BENEMÉRITA **U**NIVERSIDAD **A**UTÓNOMA DE **P**UEBLA

FACULTAD DE **C**IENCIAS DE LA **E**LECTRÓNICA

“Implementación de un Generador de Números Pseudoaleatorios con herramientas de Diseño Asistido por Computadora”

Tesis presentada como requisito
para la obtención del grado de:

Licenciado en Electrónica

por

Sergio Romero Camacho

dirigida por

Dr. Víctor Rodolfo González Díaz

PUEBLA, PUE.
OTOÑO 2014

Dedicatoria

A mis padres por su amor y apoyo incondicional desde siempre.

A mi hermana por ser una guía para lo que vendría.

A mis abuelos por sus atenciones y anécdotas que me ayudaron a crecer.

A mis amigos por su amistad y aceptarme como soy.

Agradecimientos

A mi asesor el Dr. Víctor Rodolfo González Díaz por sus lecciones impartidas, su apoyo y confianza para lograr la culminación de este trabajo.

A la Vicerrectoría de Investigación y Estudios de Posgrado de la Benemérita Universidad Autónoma de Puebla por el otorgamiento de un beca correspondiente al proyecto "Estudio y Compensación de Ganancia Finita en Amplificadores Operacionales Integrados", cuyo responsable es el Dr. Víctor Rodolfo González Díaz.

Índice general

Índice de figuras

Índice de tablas

Índice de anexos

Resumen

El trabajo presentado en ésta tesis consiste en la implementación de un Generador de Números Pseudoaleatorios en un Circuito Integrado de Aplicación Específica ASIC (*Application Specific Integrated Circuit*) basado en una arquitectura de acumuladores digitales con coeficientes variables en el tiempo. El circuito se implementó con herramientas de síntesis digital y de diseño a nivel transistor. Una contribución adicional del trabajo es la mejora de las propiedades aleatorias por medio de un Oscilador Aleatorio Booleano con una implementación en señal mixta que integra modelos a nivel transistor y en Lenguaje de Descripción de Hardware HDL (*Hardware Description Language*).

Capítulo 1
Introducción

La aleatoriedad hace referencia a todo proceso impredecible con aparente carencia de propósito u orden, no obstante juega un papel muy importante en diferentes campos de la ciencia y la ingeniería, como fenómeno de estudio y aplicación. Puede hallarse en la naturaleza o producirse con software o medios electrónicos. Esta última modalidad conlleva a la implementación de circuitos generadores de patrones aleatorios la cual no es una tarea trivial debido al requerimiento de un algoritmo con las características fundamentales de un evento azaroso, como la aperiodicidad e impredecibilidad.

Dentro de los circuitos generadores de aleatoriedad se encuentran los generadores de números aleatorios verdaderos TRNG (*True Random Number Generator*). Presentan un mejor nivel de impredecibilidad y su funcionamiento se basa en fenómenos naturales fortuitos. No obstante, esto último complica su implementación a nivel hardware [1]. Por otro lado, existen los generadores de números pseudoaleatorios PRNG (*Pseudo Random Number Generator*), son similares a los TRNG por producir secuencias de números impredecibles con la diferencia de ser periódicos y completamente repetibles a partir de un valor inicial o semilla. Esta propiedad de repetibilidad los hace fundamentales en muchas aplicaciones como la criptografía [2].

La utilidad de secuencias pseudoaleatorias es amplia. Los procesos de simulación estadística y de tipo Monte - Carlo las emplean como entradas para entender el comportamiento de sistemas estocásticos los cuales no son fáciles de analizar por métodos matemáticos directos [3]. En el desarrollo de software, los videojuegos se han beneficiado para recrear nuevas condiciones de juego en cada sesión. La criptografía protege la información con el empleo de un proceso pseudoaleatorio y de esta manera el mensaje a transmitir es inaccesible para usuarios no autorizados; la telemedicina, área encargada del monitoreo de la salud a distancia apoyándose de la tecnología, salvaguarda el envío de datos sobre el estado de salud de los pacientes ante posibles ataques de espionaje [4].

El tipo de implementación de esta clase de generadores depende de la arquitectura usada y el proceso de simulación aplicado. La síntesis, definida como la transformación de una especificación comportamental a una interconexión de elementos [5], es la solución más óptima para implementar y simular a nivel lógico y físico circuitos digitales de gran extensión. Si la prioridad es solo la observación del comportamiento lógico, basta simplemente con la descripción y comprobación en algún lenguaje de programación o de descripción de hardware (HDL) debido a una mayor rapidez en el tiempo de simulación en comparación a aquellas probadas a nivel transistor.

Los inconvenientes surgen al pretender integrar bloques descritos en HDL con otros a nivel transistor, cuando alguno de ellos o ambos han sido previamente diseñados. Una opción es sintetizar los modelos de HDL para el acoplamiento con aquellos a nivel transistor. No obstante, si el circuito resultante es muy extenso, la duración de la simulación sería considerable, lo cual incrementa el tiempo de diseño y por ende la puesta en el mercado. Un cambio de modelo de nivel transistor a HDL no es posible automáticamente, se requeriría describir manualmente el código fuente y conllevaría a una prolongación en el desarrollo. Es necesaria una implementación para circuitos digitales que pretenda integrar bloques descritos en HDL con algunos a nivel transistor sin necesidad de tareas intermedias, además de disponer de un método para sintetizar circuitos.

El siguiente trabajo de tesis pretende satisfacer dicha necesidad con la implementación de un generador de números pseudoaleatorios con herramientas de síntesis digital y de diseño a nivel transistor. Sin embargo, el método propuesto no se restringe únicamente para circuitos generadores de aleatoriedad, sino a sistemas digitales en general.

1.1 Objetivo

General

Investigar y documentar los procedimientos de diseño comportamental, estructural y físico para la implementación de un Circuito Integrado de Aplicación Específica (ASIC) digital con herramientas de síntesis y a nivel transistor, ejemplificado a través del diseño de un PRNG.

Específicos

- Describir un PRNG en lenguaje VerilogA.
- Diseñar un PRNG en un ASIC.
- Diseñar un oscilador aleatorio booleano a nivel transistor para mejorar las características aleatorias del PRNG.
- Caracterizar el PRNG mejorado con un *testbench* en señal mixta.

1.2 Justificación

Los beneficios principales del presente trabajo repercutirán en los procesos de diseño de sistemas digitales así como en el manejo de herramientas CAD para tales fines. Los tiempos de desarrollo podrán reducirse [6] debido a la pronta integración de los bloques constituyentes de un sistema cuando estos o algunos de ellos ya existen sin necesidad de hacer posibles conversiones de un modelo a otro o trazar circuitos muy extensos en forma manual. Tal beneficio también surge por la facultad de diseñar cada bloque del sistema en forma independiente y en el modelo más apropiado [6]. Lo anterior ofrece la ventaja de utilizar esos mismos bloques para otros proyectos futuros, similar a tener una librería de elementos estándar, solo que en este caso podrán ser configuradas al criterio del diseñador. La simulación, al tener la opción de ser de naturaleza lógica, aumentará su rapidez por la reducción de los nodos y las ramas a analizar. Para fines de fabricación, las tareas podrán ser más sencillas y rápidas por la existencia de herramientas de síntesis automática, generando una mayor confiabilidad.

Desde el punto de vista del manejo de herramientas CAD, habrá una mayor concentración de tiempo y esfuerzo en detalles propios del diseño en lugar de estudiar particularidades del manejo del programa donde, por los tiempos ajustados de desarrollo, es preferible reducirlo [6]. Lo anterior es debido a una plataforma de diseño más visual e inmediato. Al ser de tal naturaleza, se favorecerá la percepción de detalles y un aumento en decisiones estratégicas [6]. En condiciones de tiempos ajustados, se podrá evitar profundizar en lenguajes de descripción, como lo es VHDL – AMS, para realizar las mismas tareas de diseño mixto. Lo anterior no justifica el evitar aprenderlo, más bien se presenta como una alternativa de programación gráfica para simplificar las tareas y minimizar los tiempos de desarrollo, como lo son las conocidas herramientas *Simulink* de *Matlab* o *LabView*. Esta última surgió por la necesidad de cambiar el modo de construcción de instrumentos virtuales cuando hace algunos años se hacían con paquetes de programas, esto implicaba mucha inversión de tiempo en detalles de programación, lo cual no revertía en la finalidad real del instrumento [7].

En resumen, los beneficios consecuentes se darán en la productividad, tiempo y fabricación de sistemas mixtos, así como en la facilidad de las herramientas CAD para diseñar dichos sistemas. Lo anterior es de suma importancia en la industria de los circuitos integrados porque tales herramientas podrán facilitar posibles modificaciones de los diseños ante eventualidades externas como reajustes en las dimensiones del sistema, adelantos en las fechas de entrega o cambios en la arquitectura. Como ejemplo de esta última eventualidad, si se requiriese modificar gran parte del diseño físico únicamente se cambiaría la descripción comportamental en HDL sin necesidad de alterar total y manualmente las posiciones y conexiones de cada una de las celdas, considerando la cantidad elevada de ellas en un circuito integrado que puede ser del orden de millones, como en un microprocesador. Otro sector favorecido será el estudiantil perteneciente a electrónica y computación así como aquel ligado a algún posgrado o especialidad relacionada a sistemas digitales, instrumentación electrónica, circuitos integrados y comunicaciones.

1.3 Organización de la tesis

El presente trabajo consta de 5 capítulos. En el primero se exponen los antecedentes más importantes, el planteamiento del problema, los objetivos y la justificación. El capítulo 2 contiene los fundamentos acerca de los TRNG y los PRNG, se hace énfasis en una arquitectura basada en acumuladores digitales de coeficientes variables. En el capítulo 3 se muestra la implementación en ASIC del PRNG descrito en el capítulo 2 con uso de síntesis digital. El capítulo 4 comprende el diseño de un oscilador aleatorio booleano para implementar un PRNG mejorado y las verificaciones respectivas. Finalmente las conclusiones y el trabajo a futuro se formulan en el capítulo 5.

Capítulo 2

Generadores de números aleatorios

Los generadores de números aleatorios son sistemas capaces de crear secuencias de valores impredecibles. Pueden ser implementados por medios computacionales o físicos, como en FPGA o ASIC. Dichos generadores encuentran sus aplicaciones más importantes en los sistemas criptográficos modernos, de comunicación, de simulación estadística como los de tipo Monte - Carlo, entre otros [8]. De ahí la importancia de su estudio para lograr avances en otras áreas de la ciencia y la tecnología como el resguardo y protección de información ante ataques de espionaje y la ejecución de bancos de prueba a sistemas estocásticos para comprobar su funcionamiento.

En cualquiera de sus modelos, las características más importantes son:

- Estructura: referente a la arquitectura.
- Comportamiento estadístico: relacionada a la distribución de datos de salida que puede ser uniforme o no uniforme.
- Consumo de recursos: ligado a aspectos físicos y temporales como cantidad de memoria requerida, portabilidad, tiempo de arribo de datos y consumo de potencia [9].

Lo anterior varía según la aplicación para la cual se diseñe un sistema de tal tipo. El desarrollo de la tecnología en circuitos integrados ha mejorado los requerimientos computacionales donde estos últimos también están en función de la arquitectura empleada ya que según la cantidad de recursos a utilizar será el espacio y velocidad en consecuencia.

En función del método para crear aleatoriedad, los generadores de números aleatorios pueden clasificarse en dos categorías: Generadores de Números Aleatorios Verdaderos (TRNG) y Generadores de Números Pseudoaleatorios (PRNG).

2.1 Generadores de Números Aleatorios Verdaderos (TRNG)

Los TRNG son sistemas cuyo funcionamiento se basa en el empleo de fuentes naturales aleatorias, como pueden ser térmicas, amplificadoras de ruido, de ruido de fase o fenómenos cuánticos [1]. Por lo general los TRNG explotan el ruido analógico en dispositivos electrónicos para producir secuencias de bits aleatorios [10]. Las propiedades más sobresalientes de los TRNG son:

- arquitectura analógica o mixta
- funcionamiento basado en aleatoriedad natural
- longitud de periodo infinita
- alto nivel de impredecibilidad
- alto nivel de entropía en la salida

Así, debido al empleo de fuentes aleatorias naturales, los TRNG ofrecen altas prestaciones para aplicaciones de encriptado de información debido a la longitud infinita del periodo así como por la impredecibilidad. Sin embargo, en términos computacionales las arquitecturas mixtas pueden presentar dificultades al implementarse a nivel hardware por la integración de dichas fuentes de aleatoriedad natural con bloques digitales. El acoplamiento de señales así como el uso de convertidores para tales fines puede aumentar el volumen del circuito y disminuir su velocidad de operación, así como deformar la distribución de datos en la salida.

2.2 Generadores de Números Pseudoaleatorios (PRNG)

Los PRNG son sistemas cuyo funcionamiento se basa en algún algoritmo numérico. La generación de números se da a partir de un valor inicial o semilla (*seed value*), en consecuencia las secuencias son periódicas, completamente repetibles [2] y aleatorias en apariencia.

A nivel hardware estos algoritmos numéricos son implementados con módulos digitales combinacionales y secuenciales. Las propiedades más sobresalientes de los PRNG son:

- Arquitectura completamente digital
- Funcionamiento basado en un algoritmo numérico
- Longitud de periodo finita
- Predictibilidad modelada por un sistema de ecuaciones
- Nivel de entropía en función del valor inicial o semilla

De esta forma, debido al empleo de algoritmos numéricos los cuales son determinísticos, los PRNG presentan un nivel menor de impredecibilidad y entropía a los TRNG. Aun así, los PRNG pueden ofrecer también excelentes características de esta naturaleza. La impredecibilidad y la longitud del periodo, aunque están en función de la arquitectura y esta a su vez modelada por un conjunto de ecuaciones, pueden presentar un alto nivel de dificultad de resolución. En ambos casos estas propiedades mejoran al implementar un algoritmo numérico de mayor complejidad. Por lo cual la longitud del periodo, aunque finito, puede llegar a ser tan extenso que se requerirían demasiadas muestras experimentales para hallar su valor.

