtuvo desde una fuente de ruido proveniente de resistencias. El sistema completo se implementó en una tecnología de Tower Jazz de 180 nm.

En el Capítulo 1 se realizó una introducción a los sistemas en un chip (SoCs), los cuales surgen como consecuencia de los avances en las tecnologías y técnicas de fabricación de circuitos integrados. Los fabricantes de SoCs buscan que sus componentes sean reutilizables, con el fin de reducir costos y tiempos para salir al mercado. Debido a este requerimiento se hizo énfasis en la importancia de realizar un diseño de un sistema de comunicación dentro de un SoC que permita bloques reutilizables. Para alcanzar este objetivo es de suma importancia el desarrollo de interfaces estandarizadas.

Se expusieron los factores que afectan el desempeño del circuito en cuanto aumentan las interconexiones: tiempo de propagación, impacto en la disipación de energía y distribución de potencia e introducción de fuentes de ruido. Por estos efectos, el análisis del rol y comportamiento de las líneas de interconexión resulta trascendente. Luego se especificaron los elementos que definen una arquitectura de

comunicación: en primer lugar la topología del bus, que es la estructura física de interconexión de los dispositivos maestros y esclavos. Se clasifica en bus compartido, bus jerárquico o de tipo anillo. En segundo lugar, los parámetros del protocolo que gestionan la prioridad de acceso a los canales: *static priority*, *time division multiple access*, *lottery*, *token passing* y *round-robin*. Se realizó una breve descripción de los distintos estándares de comunicación dentro de un circuito integrado, entre ellos: AMBA; Wishbone; Avalon; CoreConnect; Virtual Component Interface (VCI); STBus; CoreFrame; Bus PI; Protocolo Open Core; SiliconBackplane uNetwork. Se mostraron ejemplos, y en algunos casos, aplicaciones.

Luego de realizar un reconocimiento de los estándares existentes se concluyó en la elección del bus AMBA por su dominación del mercado mundial y disponibilidad de documentación.

En el Capítulo 2 se realizó el desarrollo de la especificación AMBA: en primer lugar se mostró la evolución y características del estándar para cumplir con los requerimientos de las nuevas tecnologías, desde APB en 1995 hasta CHI en 2013. Luego de analizar los distintos estándares se concluyó en utilizar un bus AMBA APB y un bus AMBA AHB. Estas especificaciones cumplen con los requerimientos del sistema de periféricos a implementar: el generador de números aleatorios real y la interfaz de comunicación serie.

Una vez definidos los tipos de buses a utilizar, se describieron las características y el funcionamiento del bus APB. Se presentaron las señales que intervienen en la comunicación y las fases involucradas en una transferencia. Se mostraron por medio de diagramas de estados, como son las transferencias de lectura y escritura simples y las extendidas.

Se presentó el bus AHB, con las características y modos de operación. A diferencia del bus APB el AHB permite transferencias por ráfagas, admite operaciones *pipelined* y tiene la posibilidad que varios maestros controlen el bus. Todas estas características resultan en un bus con mayor desempeño que el APB y apto para dispositivos más rápidos como un procesador. Se encuentra a un nivel de jerarquía

mayor que el APB y se comunican por medio de un dispositivo adaptador de buses denominado puente o *bridge* APB-AHB.

Por último se mostró la implementación del puente APB-AHB. Mediante diagramas de estado, se describieron las señales y su funcionamiento cuando el bus AHB realiza operaciones de lectura o escritura hacia los esclavos APB. Se presentó el diseño del puente, la descripción en VHDL y la síntesis lógica utilizando la herramienta Design Vision de Synopsys. Se realizaron simulaciones, utilizando Modelsim, con celdas estándares del proceso de Tower Jazz de 180 nm. El puente APB-AHB no se implementó en el chip, y su diseño se encuentra disponible para implementarse en un FPGA. Las simulaciones se realizaron contemplando distintos casos de lecturas simples o extendidas, y seleccionando diferentes esclavos.