La mayor ventaja de los PRNG son las prestaciones computacionales. Al ser completamente digitales se eliminan los problemas de la integración en señal mixta y la susceptibilidad de deformar la distribución de datos en la salida. La implementación puede realizarse con FPGA o en un ASIC. Si es en esta última modalidad puede desarrollarse con celdas estándar. Principalmente ofrecerá una mayor velocidad de operación, menor consumo

de potencia y volumen. Independientemente del método ejecutado, la implementación en hardware de generadores de números pseudoaleatorios es cada vez más difícil debido a las fuertes restricciones en términos de consumo de potencia y área que los dispositivos modernos requieren, particularmente en el campo de aplicaciones móviles y de consumo [2]. Por lo tanto la implementación en ASIC puede ser una solución importante ante el panorama anterior.

Conforme a los objetivos de esta tesis y a la comparación entre las dos clases de generadores de valores aleatorios, el presente trabajo se enfocará en una arquitectura de PRNG basada en la conexión en cadena de acumuladores digitales de coeficientes variables. Se tratarán sus características básicas así como una comparación entre otras arquitecturas de PRNG.

2.3 PRNG basado en acumuladores digitales de coeficientes variables

Es un sistema conformado por una cadena de acumuladores digitales cuyos coeficientes varían en el tiempo por medio de un registro de desplazamiento de retroalimentación lineal LFSR (*Linear Feedback Shift Register*) [2]. Para comprender mejor tal arquitectura se describirá en primer lugar un acumulador simple sin modulación.

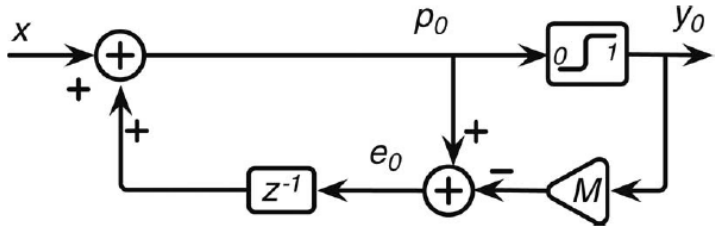


Figura 2.1. Diagrama a bloques de un acumulador digital simple de módulo *M*, sin variación de coeficientes [2].

Con base en la figura anterior, su función es sumar un valor constante o semilla (*X)* con el resultado de la adición anterior ($z^{-1}e_0$). El resultado de la operación anterior es p_0 el cual ingresa a un comparador que determina si existe un acarreo de salida (y_0), esto cuando p_0 es igual o rebasa el módulo (*M*) o valor límite que ya no es capaz de generar el acumulador. Al haber un acarreo de salida de 1 ($y_0 = 1$) la salida del generador (e_0) es la diferencia entre p_0 y el valor del módulo *M*, es decir el conteo se reinicia a 0 y se suma a esto el excedente de p_0 si lo hay. Finalmente se aplica a e_0 un retardo (*1/z*) en el tiempo discreto para retener el resultado y poder sumarse en el siguiente proceso. Las ecuaciones que describen la evolución en el tiempo de dicho acumulador son las siguientes.

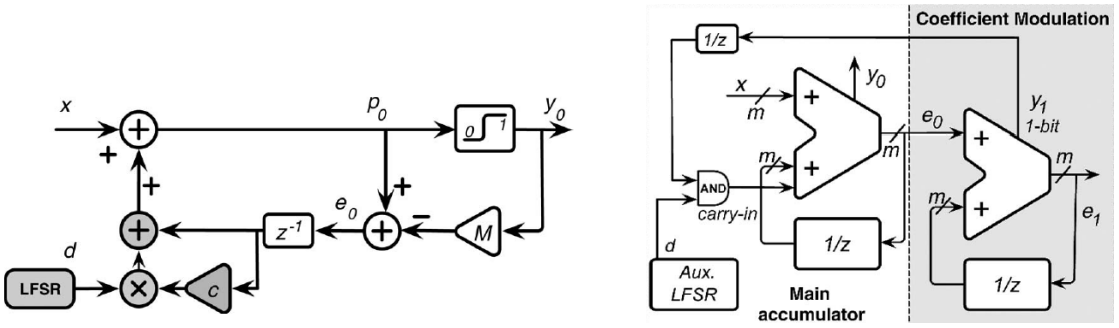
$$p_0[i] = X + e_0[i-1]$$

$$y_0[i] = \begin{cases} 0 & p_0[i] < M \\ 1 & p_0[i] \geq M \end{cases}$$

$$e_0[i] = p_0[i] - My_0[i] = p_0[i] \quad (mod.M) \qquad (1.1)$$

Si el acarreo de salida es 0 ($y_0 = 0$), e_0 es igual a p_0 ya que el acumulador aún puede representar tal cantidad por ser menor a *M*.

Un acumulador digital de coeficientes variables es un acumulador simple como el descrito anteriormente, en conjunto con una etapa moduladora. A continuación su respectivo diagrama a bloques junto a su circuito respectivo.



(a) **(b)**

Figura 2.2. Acumulador digital de coeficientes variables de módulo *M*. (a) Diagrama a bloques. (b) Circuito digital respectivo, donde $M = 2^m$ [2].

En la figura 2.2(a) el modulador de coeficientes está formado por aquellos bloques sombreados que son: un multiplicador, un Registro de Corrimiento de Retroalimentación Lineal LFSR (*Linear Feedback Shift* Register) y un bloque *c* que puede ser ejecutado como un modulador de coeficientes. Estos dos últimos generan salidas 1 o 0 donde, al multiplicarse, generan un valor 1 o 0 que finalmente es sumado al resultado con un retardo de un ciclo de reloj del acumulador principal y al valor semilla *X*. En la figura 2.2(b) se aprecia la implementación de dicho sistema, conformado por *m* bits, de tal manera que el valor límite del acumulador es $M = 2^m$. En él se observa que el bloque variante *c* es en realidad un acumulador simple, donde la salida es el bit de acarreo y_1. Esta salida ingresa a una compuerta AND que funge como multiplicador de 1 bit para realizar la operación respectiva junto con la señal del LFSR. La salida de dicha compuerta es la salida del multiplicador, la cual es ingresada como acarreo de entrada al acumulador principal.

Finalmente, si se conectan en cadena este tipo de acumuladores se obtiene la arquitectura del PRNG de esta sección. La implementación respectiva es la siguiente.

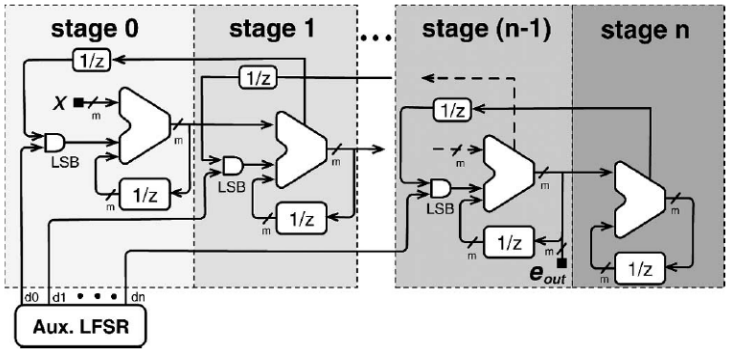


Figura 2.3. Circuito del PRNG formado por una cadena de acumuladores digitales de coeficientes variables [2].

Acorde a la figura anterior, la salida del acumulador de la etapa *n - 1* (e_{out}) es también la del PRNG la cual es de *m* bits de resolución. El número de bits del LFSR depende únicamente del número de etapas insertadas, de tal forma que si existe un total de *n +1*, donde *n* es la posición de la última etapa, entonces se requerirá un LFSR de *n* bits puesto que el último bloque no requiere un multiplicador. Los bloques de retardo se implementan con un registro

de entradas - salidas paralelas. A excepción del LFSR y las compuertas AND, todos los bloques son de *m* bits.

Las ecuaciones que describen la evolución de las variables dentro del PRNG son:

$$p_k[i] = \begin{cases} X + e_0[i-1] + y_1[i-1]d_0[i], & k = 0 \\ e_{k-1}[i] + e_k[i-1] + y_{k+1}[i-1]d_k[i], & 0 < k < n \\ e_{n-1}[i] + e_n[i-1], & k = n \end{cases}$$

$$y_k[i] = \begin{cases} 0 & p_k[i] < M \\ 1 & p_k[i] \geq M \end{cases}$$

$$e_k[i] = p_k[i] - My_k[i] = p_k[i] \quad (mod.M) \qquad (1.2)$$

Donde el subíndice *k* hace referencia a la señal de la *k - esima* etapa. A excepción de la primera y la última, la señal p_k depende de la salida de la etapa que le precede ($e_{k-1}[i]$), el resultado anterior arrojado ($e_k[i - 1]$) y la modulación del acarreo de salida de la etapa sucesora y el *k - esimo* bit del LFSR ($y_{k+1}[i - 1]d_k[i]$). Las ecuaciones para y_k y e_k mantienen la misma estructura del acumulador simple.

El generador anterior, por ser pseudoaleatorio, presenta periodicidad. La periodicidad del sistema es definida como el menor entero $I > 0$ tal que $e_k[i + I] = e_k[i]$, o en forma equivalente $e_k[I] = \bar{e}_k$, donde $\bar{e}_k = e_k[0]$, es decir, representa la condición inicial de la salida de la *k-esima* etapa del PRNG [2]. Para el caso especial de la salida del generador $e_{out}[I] = e_{n-1}[I] = \bar{e}_{n-1}$, con $k = n - 1$. De acuerdo a [2], la periodicidad es modelada por el siguiente sistema de ecuaciones en aritmética modular no lineales en *I*.

$$\begin{cases} IX = 0 & (mod\ M) \\ \frac{I^{(2)}}{2!}X + I\bar{e}_0 = 0 & (mod\ M) \\ \dots & \\ \frac{I^{(n)}}{n!}X + \sum_{l=1}^{n-1}\frac{I^{(l)}}{l!}\bar{e}_{n-l-1} = 0 & (mod\ M) \end{cases} \qquad (1.3)$$

Donde I depende de X, M, n y de todo $\bar{e}_k$. Si se desprecia la solución trivial $I = 0$ y $X = 0$, debido a la periodicidad del LFSR, I es un múltiplo entero de la periodicidad del LFSR el cual es independiente de X. Por lo cual el periodo puede mejorarse a través del LFSR [2], esto también es posible con un número mayor de etapas en la cadena de acumuladores. Así, el valor semilla solo afecta la secuencia de números generados, siendo diferentes para cada valor particular de X.

Como puede observarse, dicho sistema consta de módulos digitales simples como una ALU, una compuerta AND, un registro de entradas - salidas paralelas así como un registro de desplazamiento de retroalimentación lineal. Por lo anterior, es un generador de bajo consumo de recursos como lo demuestra la siguiente comparación entre otras arquitecturas.

Tabla 2.1. Comparativo entre diferentes arquitecturas de PRNG y el del presente trabajo [2].

Diseño	**PRNG elegido**	**[11]**	**[12]**	**[13]**
Arquitectura	xc4vfx12 (Virtex-4)	xc4vfx100 (Virtex-4)	xc2000e (Virtex-E)	N/A
Slices (LTUs)	57 (91)	128 (213)	330 (539)	429 (N/A)
RAM	No	4 bloques	2 bloques	Sí (no especificado)
Velocidad (MSa/s)	90.98	26.13	24.16	38.14

De acuerdo a la tabla anterior, el bajo consumo de recursos del PRNG elegido se ve reflejado en la mínima cantidad de Slices *y* LTUs utilizados en los FPGA respectivos de la familia Virtex de Xilinx. Lo anterior representa un ahorro mínimo del 55% y 57% respectivamente en comparación a [11]. Como consecuencia de tal ahorro el generador elegido posee la mayor velocidad, siendo esta aproximadamente más del doble que la de [13]. La ausencia de RAM en el PRNG elegido se debe a la arquitectura basada en acumuladores digitales, donde únicamente requiere un valor inicial o semilla para crear la cadena de valores pseudoaleatorios. En cambio, los PRNG de arquitectura Mersenne Twister como los diseños [11], [12] y [13] requieren bloques RAM para inicializar el proceso con un conjunto de números pre - calculados. Así, para [11] los dos bloques de RAM emiten en la salida 624 números inicializados, lo cual consume más tiempo.

De acuerdo a [2], el PRNG se implementó también en un ASIC con las siguientes características.

- Proceso tecnológico: CMOS 0.35µm
- Herramienta de software usada: *Cadence Silicon Ensemble*
- Número de celdas empleadas: 474

De igual manera el número de celdas empleadas son reflejo del ahorro de recursos, sobre todo por la omisión de bloques RAM. Los parámetros anteriores servirán para compararlos con la implementación en ASIC que se realizó en este trabajo, con las herramientas de *Menthor Graphics* y con el mismo proceso tecnológico de 0.35µm.

2.4 Resumen

Los generadores de números pseudoaleatorios son sistemas capaces de crear secuencias de valores impredecibles. Se emplean principalmente en la criptografía, sistemas de comunicación y procesos de simulación estadística. Las características más importantes de estos sistemas son la estructura, el comportamiento estadístico y los recursos consumidos para su implementación. Existen dos clases de generadores: los TRNG basados en fuentes naturales aleatorias y los PRNG basados en algoritmos numéricos. Los primeros presentan mejor impredecibilidad y los segundos mayores facilidades para su implementación física. Por tales razones se expone una arquitectura de PRNG basada en una cadena de acumuladores digitales de coeficientes variables en el tiempo. Se describió su comportamiento en el tiempo discreto así como el sistema de ecuaciones que modelan la periodicidad. Un comparativo de dicho generador con otros tres de arquitectura Mersenne Twister demuestra su superioridad en velocidad y ahorro de recursos en bloques de RAM y LTUs. Se han mostrado las características principales del generador implementado en ASIC para compararlo más adelante con el desarrollado en este trabajo.

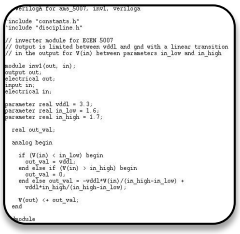
Capítulo 3

Implementación de un PRNG en un ASIC

En el capítulo anterior se estudiaron las características de los TRNG y PRNG. Se enfatizó en estos últimos por su facilidad de implementación a nivel hardware y por sus múltiples aplicaciones en procesos de simulación y criptografía. En específico, se profundizó en una arquitectura basada en acumuladores digitales y se demostró su bajo consumo de recursos tanto en FPGA como en ASIC. Por estas razones se eligió dicho sistema para ejemplificar el flujo de diseño del PRNG en un ASIC con un proceso de síntesis digital partiendo desde el nivel comportamental, lo cual se aborda en el presente capítulo.

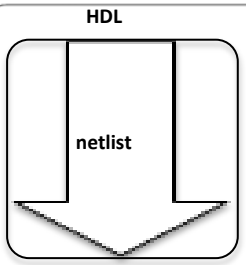
3.1 Flujo de diseño de un ASIC mediante síntesis digital

Los ASIC son circuitos integrados que desempeñan única y exclusivamente una función particular; son diseñados en forma total o parcialmente a la medida. Para esta última categoría y en el caso de circuitos digitales, el diseño puede crearse con celdas estándar las cuales son un conjunto de bloques digitales como compuertas lógicas, circuitos combinacionales y secuenciales, de tamaños y retardos de propagación bajo condiciones de carga finitas. El diseño puede ser trazado como un esquemático conformado por dichas celdas o generado automáticamente desde un alto lenguaje de hardware. Con ello, más tarde es posible el trazado físico o *layout* del diseño [5]. El proceso de generar en forma automática un circuito a nivel de celdas desde un lenguaje de descripción de hardware se conoce como síntesis digital. Dicho proceso forma parte del flujo de diseño de un ASIC parcialmente a la medida. Éste se divide en diferentes niveles de diseño, los cuales se explican en la tabla 3.1.



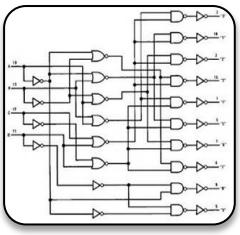
Diseño comportamental

- Se detalla unicamente el comportamiento del sistema sin ahondar en los componentes digitales para lograrlo, a travez de algun lenguaje de descripción de hardware. Posteriormente se desarolla un proceso de simulación el cual solo mostrará el estado lógico de las señales.



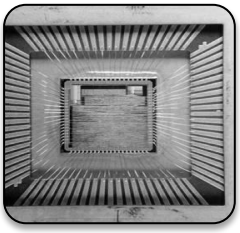
Síntesis

- Por medio de un software sintetizador se ejecuta la conversión automática de un sistema descrito en HDL a una descripción basada en celdas digitales, es decir en componentes electrónicos. El resultado es un código fuente de síntesis o *netlist* el cual es importado más adelante para construir el circuito a nivel esquemático. Tareas como la optimización, reporte de retardos de propagación así como el reporte de celdas empleadas son tambien elaboradas.



Diseño estructural

- Se importa el *netlist* para trazar automaticamente el circuito a nivel esquemático. Se verifica la estructura del circuito así como los puertos y celdas que lo conforman, desde simples compuertas lógicas hasta circuitos combinaciones y secuenciales. La simulación expondrá no solo el estado lógico de las señales, sino tambien el físico como los retardos de propagación, voltajes y corrientes, derivados del empleo de celdas digitales.



Diseño físico

- Se implementa en forma automática el circuito a nivel chip o *layout*, con base en los componentes y conexiones del diagrama esquemático en la etapa anterior. Se trazan las areas asignadas para la colocación de celdas, pads y pistas de conexión. Posteriormente se colocan las celdas y pads respectivas y finalmente se conectan todos los elementos. En este nivel se indican, además de las celdas, los materiales de las que están hechas, como metales y semiconductores. Es finalmente este circuito el diseño físico del sistema digital, a partir del cual se inicia el proceso de fabricación.

Tabla 3.1. Flujo de diseño de un ASIC parcialmente a la medida [5].

La importancia de conocer el flujo de diseño de un ASIC parcialmente a la medida radica en la posibilidad de diseñar circuitos con un menor consumo de potencia, mayor portabilidad, eliminación de componentes externos, menores tensiones de alimentación y una mayor velocidad de operación [14], en comparación a otras implementaciones basadas en algún dispositivo digital reprogramable, como FPGA o microcontrolador. Si bien el costo fijo inicial de estos últimos es menor en bajos volúmenes de producción, lo contrario ocurre para una alta cantidad de piezas [15]. Los ASIC, a pesar de requerir un mayor tiempo de desarrollo contra el

empleado para sistemas basados en dispositivos reprogramables, acortan en gran parte el proceso gracias al diseño parcialmente a la medida con el uso de celdas estándar, en contraste también con el estilo totalmente a la medida (*full custom*). Aun así, el tiempo invertido se justifica para las altas prestaciones del circuito a generar.

Las aplicaciones de los ASIC alcanzan los productos de gran consumo como cámaras fotográficas, automóviles, microprocesadores y equipos telefónicos [15]. Los sistemas de telecomunicaciones se han visto favorecidos ya que en especial la tecnología CMOS convencional tiene un papel más claro en receptores de alta frecuencia. Los ASIC han logrado avances en los circuitos de comunicación para las áreas de: receptores directos de satélites, redes locales de alta velocidad, comunicación inalámbrica, comunicaciones de banda ancha y transferencia y proceso de imágenes [14]. Los microprocesadores son una mención especial puesto que se emplea el flujo de diseño antes mencionado para su desarrollo en adición con otros procesos particulares.

3.2 Procedimiento de implementación

A continuación se resumen los pasos ejecutados en el presente trabajo para la implementación del PRNG en un ASIC, el cual se basa en el flujo de diseño mostrado en la sección anterior. Todos los entornos de desarrollo forman parte del paquete de software *Menthor Graphics*. La elección de su uso se fundamenta por las prestaciones para el desarrollo de ASIC de manera automática, además de ser líder en la automatización de diseño electrónico con vigencia actual en universidades del extranjero y en México, entre ellas la BUAP en la Facultad de Ciencias de la Electrónica. Los entornos de desarrollo se manejaron conforme a [16] por ser la referencia más actualizada y de fácil acceso.

Tabla 3.2. Procedimiento seguido para la implementación del PRNG en un ASIC.

Orden de ejecución	Etapa	Tareas	Entorno de desarrollo
1	Descripción y simulación en HDL a nivel comportamental	▪ Compilación ▪ Simulación	VSIM
2	Síntesis digital	▪ Creación del *netlist*	Leonardo Spectrum
3	Generación automática del circuito esquemático	▪ Importación del *netlist* ▪ Edición del archivo de mapeo ▪ Chequeo del circuito	Editor de símbolos y esquemáticos del Kit de Diseño de ASIC ADK (*ASIC Design Kit*)
4	Simulación estructural	▪ Configuraciones de tipo de análisis ▪ Tecnología empleada ▪ Forzamiento de señales ▪ Visualización de salidas	Editor de esquemáticos de ADK
5	Generación automática del *layout*	▪ Trazado del área ▪ Emplazamiento ▪ Ruteo	Editor de layout de ADK

3.3 Descripción y simulación en HDL

Con uso del entorno VSIM se empleó el lenguaje VerilogA por ser en la industria un estándar para sistemas no solamente digitales sino también analógicos y mixtos. La descripción se realizó en estilo comportamental por ser más sencillo, entendible y de menor inversión de tiempo, sobre todo para la complejidad del PRNG electo.

De acuerdo a la figura 2.3, el generador se compone de una cadena de $n + 1$ acumuladores de m bits, donde n es la posición del último, junto a un LFSR de n bits. Las cantidades definidas fueron: 9 etapas de acumuladores ($n = 8$) de 8 bits cada uno ($m = 8$) y un LFSR también de 8 bits en consecuencia por el número de etapas. Con dichas especificaciones se logró un PRNG de propiedades estadísticas aprobatorias a los estándares solicitados en [2].

Previo a la elaboración del código fuente y para una mejor comprensión del mismo, se enlistaron cada uno de los módulos integrantes del generador y se agruparon en jerarquías, debido a que dicha organización es la propia del sistema. Todos los módulos fueron descritos en lenguaje VerilogA y organizados como lo muestra la siguiente figura.

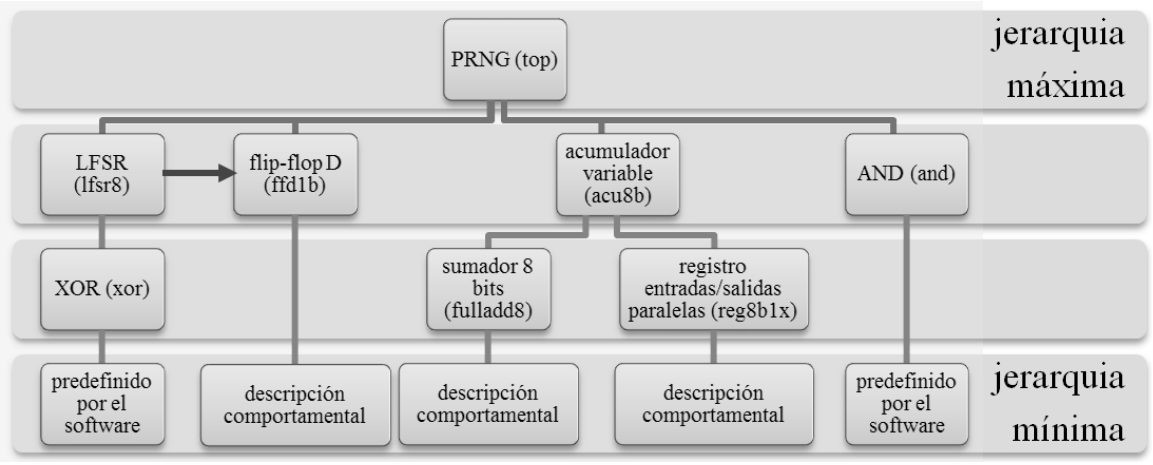


Figura 3.1. Jerarquía y relación entre módulos respectivos del PRNG.

Para la composición del código fuente con base al diagrama anterior, se comenzó con la declaración del módulo principal, nombrado como *top* y representando al PRNG en su totalidad. De igual forma se procedió con el LFSR, el acumulador de coeficientes variantes y el flip - flop D como módulos de menor jerarquía. La tarea se repitió hasta alcanzar la descripción total de aquellos con la menor jerarquía. La siguiente figura plasma en forma explícita las entradas y salida respectivas del PRNG como un módulo de nombre *top*.

Figura 3.2. Entradas y salidas del PRNG a implementar en VerilogA.

Donde *clk* es la señal que sincroniza los cambios de estado internos y de valores pseudoaleatorios ocurridos en cada flanco de subida; *reset* es la señal de reinicio asíncrono tal que cuando es activa en nivel alto se suspende la secuencia de números y la salida adopta el valor semilla (*xin[7:0]*) con el cual se comienza el proceso de generación de números y *pout[7:0]* es la salida de valores pseudoaleatorios.

La descripción final resultó en el siguiente código fuente en VerilogA.

```
1   //Digital pseudo random generator wit time variant coefficients
2   //`timescale 1ns/100ps
3   module top (xin, pout, res, clk);
4   input [7:0] xin;
5   input res, clk;
6   output [7:0] pout;
7   wire [7:0] err0; wire [7:0] err1; wire [7:0] err2; wire [7:0] err3;
8   wire [7:0] err4; wire [7:0] err5; wire [7:0] err6; wire [7:0] err7;
9   wire ci0, ci1, ci2, ci3, ci4, ci5, ci6, ci7;
10  wire co1, co2, co3, co4, co5, co6, co7, co8;
11  wire co1d, co2d, co3d, co4d, co5d, co6d, co7d, co8d;
12  wire l8, l7, l6, l5, l4, l3, l2, l1;
13
14  lfsr8 lfsr0 (clk, res, l8, l7, l6, l5, l4, l3, l2, l1);
15
16
17  acu8b acu0 (ci0 ,   xin,   err0,     , res, clk);
18  acu8b acu1 (ci1 ,  err0,   err1,  co1, res, clk);
19  acu8b acu2 (ci2 ,  err1,   err2,  co2, res, clk);
20  acu8b acu3 (ci3 ,  err2,   err3,  co3, res, clk);
21  acu8b acu4 (ci4 ,  err3,   err4,  co4, res, clk);
22  acu8b acu5 (ci5 ,  err4,   err5,  co5, res, clk);
23  acu8b acu6 (ci6 ,  err5,   err6,  co6, res, clk);
24  acu8b acu7 (ci7 ,  err6,   pout,  co7, res, clk);
25  acu8b acu8 (1'b0,  pout,       ,  co8, res, clk);
26
27
28  ffd1b  ff0 (co1d,    ,  co1,  res, clk);
29  ffd1b  ff1 (co2d,    ,  co2,  res, clk);
30  ffd1b  ff2 (co3d,    ,  co3,  res, clk);
31  ffd1b  ff3 (co4d,    ,  co4,  res, clk);
32  ffd1b  ff4 (co5d,    ,  co5,  res, clk);
33  ffd1b  ff5 (co6d,    ,  co6,  res, clk);
34  ffd1b  ff6 (co7d,    ,  co7,  res, clk);
35  ffd1b  ff7 (co8d,    ,  co8,  res, clk);
36
37  and (ci0, co1d, l8);
38  and (ci1, co2d, l7);
39  and (ci2, co3d, l6);
40  and (ci3, co4d, l5);
41  and (ci4, co5d, l4);
42  and (ci5, co6d, l3);
43  and (ci6, co7d, l2);
44  and (ci7, co8d, l1);
45
46  endmodule
47
48
49
50
51
52  module acu8b (c_in, in, err, c_out, res, ck);
53  input [7:0] in;
54  input c_in, res, ck;
55  output [7:0] err;
56  output c_out;
57  wire [7:0] err;
58  wire [7:0] errd;
59  fulladd8 add1 (err, c_out, in, errd, c_in);
60  reg8b1x  reg1 (errd, err, res, ck);
61  endmodule
```