En el Capítulo 3 se presentó uno de los circuitos periféricos que componen el sistema AMBA. Un generador de números aleatorios real. En primer lugar se realizó la clasificación de los generadores de números aleatorios y se describió cada uno de ellos. Luego se presentaron los estándares y las guías de evaluación para validar la secuencia de salida del generador de números aleatorios. Como este dispositivo se pensó para utilizar en aplicaciones de seguridad de una tarjeta de telefonía celular, se decidió implementar un generador de números aleatorios real o TRNG.

Con el fin de elegir la topología adecuada de la arquitectura se realizó una búsqueda bibliográfica de las distintas implementaciones de generadores de números aleatorios reales. Se optó por desarrollar un diseño propuesto por Intel, cuya arquitectura sigue vigente pese a su antigüedad. Se realizó una justificación del método de obtención de aleatoriedad y se seleccionaron los criterios del diseño para la aplicación. Se describió cada uno de los bloques a nivel circuito y se mostraron sus respectivas simulaciones. Se observó el comportamiento en función de posibles cambios en las condiciones de temperatura, proceso y variaciones en la tensión de alimentación, comprobando que se cumplen con las condiciones de aleatoriedad. Se realizaron pruebas de aleatoriedad para distintas secuencias, produciendo resultados satisfactorios. En referencia a estas pruebas, varias necesitan secuencias de longitud

extremadamente grandes, generando tiempos de simulación muy largos. Se tomó la decisión de reducir la longitud de las secuencias, dejando el resultado final para la validación en el circuito integrado. Por último se implementó la interfaz AMBA APB. El diseño completo se desarrolló en una tecnología de Tower Jazz de 180 nm. Las máscaras para la fabricación se realizaron utilizando la herramienta Virtuoso de Cadence.

En el Capítulo 4 se describió una interfaz de comunicación serie cuyo desarrollo se elaboró para el estudio del bus AMBA. Se optó por realizar una UART dado que se tenía conocimiento previo del funcionamiento del protocolo de comunicación serie y es un dispositivo muy común y utilizado. Se detalló el diseño, con su protocolo de comunicación, se implementó un registro de control para variar la velocidad de transmisión dentro de las velocidades estándares y control de paridad, se implementó un registro para controlar el estado de la transmisión, pudiéndose detectar errores de comunicación, si los hubiera. También se realizó una FIFO con el fin de evitar la pérdida de datos e implementar interrupciones a futuro. Por último, se diseñó su interfaz AMBA APB con el fin de poder incorporar el periférico a un sistema actual. Con respecto a la implementación de la FIFO se contemplaron diferentes señales de reloj para el control de lectura y de escritura realizando una FIFO transmisora y otra receptora. Se mostraron simulaciones contemplando todos los casos de transmisión (FIFO, paridad, velocidad) donde se verificó el correcto funcionamiento tanto en el transmisor como en el receptor y también en transferencias con el bus APB. Luego de realizar las simulaciones correspondientes para asegurar el funcionamiento del periférico, se realizó la síntesis lógica con la herramienta Design Vision de Synopsys y la síntesis física con le herramienta de implementación digital Encounter, de Cadence. En la síntesis se obtuvieron resultados que cumplen con las restricciones con un reloj de sistema de 100 Mhz y máxima velocidad de transmisión de 19200 baudios.

Por último, en el Capítulo 5, se presentó el protocolo de validación para realizar en el circuito integrado luego de su fabricación.

Bibliografía

Altera (2005a). *Avalon Interface Specification*. Disponible desde: `http://www.altera.com`, [acceso en Enero de 2006].

Altera (2005b). *Avalon Interface Specification*. Disponible desde: `http://www.altera.com`, [acceso en Marzo de 2006].

ARM (1999). *AMBA Specifications v2.0*. Disponible desde: `http://www.arm.com`.

Ayala, J., Lopez Vellejo, M., Bertozzi, D. y Benini, L. (2006). "State-of-the-art SoC communication architecture". En: *Embedded Systems Handbook*. Ed. por Richard Zurawski. Boca Raton: CRC Press, págs. 20.1-20.22.