Figura 3.3. Descripción del PRNG en VerilogA, líneas 1 a 61.

```
62
63
64
65  module fulladd8 (sum, c_out, a, b, c_in);
66  output [7:0] sum;
67  output c_out;
68  input [7:0] a, b;
69  input c_in;
70  assign {c_out, sum} = a + b + c_in;
71  endmodule
72
73
74
75
76  module reg8b1x (q, d, r, ck);
77  input [7:0] d;
78  input r, ck;
79  output [7:0] q;
80  reg [7:0] q;
81  always @( posedge ck or posedge r)
82  if (r)
83    q = 1'b0;
84  else
85    q = d;
86  endmodule
87
88
89
90  module lfsr8 (clk, res, q7, q6, q5, q4, q3, q2, q1, q0);
91  input clk, res;
92  output q7, q6, q5, q4, q3, q2, q1, q0;
93  wire q0, q1, q2, q3, q4, q5, q6, q7;
94  wire qb7;
95
96  wire q0x, q1x, q2x;
97  ffd1b  ff0 (q0  ,    ,qb7 ,   res, clk);
98  ffd1b  ff1 (q1  ,    ,q0x ,   res, clk);
99  ffd1b  ff2 (q2  ,    ,q1x ,   res, clk);
100 ffd1b  ff3 (q3  ,    ,q2x ,   res, clk);
101 ffd1b  ff4 (q4  ,    ,q3  ,   res, clk);
102 ffd1b  ff5 (q5  ,    ,q4  ,   res, clk);
103 ffd1b  ff6 (q6  ,    ,q5  ,   res, clk);
104 ffd1b  ff7 (q7  , qb7,q6  ,   res, clk);
105 xor (q0x, q0, qb7);
106 xor (q1x, q1, qb7);
107 xor (q2x, q2, qb7);
108 endmodule
109
110
111 module ffd1b (q, qn, d, r, ck);
112 input d;
113 input r, ck;
114 output q;
115 output qn;
116 reg  q;
117 wire qn;
118 assign qn = !q;
119 always @( posedge ck or posedge r)
120 if (r)
121   q=1'b0;
122 else
123   q = d;
124 endmodule
```

Figura 3.4. Descripción del PRNG en VerilogA, líneas 62 a 124.

Finalmente se guardó el archivo de descripción con el nombre *prng.v*. Seguidamente se compiló con el comando `vlog prng.v`.

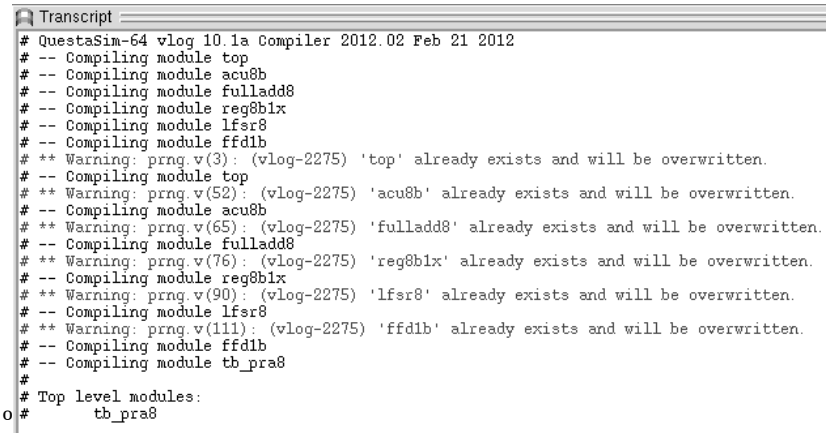


Figura 3.5. Confirmación de proceso de compilación.

Una vez verificada la sintaxis se prosiguió con la simulación a nivel comportamental a fin de comprobar el funcionamiento esperado conforme a lo explicado en el capítulo 2. Se ingresó el comando `vsim top` propiamente en la ventana de comandos para establecer el modo de simulación, donde *top* es el módulo de mayor jerarquía. Al ingresar a tal modalidad se forzaron las señales de entrada mediante el método de serie de comandos, el cual requirió una menor cantidad de líneas de código, a diferencia de un banco de pruebas o código *testbench*.

El forzamiento de señales se indicó en la siguiente forma:

- clk: señal de reloj de periodo 30ns, con 15ns en nivel bajo y alto.
- res: pulso periódico de 210ns, con 60ns en alto y un retardo de 30ns.
- xin: constante de valor “10101010” (170 en decimal) a partir de 0ns.

Lo anterior quedó expresado en las siguientes líneas de comando.

```
force clk 0 0ns, 1 15ns –repeat 30ns
force res 0 0ns
force res 1 30ns
force res 0 90ns
force res 1 240ns
force res 0 300ns
force xin 10101010 0ns
```

Seguidamente se estableció un tiempo de simulación de 450ns especificado como `run 450ns`, con el fin de determinar el comportamiento del generador antes y después de dos ciclos de la señal de reinicio (*res*) y comparar resultados entre ellos.

Una vez verificado el sistema se compiló un conjunto de librerías pertenecientes al Kit de Diseño de ASIC llamadas ADK *(ASIC Design Kit)*, requeridas para sintetizar más tarde el código fuente del PRNG. Para ello se ingresaron los siguientes comandos:

```
vlib /home/us1/wdmen/tutorial_vsim/adk
vmap adk /home/us1/wdmen/tutorial_vsim/adk
vlog /opt/mgc/adk3_1/technology/adk.v -work adk
```

El comando `vlib` crea una nueva carpeta en la ruta especificada donde se instanciarán las librerías de ADK, `vmap` mapea dichas librerías hacia la carpeta creada y por último `vlog` las compila.

3.4 Síntesis digital

Se utilizó *Leonardo Spectrum* como software sintetizador. También se contemplaron otros detalles de importancia como los retardos de propagación y el tipo y número de celdas digitales requeridos para la conversión a circuito. El proceso concluyó con la creación de un código fuente de síntesis o *netlist*, de terminación *.gate.v*, con la descripción del generador a nivel de celdas.

A continuación se enlistan los pasos ejecutados en *Leonardo Spectrum*:

a) **Declaración de la tecnología:** especificada en µm. Esta medida indica la resolución de las máscaras fotolitográficas para fabricar el circuito. Usualmente coincide también con la longitud mínima de canal de los dispositivos CMOS. La tecnología indicada para este trabajo fue de 0.35µm ya que, debido a la gran extensión del circuito y por ello la gran longitud de las pistas de conexión, permitiría un grosor resistente de componentes a nivel *layout*, además de poder trabajar con tensiones de 3.3V.

   Comando ingresado

   ```
   load_library/opt/mgc/adk3_1/technology/leonardo/tsmc035_typ.syn
   ```

   Donde `tsmc035_typ.syn` es el archivo para cargar dicha tecnología y `/opt/mgc/adk3_1/technology/leonardo/` su ruta de acceso.

b) **Lectura del código fuente del PRNG:** para indicar el archivo de código fuente.

   Comando ingresado

   ```
   read {/home/us1/wdmen/tutorial_vsim/prng.v }
   ```

   Donde `/home/us1/wdmen/tutorial_vsim/` es la ruta de acceso al código fuente `prng.v`.

c) Optimización: para implementar los módulos del PRNG en forma de circuitos combinacionales y secuenciales, con el empleo de compuertas lógicas y biestables de la tecnología declarada. En la parte secuencial, se reducen el número de estados lógicos.

Comando ingresado

```
optimize
```

d) Generación de reporte de retardos de propagación en los nodos del circuito: para observar las características temporales, resultado de la optimización anterior.

Comando ingresado

```
report_delay
```

e) Generación de reporte de área de celda: para obtener una listar del tipo y la cantidad de ellas que empleó el sintetizador.

Comando ingresado

```
report_area –cell
```

f) Escritura del *netlist*: para ordenar la creación de tal archivo.

Comandos ingresados

```
1) set vhdl_write_use_packages {library ieee,adk; use
   ieee.std_logic_1164.all; use adk.adk_components.all;}
2) apply_rename_rules -ruleset VERILOG
3) auto_write /home/us1/wdmen/tutorial_vsim/top.gate.v
```

El comando 1) invoca las librerías necesarias para la tarea, similar a las descripciones en VHDL, 2) establece el conjunto de reglas de Verilog y finalmente 3) ordena la escritura del *netlist*.

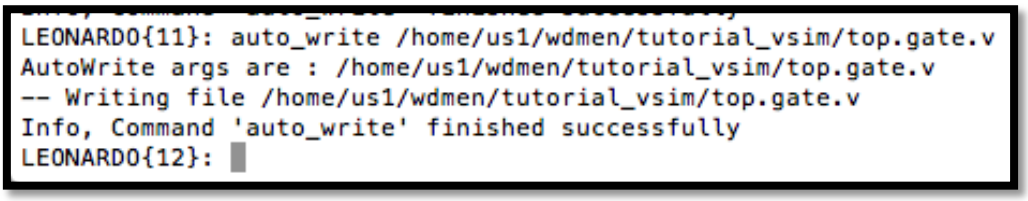


Figura 3.6. Confirmación de la generación del *netlist*.

Espacio en blanco en forma intencional

3.5 Generación automática del circuito esquemático

El *netlist* del paso anterior se importó al editor de esquemáticos de ADK de *Menthor Graphics* con el fin de trazar automáticamente el circuito del PRNG a nivel estructural. En el editor se empleó la opción *Import Verilog*. Fueron tres los requisitos para dicho trazado: archivo *netlist*, directorio para almacenar el circuito generado y un archivo de mapeo. Este último requirió un proceso adicional.

El archivo de mapeo es un conjunto de sentencias cuyo objetivo es asociar cada módulo o celda que describe al generador en el *netlist* con su equivalente visual a nivel transistor proveniente de alguna librería. En otras palabras, sirve para convertir el código a un circuito transistorizado visible. En dichas sentencias se especifican:

a) el tipo de componente, si es de tipo módulo (m) o puerto (p)
b) el nombre del componente (por ejemplo inv02, xor2, portin, portout, entre otros)
c) la ruta hacia el símbolo del componente transistorizado reemplazante
d) el nombre de los puertos del componente transistorizado

Para tal proceso, se indicaron componentes de las librerías de ADK y de algunas genéricas. En un inicio el programa genera un archivo de mapeo genérico el cual contiene las sentencias de los componentes requeridos para la creación del circuito, pero sin indicar las rutas hacia los símbolos. La especificación de ellas se realizó manualmente. El archivo final se muestra a continuación.

```
p portin $ADK/technology/ic/symbols/portin X
p portout $ADK/technology/ic/symbols/portout X
p portbi $ADK/technology/ic/symbols/portbi X
p ripper $MGC_IC_GENERIC_LIB/rip/1X1 bundle wire
p pagein $MGC_IC_GENERIC_LIB/offpag.in IN
p pageout $MGC_IC_GENERIC_LIB/offpag.out OUT
p netcon $ADK/technology/ic/symbols/netcon IN OUT
p ground $ADK/technology/ic/symbols/gnd X
p power $ADK/technology/ic/symbols/vdd X
//
m PadInC $ADK/parts/pads/tsmc035/PadInC DataIn DataInB Pad
m PadOut $ADK/parts/pads/tsmc035/PadOut Pad DataOut
m xnor2 $ADK/parts/xnor2 Y A0 A1
m nand02 $ADK/parts/nand02 Y A0 A1
m dffr $ADK/parts/dffr Q QB D CLK R
m mux21_ni $ADK/parts/mux21_ni Y A0 A1 S0
m xor2 $ADK/parts/xor2 Y A0 A1
m aoi32 $ADK/parts/aoi32 Y A0 A1 A2 B0 B1
m mux21 $ADK/parts/mux21 Y A0 A1 S0
m aoi22 $ADK/parts/aoi22 Y A0 A1 B0 B1
m inv01 $ADK/parts/inv01 Y A
m buf04 $ADK/parts/buf04 Y A
m inv02 $ADK/parts/inv02 Y A
```

Las configuraciones finales de la opción *Import Verilog* quedaron indicadas como se muestra a continuación.

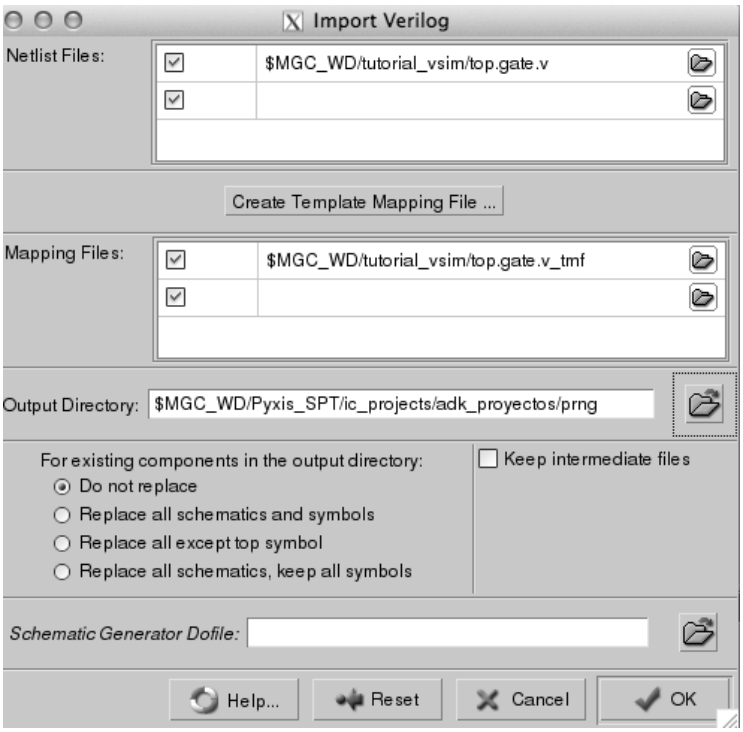


Figura 3.7. Configuraciones previas al trazado del PRNG a nivel esquemático.

Una vez editado el archivo de mapeo, se trazó automáticamente el circuito en forma exitosa, el cual consistió en una celda con seis hojas esquemáticas interrelacionadas que componen al generador (ver anexo A), junto con un símbolo representativo para usar en procesos de simulación.

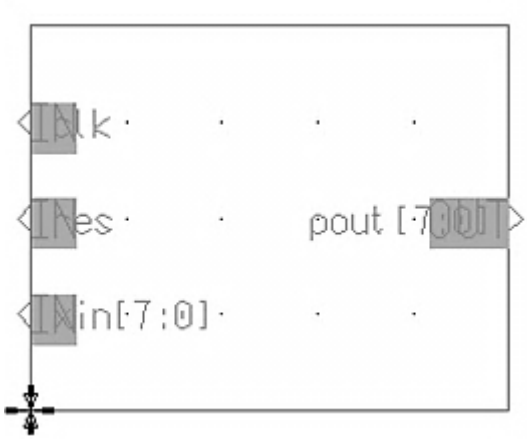


Figura 3.8. Símbolo representativo del PRNG a nivel estructural.

Posteriormente se verificó la validez de cada una de las hojas anteriores con la opción *check & save*, además del símbolo.

```
Note: "top/schematic/sheet1" passed check
Note: Version 2 of sheet $MGC_WD/Pyxis_SPT/ic_projects/adk_proyectos/prng/top/schematic/sheet1 has been written
Note: "top/schematic/sheet2" passed check
Note: Version 2 of sheet $MGC_WD/Pyxis_SPT/ic_projects/adk_proyectos/prng/top/schematic/sheet2 has been written
Note: "top/schematic/sheet3" passed check
Note: Version 2 of sheet $MGC_WD/Pyxis_SPT/ic_projects/adk_proyectos/prng/top/schematic/sheet3 has been written
Note: "top/schematic/sheet4" passed check
Note: Version 2 of sheet $MGC_WD/Pyxis_SPT/ic_projects/adk_proyectos/prng/top/schematic/sheet4 has been written
Note: "top/schematic/sheet5" passed check
Note: Version 2 of sheet $MGC_WD/Pyxis_SPT/ic_projects/adk_proyectos/prng/top/schematic/sheet5 has been written
Note: "top/schematic/sheet6" passed check
Note: Version 2 of sheet $MGC_WD/Pyxis_SPT/ic_projects/adk_proyectos/prng/top/schematic/sheet6 has been written
```

Figura 3.9. Verificación exitosa del PRNG esquemático.

Finalmente se empleó la opción *Update LVS* cuya función es actualizar cualquier modificación hecha en el circuito esquemático y trasladarla hacia el modelo físico o *layout*, además de crear un archivo de registro de extensión *.ncf* usado para procesos de simulación. Lo anterior quedó confirmado con el siguiente mensaje.

```
Note: Current session Design Configuration set to: $MGC_WD/Pyxis_SPT/ic_projects/adk_proyectos/prng/top/default
Note: Current session Design Configuration unset.
Note: The log file is located at /home/us1/wdmen/Pyxis_SPT/ic_projects/adk_proyectos/prng/top/default/default.netlist_transcript
Note: Netlist completed successfully.
```

Figura 3.10. Confirmación de actualización LVS válida.

3.6 Simulación estructural

Con el símbolo representativo del PRNG de la etapa anterior se ejecutó una simulación a nivel estructural, es decir a nivel transistor. En ella se apreciaron el comportamiento lógico y físico de las señales. Se empleó el entorno *Menthor Graphics* editor de esquemáticos de ADK.

En primer lugar, se instanció el símbolo representativo del PRNG llamado TOP1 y se trazaron los buses de datos para XIN, POUT, RES y CLK. Por último se colocaron los símbolos universales de VDD y GND como se muestra a continuación.

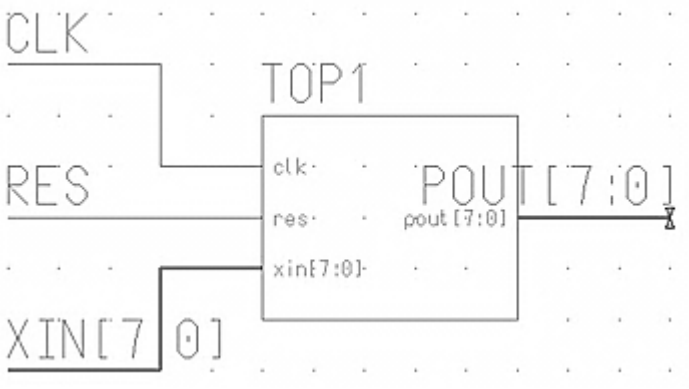


Figura 3.11. Circuito esquemático del PRNG para fines de simulación.

Una vez trazado el diagrama, se verificó su validez con la opción *check & save*. Seguidamente se invocó al modo de simulación con la opción *simulation mode*, donde se configuraron las condiciones iniciales de igual forma que la simulación comportamental.

a) **tipo de análisis:** transitorio, con tiempo de corrimiento de 450ns.

b) **tecnología empleada:** 0.35µm, electa por razones ya mencionadas previamente.

c) forzamiento de señales:

- VDD: señal de DC de 3.3 V/A por ser la tensión de trabajo de la tecnología
- GND: señal de DC de 0 V/A
- CLK: señal de reloj con periodo de 30ns, tiempo de elevación y caída de 1ns, valor en alto y bajo de 3.3 y 0 V/A respectivamente.
- RES: pulso de 3.3 V/A con un ancho de 60ns, periodo de 210ns, retardo de 30ns y un tiempo de elevación y caída de 1ns.
- XIN: señal vector de bits constante de valor $(10101010)_2 = (170)_{10}$, nivel alto y bajo de 5 y 0 V/A, a partir de 0ns.

d) Visualización de las salidas: graficar voltajes de CLK, RES, XIN y POUT.

Una vez indicadas las configuraciones anteriores, se ejecutó la simulación. Las formas de onda de las señales CLK, RES, XIN y POUT se plasman continuación.

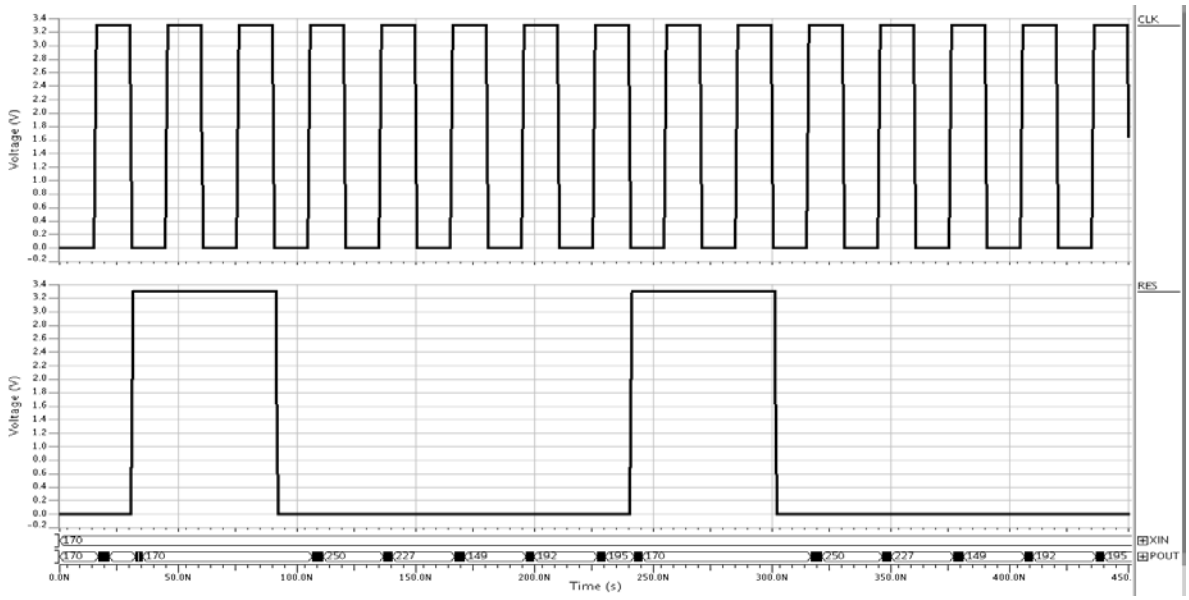


Figura 3.12. Formas de onda de las señales CLK, RES, XIN y POUT.

Los resultados son analizados en la sección 3.8 para una mejor comprensión de estos.

3.7 Generación automática del *layout*

Para esta etapa y por cuestiones del software, se extrajo una copia del PRNG esquemático para ejecutar las tareas preparatorias al *layout*. Una vez copiado el circuito, se abrió todo el conjunto de hojas esquemáticas generadas. Se verificó cada una con la opción *check & save*. Concluida la tarea se actualizaron con la opción *Update LVS*. Más tarde se procedió a generar los *viewpoints*, definidos como el conjunto de instrucciones para trazar automáticamente el circuito del PRNG a nivel físico, similar al *netlist*. Lo anterior se realizó con el ingreso de los siguientes comandos.

```
cd $MGC_WD
adk_dve prng_prelayout –t tsmc035
```

El primero fija como directorio de trabajo a $MGC_WD para hallar allí la copia del PRNG. El segundo invoca un archivo *adk_dve* el cual es un programa para crear los *viewpoints* del PRNG de la celda *prng_prelayout*. La expresión `–t tsmc035` indica la tecnología empleada, de 0.35µm.

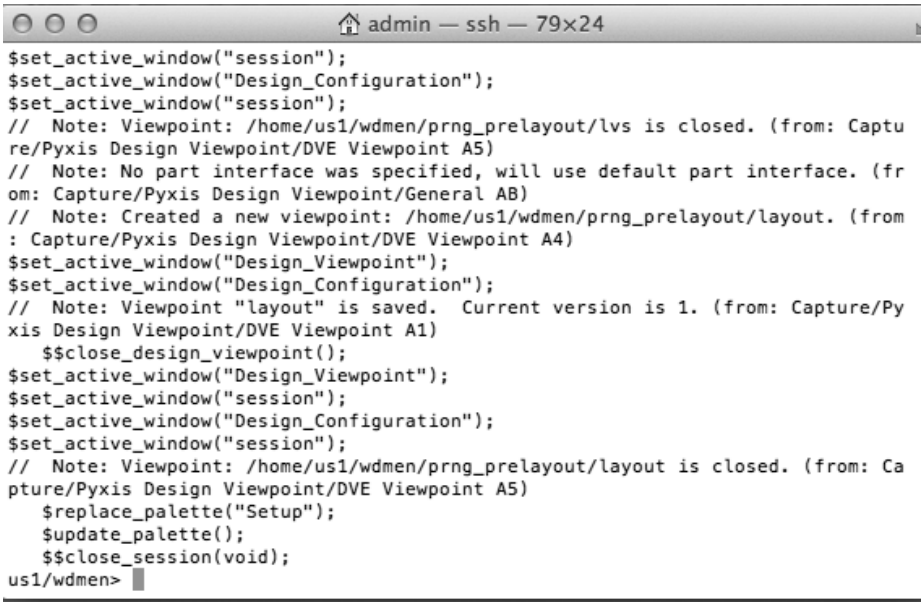


Figura 3.13. Confirmación de creación de *viewpoints*.

Los *viewpoints* fueron almacenados en una carpeta llamada *layout*, dentro de la misma celda *prng_prelayout*, como se demuestra en la siguiente figura.

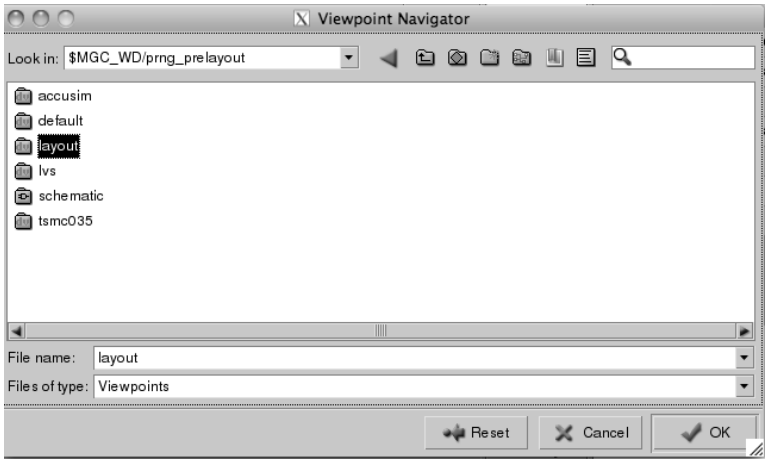


Figura 3.14. Ubicación de carpeta de *viewpoints*.

Con los *viewpoints* generados se procedió a la implementación del PRNG a nivel físico con el editor de *layout* de *Menthor Graphics* y las herramientas de ADK. Dicho entorno permitió las tareas automáticas de: trazado del área del chip, emplazamiento de componentes y su conexión. En esta última algunas conexiones de muchos cruces se ejecutaron manualmente.

Como primer paso se creó un nuevo *layout* a partir de una conectividad, es decir del circuito esquemático previo. En el cuadro de dialogo *new layout* del entorno *Menthor Graphics* se indicaron las siguientes configuraciones:

- **Ruta de acceso al *viewpoint* para ejecutar el trazado automático**
  Ruta/archivo:
  $MGC_WD/prng_prelayout/layout

- **Nombre de la celda y ubicación donde se almacenará el nuevo *layout***
  Ruta/nombre:
  $MGC_WD/prng_ASIC

- **El proceso tecnológico a emplear, de 0.35µm**

  Ruta/archivo:

  $ADK/technology/ic/process/tsmc035

- **Reglas de diseño del proceso tecnológico empleado**

  Ruta/archivo:

  $ADK/technology/ic/process/tsmc035.rules

- **Librería del proceso tecnológico empleado**

  Ruta/librería:

  $ADK/technology/ic/process/tsmc035

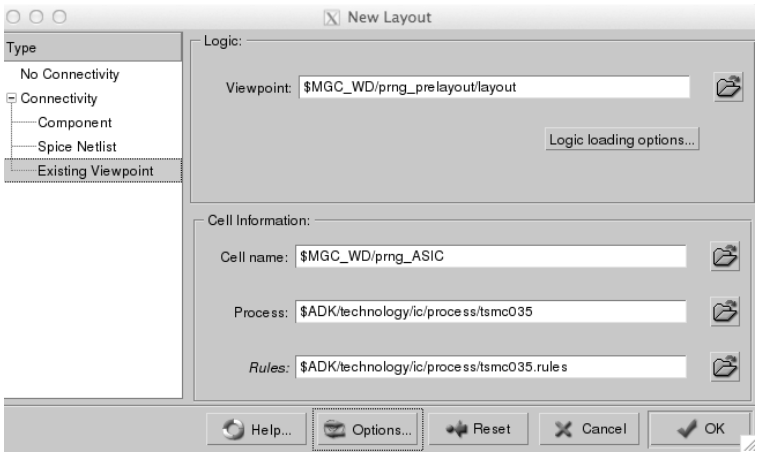


Figura 3.15. Configuraciones iniciales para la creación de un nuevo *layout* a partir de una conectividad.