Bagini, V. y Bucci, M. (1999). "A design of reliable true random number generator for cryptographic applications". En: *Cryptographic Hardware and Embedded Systems*. Springer, págs. 204-218.

Baker, R.J. (2011). *CMOS: Circuit Design, Layout, and Simulation*. IEEE Press Series on Microelectronic Systems. Wiley. ISBN: 9781118038239.

Barker, E., Kelsey, J. y col. (2012). "Recommendation for Random Number Generation Using Deterministic Random Bit Generators". En: *NIST Special Publication 800-90A*.

Bassham III., Lawrence E., Rukhin, A. L., Soto, J. y col. (2010). "SP 800-22 Rev. 1a". En: *A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications, National Institute of Standards & Technology, Gaithersburg, MD*.

Benini, L. y De Micheli, G. (2005). "Networks on chip: A new paradigm for component-based MPSoC design". En: *Multiprocessor Systems-on-Chips*. Ed. por Jerraya Ahmed Amine y Wolf Wayne. Amsterdam: Elsevier. Cap. 3, págs. 49-80.

Cordan, B. (2006). *A Bus Architecture for System-on-Chip Designs, Palmchip Corporation*. Disponible desde: `www.palmchip.com`.

Fairfield, R. C., Mortenson, R. L. y Coulthart, K. B. (1985). "An LSI Random Number Generator (RNG)". En: *Proceedings of CRYPTO 84 on Advances in Cryptology*. Santa Barbara, California, USA: Springer-Verlag New York, Inc., págs. 203-230. ISBN: 0-387-15658-5.

Fischer, V. y Drutarovskỳ, M. (2003). "True random number generator embedded in reconfigurable hardware". En: *Cryptographic Hardware and Embedded Systems-CHES 2002*. Springer, págs. 415-430.

Fischer, V., Aubert, A., Bernard, F., Valtchanov, B., Danger, J.L. y Bochard, N. (2009). *True random number generators in configurable logic devices*. Inf. téc., págs. 1-58.

Golić, J. D. (2006). "New methods for digital generation and postprocessing of random data". En: *IEEE Transactions on Computers* 55.10, págs. 1217-1229.

Hennessy, J. L. y Patterson, D. A. (2003). *Computer architecture: a quantitative approach*. Elsevier.

Horspool, N. y Gorman, P. (2001). *The ASIC handbook*. Upper Saddle River: Prentice Hall.

ISO (2011). *ISO/IEC 18031: Information technology. Security techniques. Random bit generation*. ISO.

ITRS (2005). Inf. téc. Austin, TX: Semiconductor Industry Association.

Jun, B. y Kocher, P. (1999). *The Intel Random Number Generator*. White Paper Prepared For Intel Corporation by Cryptography Research, Inc.

Killmann, W. y Schindler, W. (2001). "A proposal for: Functionality classes and evaluation methodology for true (physical) random number generators". En: *T-Systems debis Systemhaus Information Security Services and Bundesamt für Sicherheit in der Informationstechnik (BSI). Inf. téc.*

Killmann, W. y Schindler, W. (2011). *A proposal for: Functionality classes for random number generators*. Disponible desde: `https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Zertifizierung/Interpretationen/AIS_20_Functionality_classes_for_random_number_generators_e.html`.

Kim, Y., Lee, H. y Perrig, A. (2014). *Information Security Applications: 14th International Workshop, WISA 2013, Jeju Island, Korea, August 19-21, 2013, Revised Selected Papers*. Lecture Notes in Computer Science. Springer International Publishing. ISBN: 9783319051499. URL: https://books.google.com.ar/books?id=Dhi6BQAAQBAJ.

Koç, C. (2009). *Cryptographic Engineering*. Springer.

Marsaglia, G. (1996). *DIEHARD: a battery of tests of randomness*. Disponible desde: http://stat.fsu.edu/~geo/diehard.html.

Marsaglia, G., Tsang, W. W. y col. (2002). "Some difficult-to-pass tests of randomness". En: *Journal of Statistical Software* 7.3, págs. 1-9.