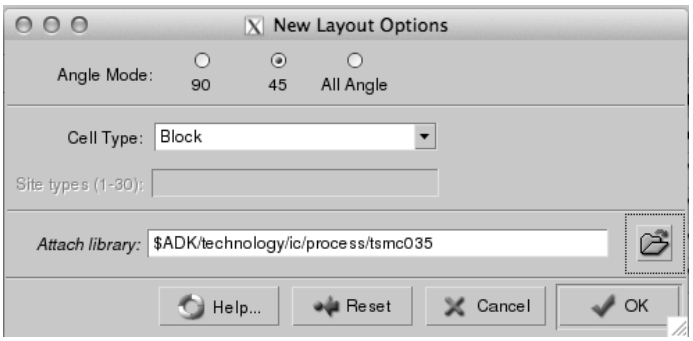


Figura 3.16. Especificación de la librería tecnológica de 0.35µm.

3.7.1 Trazado automático del área

El área del chip comprende los espacios asignados para los *pads* y el núcleo. Los *pads* son circuitos que permiten la comunicación del chip con el mundo exterior. El núcleo es el conjunto de celdas instanciadas e interconectadas, ubicado en el centro del circuito. Para su trazado automático se empleó la opción *Autofloorplan*. Se indicaron las siguientes opciones:

- **Dimensiones máximas del núcleo (*Maximum dimensions*)**

Alto: 3000µm

Ancho: 3000µm

- **Proporción de ancho/altura del núcleo (*Aspect ratio limits width/height*)**

Mínima: 0.5

Máxima: 1.5

Las dimensiones anteriores son calculadas por el software de acuerdo a la cantidad de celdas a instanciar, definida por el *viewpoint*. La proporción de ancho/altura significa un ancho mínimo de ½ y un máximo de 3/2 de la altura fijada. Esto significó un área de núcleo tendiente a una forma cuadrada, adecuada para propiciar menores retardos de propagación. Las dimensiones máximas del núcleo indican el límite máximo de longitud para el ancho y la altura, computado en 3000µm.

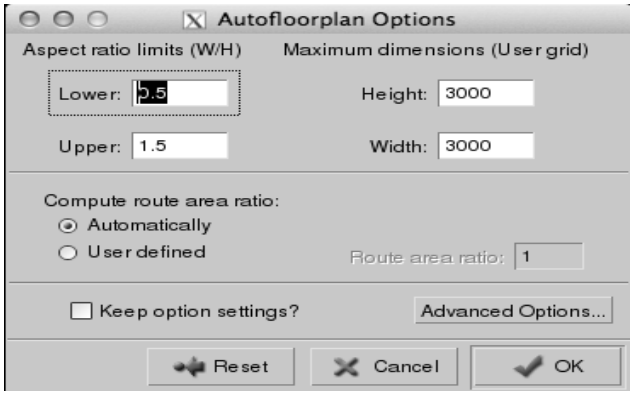


Figura 3.17. Opciones del trazado automático de área del chip.

Una vez aceptados valores, se plasmó el área del chip.

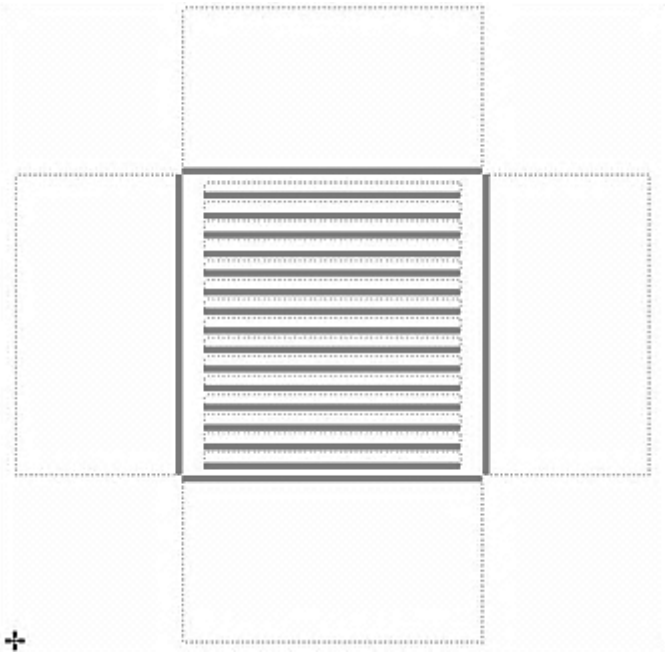


Figura 3.18. Trazado automático de área del chip.

3.7.2 Emplazamiento de celdas

Posterior al trazado del área se emplazaron automáticamente las celdas digitales a lo largo del conjunto de filas antes mencionado, con la opción *Autoplace StandartCell*. El orden para instanciar cada una consistió en colocar en forma adyacente aquellas con una conexión más próxima para así asegurar un menor número de cruces entre señales. Con base al *viewpoint* se emplearon 4 niveles metálicos, identificados por los siguientes colores y en forma ascendente: azul, violeta, gris y amarillo, siendo el azul el empleado para nodos globales, como lo son VDD y GND. Al finalizar la tarea el software mostró las conexiones entre ellas a trazar, por medio de líneas rectas de color amarillo llamadas *overflows*, como se muestra en la figura 3.19.

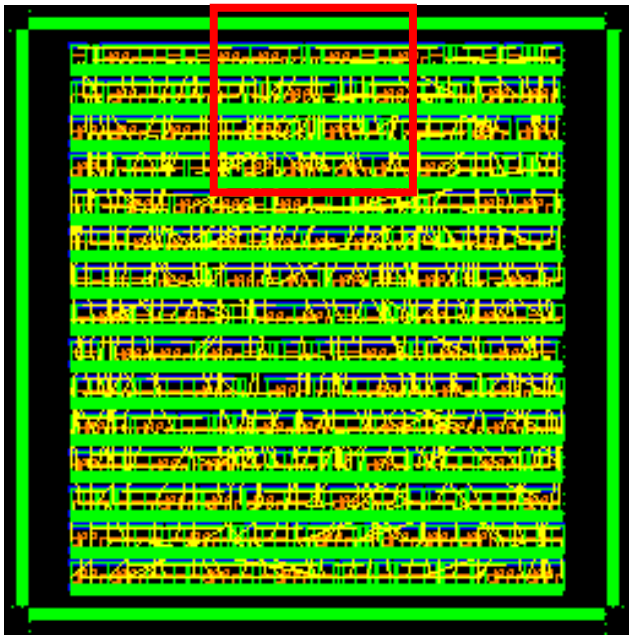


Figura 3.19. Emplazamiento de celdas del PRNG en circuito integrado.

Según se observa en la figura anterior, las celdas fueron instanciadas en un arreglo de 15 filas, conformando el núcleo del circuito. El espacio entre el contorno del núcleo y el recuadro color verde es asignado para la construcción de conexiones entre celdas y de estas hacia los *pads* de entrada y salida. De igual manera y con el mismo propósito, el software creó espacios entre filas de celdas, como se muestran en la siguiente figura.

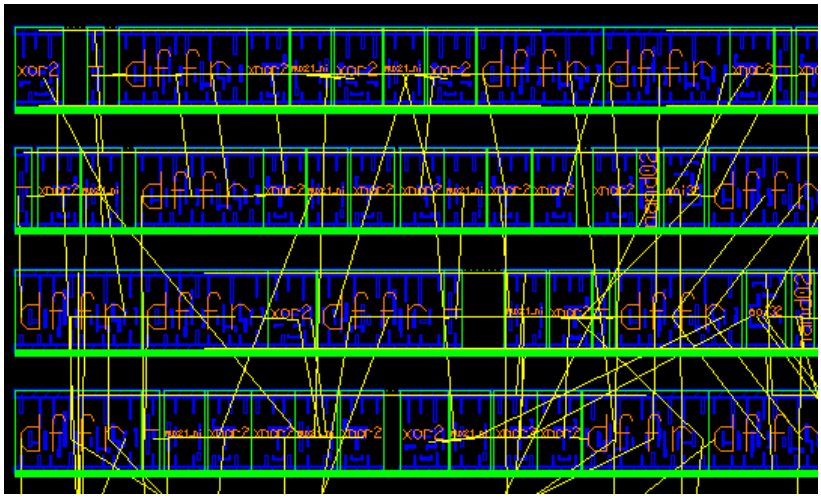


Figura 3.20. Ampliación de la zona enmarcada de la figura 3.19, con diferentes celdas colocadas a lo largo de las filas respectivas.

Acorde a la figura 3.20, también se generaron espacios laterales entre algunas celdas para mantener un núcleo de perfil cuadrado, los cuales son aprovechados para la construcción de pistas. Existen o*verflows* de longitud corta y otros que cruzan una o más filas para indicar un vínculo entre componentes, dada la extensión del circuito. No obstante todas ellas mantienen conexiones de longitud corta.

3.7.3 Emplazamiento de *pads* y puertos

Los *pads* fueron colocados alrededor del núcleo con la opción de *Autoplace ports* bajo las siguientes indicaciones y como lo demuestra la figura 3.21.

- **Método de colocación (*Placement method*):** local

- **Colocación de *pads* (*port placement*):** izquierda, derecha, abajo y arriba (*left*, *right*, *bottom* y *top*)

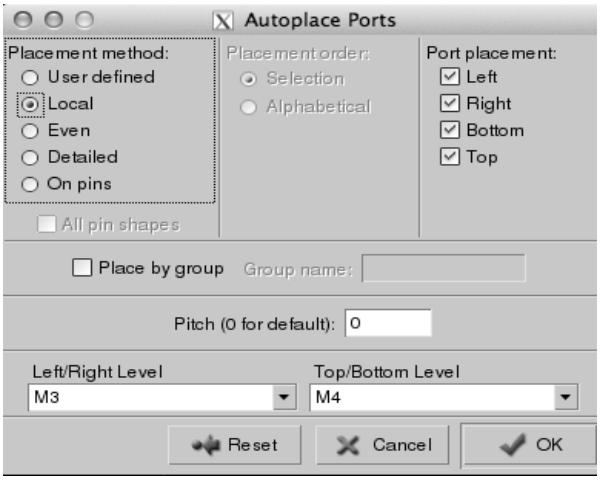


Figura 3.21. Configuraciones para el emplazamiento automático de *pads*.

De la figura 3.21, se optó por un método de colocación local para instanciarlos cerca de las zonas del núcleo donde poseían mayores conexiones. De esta forma también se redujeron los cruces entre caminos de señales. La configuración del campo *Port placement* permitió la colocación de los puertos distribuidos sobre todo el perímetro del núcleo. Los resultados de la operación anterior se muestran en la siguiente figura.

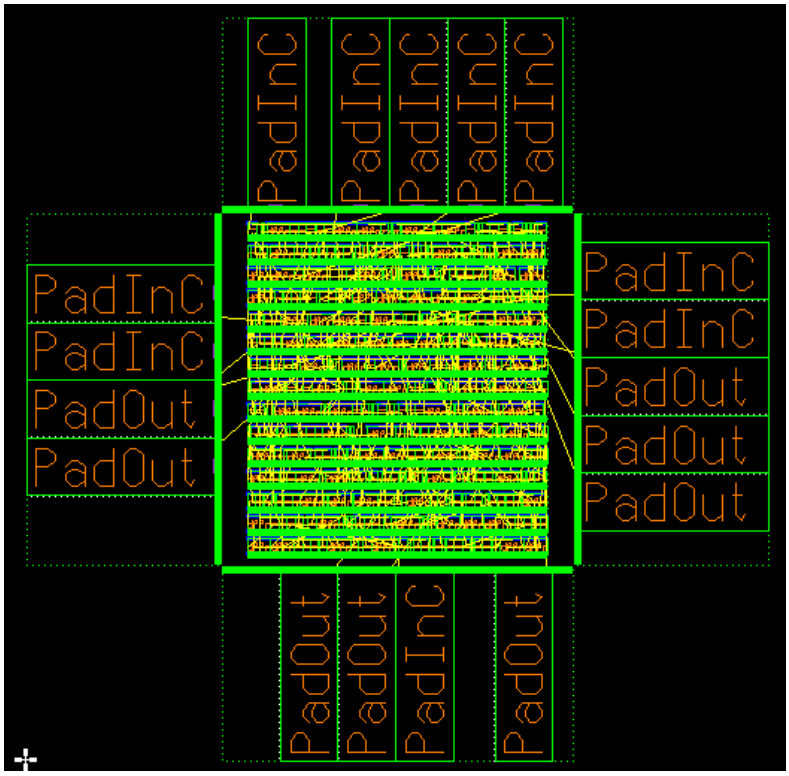


Figura 3.22. Colocación final de *pads* distribuidos alrededor del núcleo.

En la figura 3.22 se observan las escasas intersecciones entre *overflows* de los *pads* colocados, ya que estos se encuentran cerca de los conjuntos de celdas hacia donde irán conectados. Estos *overflows* son de longitud corta.

Posterior a la colocación de *pads* se procedió al emplazamiento de puertos. Dentro del mismo editor de *layout* se invocó a la hoja esquemática 1 del PRNG por ser esta la que contenía los puertos del circuito. Realizado lo anterior se empleó la opción *Port* para aparecer automáticamente cada uno de ellos con su respectiva conexión, para lo cual únicamente fueron ubicados en la región de metal 2 de su respectivo *pad*.

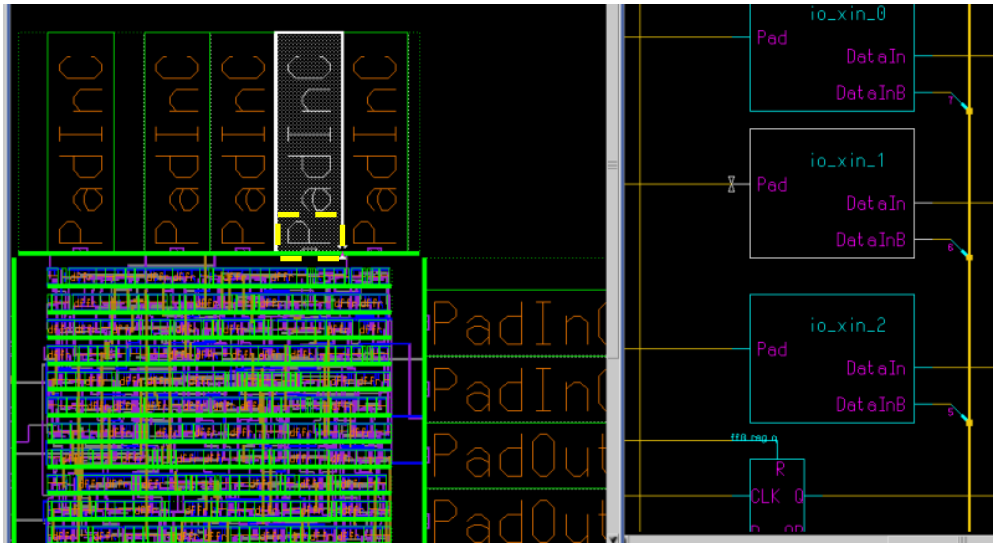
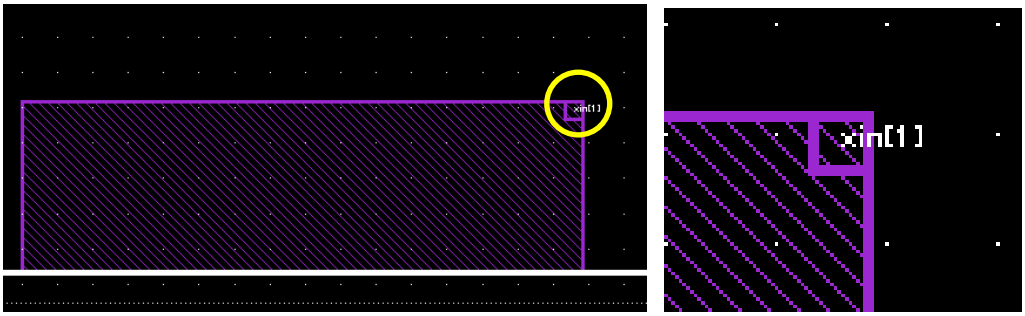


Figura 3.23. Equivalencia entre circuitos del PRNG a nivel físico (izquierda) y esquemático (derecha).

De la figura 3.23, en el circuito integrado del PRNG, el *pad* resaltado en blanco corresponde a la entrada xin[1] es decir, el bit 1 del valor semilla del PRNG. El *pad* también es resaltado en el circuito esquemático para verificar la equivalencia y detectar errores de conexión, si existen. Finalmente se agregó el nombre a los puertos con la opción *PortText*, como se muestra en la figura 3.24.



(a) **(b)**

Figura 3.24. Ampliación de la zona encerrada en líneas punteadas de la figura 3.23. (a) Región metálica del *pad* donde se colocó el puerto con su nombre respectivo. (b) Ampliación de la zona encerrada en círculo amarillo de la figura 3.23 (a).

Acorde a la figura anterior, los puertos son de nivel metálico 2, representados por el color violeta. Aunque es de área menor a la región metálica del *pad*, el software interpreta dicha región como el puerto en su totalidad, ya que son el mismo nodo.

3.7.4 Autoconexión de elementos

Una vez colocados todos los elementos necesarios sobre el área del chip se procedió al ruteo automático de celdas, *pads* y puertos. Previo al proceso se declararon las siguientes configuraciones empleando la opción *Autoroute nets/Options*.

- **Modo de operación (*Operation mode type*):** centro ponderado
- **Tamaño de paso (*Step size*):** 0.5µm
- **Orden de conexión (*Connection ordering*):** aleatorio

Lo anterior queda asentado en la siguiente figura.

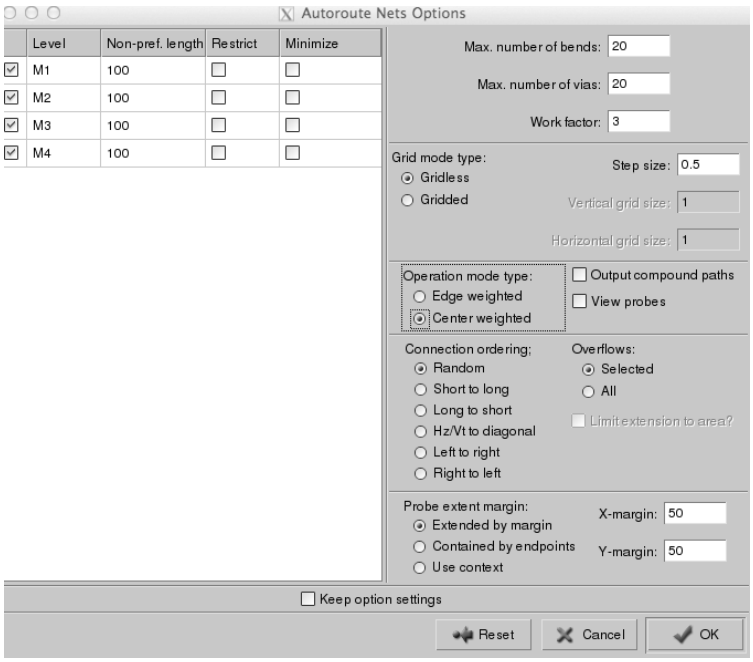
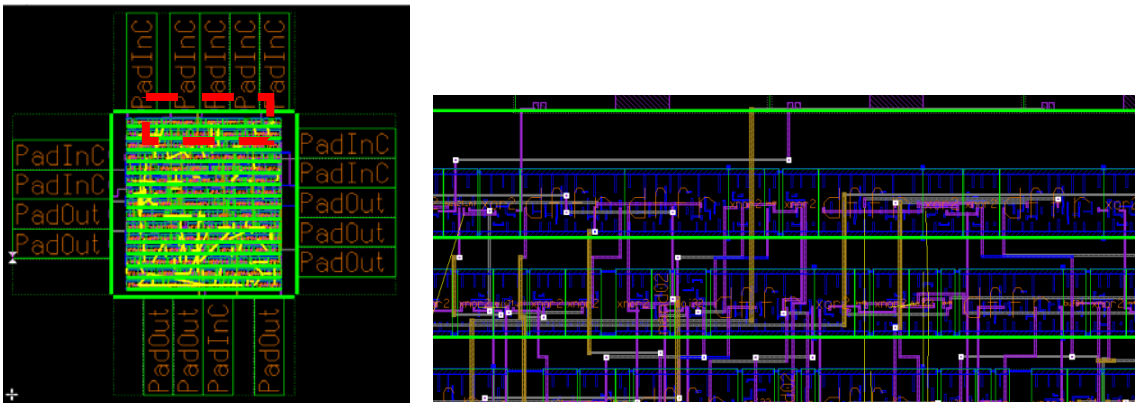


Figura 3.25. Configuraciones previas a la autoconexión de elementos.

Conforme a la figura 3.25, con la opción *Center weighted* se ubicó la mayor concentración de conexiones en el centro del chip con el fin de trazar el mayor número de pistas cortas. A esto también contribuyó la colocación de las celdas. La opción *Step size* de 0.5µm trazó pistas con longitudes múltiplos de 0.5µm para encajar mejor en los espacios asignados, sobre todo en

aquellos muy restringidos. Con la indicación de conexión tipo *random* el orden para trazar las conexiones se efectuó aleatoriamente. El resultado de estas operaciones construyó las siguientes conexiones, según se muestran en las siguientes figuras.



(a) **(b)**

Figura 3.26. Conexiones trazadas del PRNG en una primera fase. (a) Líneas amarillas resaltadas en el circuito integrado representando las conexiones faltantes u *overflows*. (b) Ampliación de la zona rectangular punteada de la figura 3.26(a), con las conexiones resultantes.

Con base en la figura 3.26(a), debido a la gran cantidad de intersecciones entre señales en algunas regiones así como la dificultad del software para trazar conexiones en espacios muy estrechos quedaron algunos *overflows* pendientes. De un total de 1760 rutas, 1646 pudieron efectuarse es decir, un 93.52% con 114 pendientes. Para terminar dicha labor se empleó la opción *autoroute/overflows*. Las indicaciones previas fueron las mismas que la del conexionado normal, descritas a continuación y mostradas en la figura 3.27.

- **Modo de operación (*Operation mode type*):** centro ponderado
- **Tamaño de paso (*Step size*):** 0.5µm
- **Orden de conexión (*Connection ordering*):** aleatorio

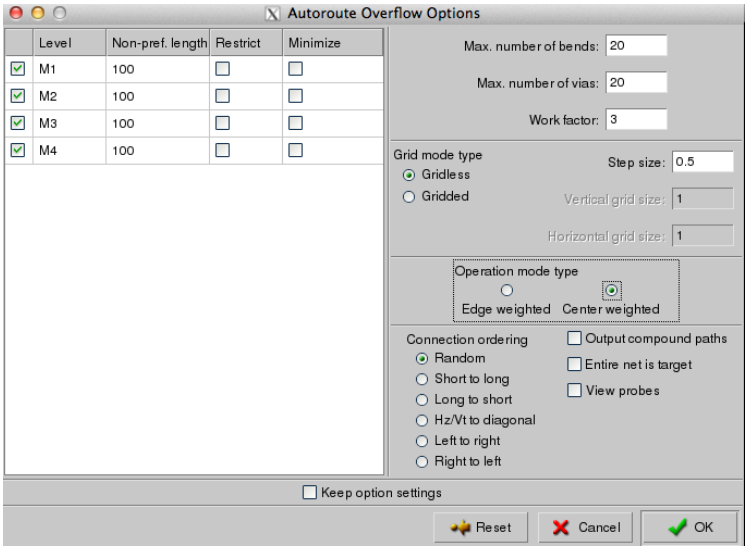
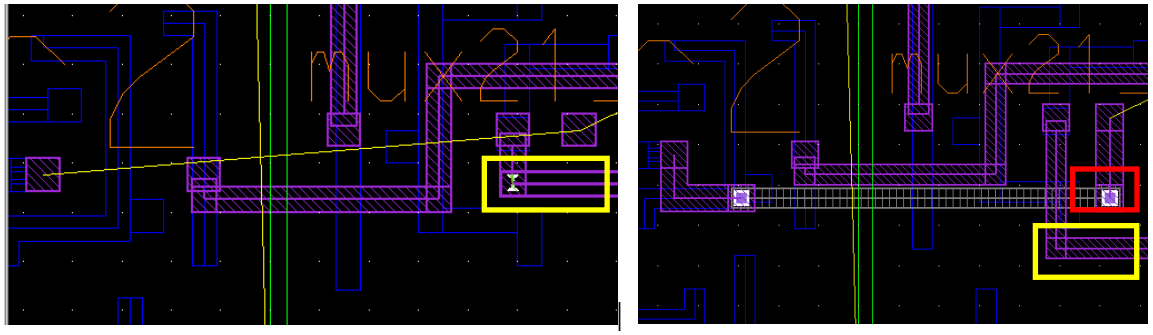


Figura 3.27. Configuraciones previas al trazado de *overflows*.

Así, las configuraciones mostradas de la figura 3.27 son idénticas a las del primer proceso de conexión. Se construyeron un total de 38 de las 114 pistas faltantes, es decir el 33.33%. Las 76 restantes fueron construidas manualmente ya que para su trazado se requirió acondicionar algunos espacios por donde pasarían, dada la gran congestión de pistas construidas tal que el software se vio incapacitado. La herramienta *Iroute* permitió tal labor ya que indica los nodos a conectar así como las direcciones y áreas por donde está permitido trazar o por donde no se puede, ya sea por violar alguna regla de diseño, por generar un corto circuito o por construir una conexión incorrecta. A continuación se presenta un caso de construcción manual de pista metálica.



(a) **(b)**

Figura 3.28. Construcción manual de una pista metálica. (a) Conexión pendiente entre dos nodos, resaltada por la línea amarilla entre ellos. (b) Conexión manual finalizada con el empleo de 2 niveles metálicos (pistas color violeta y gris) y el cambio de posición de una pista, enmarcada en color amarillo.

En la figura 3.28(a) puede observarse que el espacio para conexiones del nodo de la derecha está restringido por otras pistas y terminales de las celdas adyacentes, por lo cual surge la necesidad de ampliarlo en forma manual. Una opción fue desplazar hacia abajo la pista metálica enmarcada en color amarillo. Como lo demuestra la figura 3.28(b) fue posible la colocación de un contacto de nivel metálico 2-3, enmarcado en color rojo y a partir del cual se trazó una pista de metal nivel 3 en dirección hacia el nodo de la izquierda, con un posterior contacto nivel 2-3 y finalmente una conexión metálica nivel 2.

3.8 Análisis de resultados

La figura 3.39 expone los resultados de la simulación comportamental del PRNG.