Moore, G. E. (2006). "Cramming more components onto integrated circuits, Reprinted from Electronics, volume 38, number 8, April 19, 1965, pp. 114 ff." En: *IEEE Solid-State Circuits Newsletter* 3.20, págs. 33-35.

Neumann, J. V. (1963). " Various techniques for use in connection with random digits". En: *von Neumann's Collected Works*. Vol. 5. Pergamon, New York, págs. 768-770.

NIST (1994). *FIPS 140-1: Security Requirements for Cryptographic Modules*. Disponible desde: https://csrc.nist.gov/csrc/media/publications/fips/140/1/archive/1994-01-11/documents/fips1401.pdf.

NIST (2000). *FIPS 186-2: Digital Signature Standard (DSS)*. Disponible desde: http://csrc.nist.gov/publications/fips/archive/fips186-2/fips186-2.pdf.

NIST (2001). *FIPS 140-2: Security Requirements for Cryptographic Modules*. Disponible desde: http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.140-2.pdf.

NIST (2009). *FIPS 140-3: Security Requirements for Cryptographic Modules*. Disponible desde: https://csrc.nist.gov/csrc/media/publications/fips/140/3(2007)/draft/documents/fips140-3-final-draft-2007.pdf.

Oklobdzija, V.G. (2007). *Digital Systems and Applications*. The Computer Engineering Handbook, Second Edition. CRC Press. ISBN: 9780849386206. URL: https://books.google.com.ar/books?id=Oj4swYF0NV0C.

Palmchip (2006). *Overview of the CoreFrame Architecture*. Disponible desde: `www.palmchip.com`.

Pasricha, S. y Dutt, N. (2010). *On-chip communication architectures: System On Chip Interconnect*. Morgan Kaufmann.

Radulescu, A. y Goossens, K. (2004). "Communication services for networks on chip". En: *Domain-Specific Processors: Systems, Architectures, Modeling, and Simulation*. Ed. por Shuvra S. Bhattacharyya, Ed F. Deprettere y Jürgen Teich. New York: Marcel Dekker Inc, págs. 193-213.

Rowen, C. (2004). *Engineering the complex SOC: Fast, Flexible Design with Configurable Processors*. Upper Saddle River, NJ: Prentice Hall, PTR.

Schindler, W. (1999). "Functionality Classes and Evaluation Methodology for Deterministic Random Number Generators". En: *BSI, version* 2.

Schindler, W. (2009). "Random number generators for cryptographic applications". En: *Cryptographic Engineering*. Springer, pág. 11.

Shinde, S. V. (2014). "PVT insensitive reference current generation". En: *Proceedings of the International MultiConference of Engineers and Computer Scientists*. Vol. 2, págs. 12-14.

Siemens (1994). "OMI 324 Draft Standard: PI-Bus". En: *Rev. 0.3 d*.

Sonics (2002). *Sonics μNetwork Technical overview*. Disponible desde: `http://sonicsinc.com`, [acceso en Enero 2006].

Sunar, B., Martin, W. J. y Stinson, D. R. (2007). "A provably secure true random number generator with built-in tolerance to active attacks". En: *IEEE Transactions on Computers* 56.1, págs. 109-119.

Tkacik, T. E. (2003). "A hardware random number generator". En: *Cryptographic Hardware and Embedded Systems-CHES 2002*. Springer, págs. 450-453.

Trobec, R. (2009). "Evaluation of d-mesh Interconnect for SoC". En: *International Conference on Parallel Processing Workshops, 2009. ICPPW '09.* Págs. 507-512. DOI: `10.1109/ICPPW.2009.74`.

Vivet, P., Bernard, C., Guthmuller, E., Miro-Panades, I., Thonnart, Y. y Clermidy, F. (2015). "Interconnect Challenges for 3D Multi-cores: From 3D Network-on-Chip to Cache Interconnects". En: *2015 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, págs. 615-620. DOI: 10.1109/ISVLSI.2015.21.

VSI Alliance y On-Chip Bus Development Working Group (2001). *Virtual Component Interface Standard Version 2 (OCB 2 2.0)*.