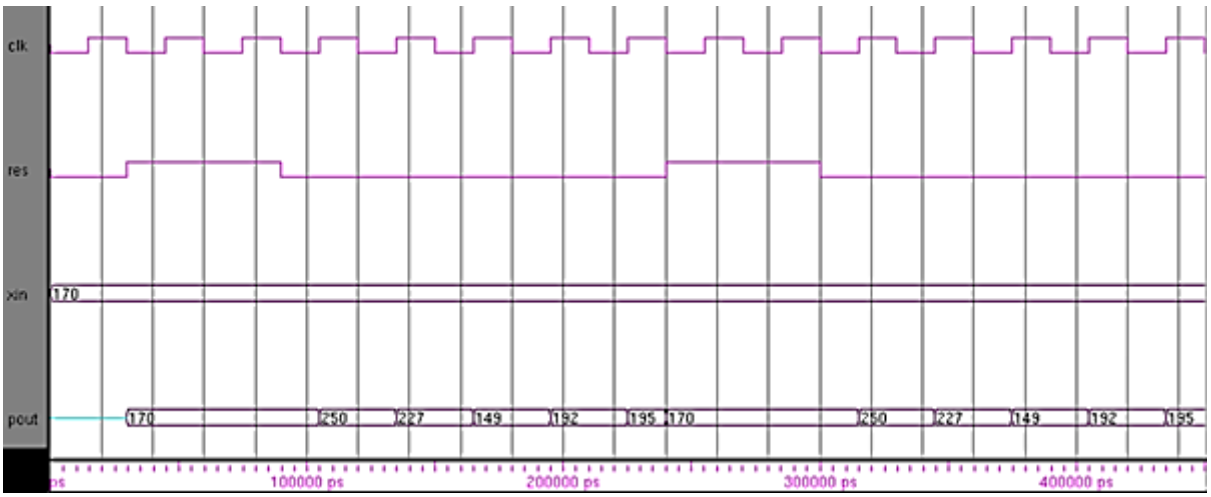


Figura 3.29. Resultados de la simulación comportamental del PRNG.

Con base a los resultados de la figura 3.29, el valor de la salida (*pout*) es la secuencia pseudoaleatoria la cual es impredecible en primera instancia, con un periodo extenso, de 0ns a 30ns (segmento de color azul) aun cuando el valor semilla (*xin*) ya ha sido establecido, el reinicio (*res*) deshabilitado y con el primer flanco de subida del reloj (*clk*) ocurrido. Lo anterior es debido a que el PRNG no posee condiciones iniciales definidas en todos los acumuladores para el modelo comportamental.

La salida copia el valor semilla en el mismo instante de la habilitación del reinicio (en 30ns) sin esperar al próximo flanco de subida, por lo cual el reinicio es asíncrono. Al deshabilitarlo en 90ns la salida retiene aun el valor semilla hasta el próximo flanco de subida, momento cuando inicia la generación de números pseudoaleatorios, en 105ns.

Durante 150ns, duración de la deshabilitación del reinicio entre 90ns y 240ns, el proceso de generación mantiene cada número el mismo tiempo de duración de un ciclo de reloj, entre flancos de subida. Al habilitar nuevamente el reinicio en 240ns, antes del próximo flanco de subida, la salida copia inmediatamente el valor semilla, confirmando una vez más el reinicio asíncrono. El tiempo de duración del segundo reinicio es el mismo que el primero. Al finalizar, el segundo proceso de generación de números comienza a repetirse en la misma forma y con la misma secuencia que el primer proceso. Lo anterior confirma la correcta funcionalidad del PRNG y la naturaleza repetitiva del PRNG después de un reinicio, importante en aplicaciones como la criptografía [2].

En la ejecución de la síntesis digital se generó un reporte de retardos. Éste consiste en un listado de diferentes tiempos en que la señal de salida arriba a su puerto respectivo. El sintetizador diseña diferentes caminos o rutas críticas (*critical path*) por las cuales se presentan los mayores retardos. Estas rutas se presentan en el reporte como una lista de nodos de salida de algunas compuertas del circuito por las cuales se propaga la señal. En cada una se registra el retardo de la salida y el tiempo acumulado por compuertas anteriores. El cálculo finaliza con la propagación de la señal por el puerto de salida a través de todos los nodos de la lista. El reporte creado consta de 10 tiempos diferentes de arribo, cada uno originado por un camino critico específico. Dada la gran extensión de la información (ver anexo B), en esta sección solo se presenta el caso para la primera ruta crítica, mostrada en la figura 3.30.

```
                    Critical Path Report

Critical path #1, (unconstrained path)
NAME                              GATE          ARRIVAL              LOAD
-------------------------------------------------------------------------
clock information not specified
delay thru clock network                        0.00 (ideal)


acu0_reg1_reg_q(0)/Q              dffr      0.00  0.54 dn            0.03
ix688/Y                           xnor2     0.20  0.74 up            0.03
ix45/Y                            inv01     0.11  0.85 dn            0.01
ix614/Y                           aoi32     0.52  1.38 up            0.04
ix724/Y                           xnor2     0.36  1.74 dn            0.05
ix163/Y                           xnor2     0.31  2.04 up            0.04
ix393/Y                           xnor2     0.28  2.33 dn            0.05
ix816/Y                           xnor2     0.31  2.64 up            0.04
ix588/Y                           mux21_ni  0.50  3.13 dn            0.04
ix586/Y                           mux21_ni  0.41  3.54 dn            0.04
ix856/Y                           xnor2     0.28  3.83 up            0.05
ix399/Y                           xnor2     0.26  4.08 dn            0.04
ix854/Y                           mux21_ni  0.51  4.59 up            0.03
ix409/Y                           xnor2     0.31  4.90 dn            0.05
ix971/Y                           xnor2     0.29  5.20 up            0.04
ix558/Y                           mux21_ni  0.50  5.70 dn            0.04
ix990/Y                           xnor2     0.28  5.98 up            0.05
ix635/Y                           xnor2     0.26  6.24 dn            0.04
ix865/Y                           xnor2     0.31  6.55 up            0.05
ix1100/Y                          xnor2     0.26  6.81 dn            0.04
ix524/Y                           mux21_ni  0.53  7.34 up            0.04
ix1116/Y                          xnor2     0.30  7.64 dn            0.05
ix1085/Y                          xnor2     0.29  7.93 up            0.04
ix1114/Y                          mux21_ni  0.49  8.42 dn            0.04
ix1095/Y                          xnor2     0.18  8.61 up            0.01
ix1240/Y                          buf04     0.50  9.10 up            0.30
io_pout(7)/Pad                    PadOut    1.56  10.66 up           0.00
pout(7)/                                    0.00  10.66 up           0.00
data arrival time                                 10.66

data required time                          not specified
-------------------------------------------------------------------------
data required time                          not specified
data arrival time                             10.66
                                            ----------
                                          unconstrained path
-------------------------------------------------------------------------
```

Figura 3.30. Reporte de retardos de propagación (hoja 1 de 10).

En la figura 3.30, de izquierda a derecha, se presentan tres columnas importantes; la primera (NAME) enlista los nodos de salida específicos por donde se propaga la señal, la segunda (GATE) indica las celdas digitales a las que pertenecen tales nodos y la tercera (ARRIVAL) se compone de dos datos, el retardo individual por compuerta (columna izquierda) y el retardo acumulado por propagaciones anteriores (columna derecha).

A manera de resumen, la tabla 3.3 enlista los diferentes tiempos de arribo de la señal de salida para cada camino crítico trazado por el software, dada la gran extensión del reporte de retardos.

Tabla 3.3. Tiempos de arribo de la señal de salida para cada camino crítico trazado por el software.

Camino critico	Tiempo de arribo de la señal de salida (ns)
1	10.66
2	10.66
3	10.65
4	10.63
5	10.62
6	10.61
7	10.52
8	10.52
9	10.51
10	10.47
Promedio	10.585

De acuerdo a los datos de la tabla 3.3, los componentes que generan mayores retardos son el multiplexor (mux21_ni) con un promedio de 0.49ns y el *pad* de salida (PadOut) de 1.56ns, lo cual se explica por la gran cantidad de transistores empleados para su construcción. El elemento con el menor retardo es el inversor (inv01) de 0.11ns, siendo la celda más pequeña. Conforme a la tabla anterior, el máximo de los tiempos de arribo de dato de salida es de 10.66ns. Esto garantiza tiempos menores al valor mencionado para la propagación de señales por rutas de menor complejidad.

También en el proceso de síntesis se creó un reporte de área, con el listado del tipo y la cantidad de celdas a emplear para el PRNG.

```
************************************************************

Cell: top    View: INTERFACE    Library: work

************************************************************


 Cell       Library  References     Total Area

PadInC      tsmc035_typ    10 x
PadOut      tsmc035_typ     8 x
aoi22       tsmc035_typ     5 x       1      7 gates
aoi32       tsmc035_typ     9 x       2     16 gates
buf04       tsmc035_typ     8 x       1     10 gates
dffr        tsmc035_typ    88 x       6    491 gates
inv01       tsmc035_typ    69 x       1     52 gates
inv02       tsmc035_typ    30 x       1     23 gates
mux21       tsmc035_typ     4 x       2      7 gates
mux21_ni    tsmc035_typ    52 x       2     94 gates
nand02      tsmc035_typ     9 x       1      9 gates
xnor2       tsmc035_typ   132 x       2    252 gates
xor2        tsmc035_typ    14 x       2     30 gates

 Number of ports :                      18
 Number of nets :                      449
 Number of instances :                 438
 Number of references to this view :     0

Total accumulated area :
 Number of gates :                     991
 Number of accumulated instances :     438
Info, Command 'report_area' finished successfully
```

Figura 3.31. Reporte de celdas empleadas.

El número total de celdas es de 438. La compuerta xnor2 presenta el mayor número de instancias con 132, representando el 30.13% del total, después le sigue el flip - flop D (dffr) con 88, siendo el 20.09%, de tal modo que aproximadamente el 50% de la arquitectura está constituida por dichas celdas, esto se debe por la implementación de funciones como la suma y retención de datos, predominantes en el PRNG en cuestión. El elemento con el menor número de instancias es el mux21, de solo 4, con el 0.91%.

Las celdas listadas anteriormente formaron el circuito PRNG a nivel esquemático conformado por 6 hojas. Dada su gran extensión, únicamente se presentará la hoja correspondiente al circuito con los *pads* de salida.

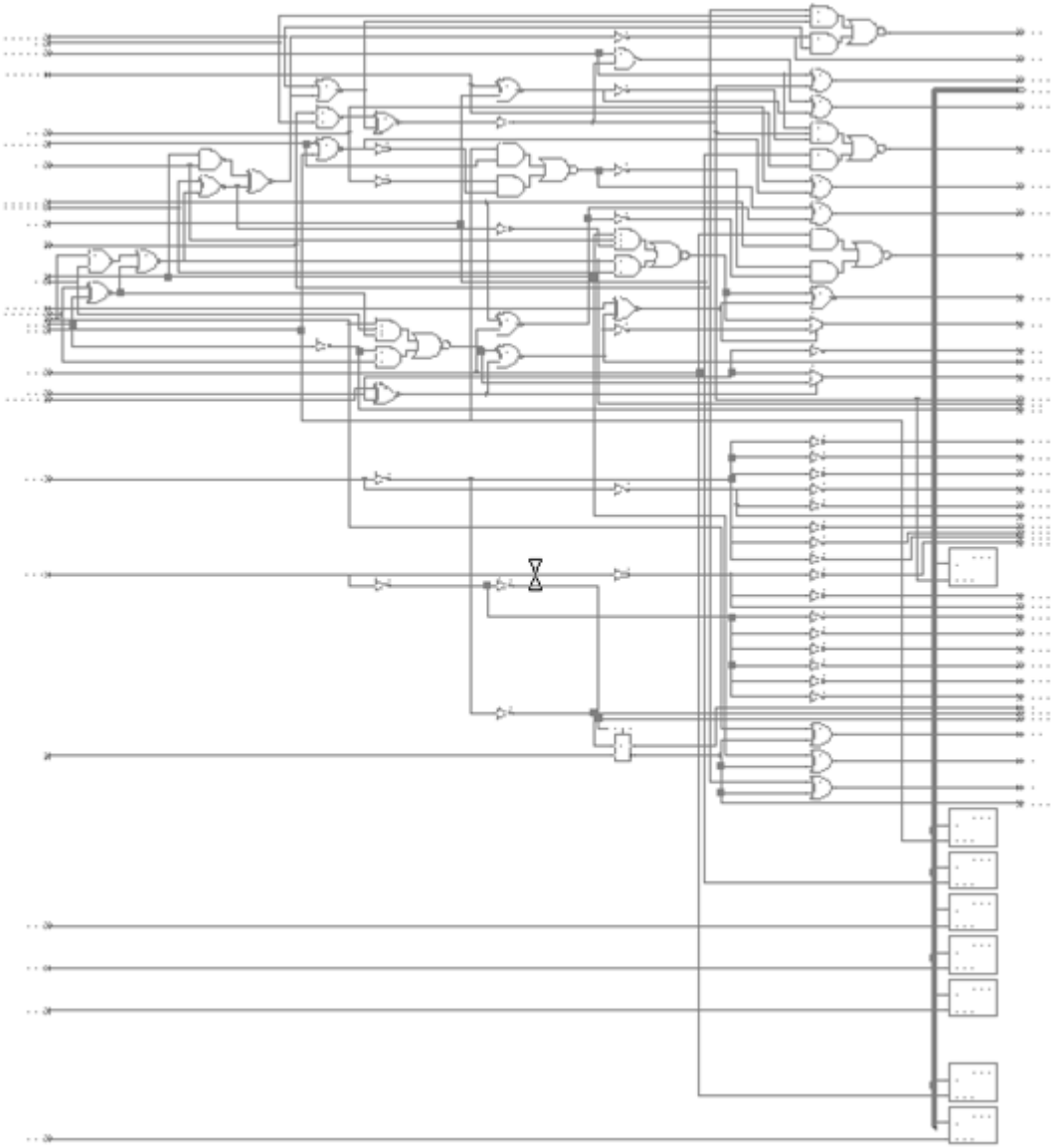


Figura 3.32. Hoja 6/6 correspondiente a los *pads* y puertos de salida del PRNG.

La simulación estructural arrojó los resultados que se muestran en la figura 3.33.

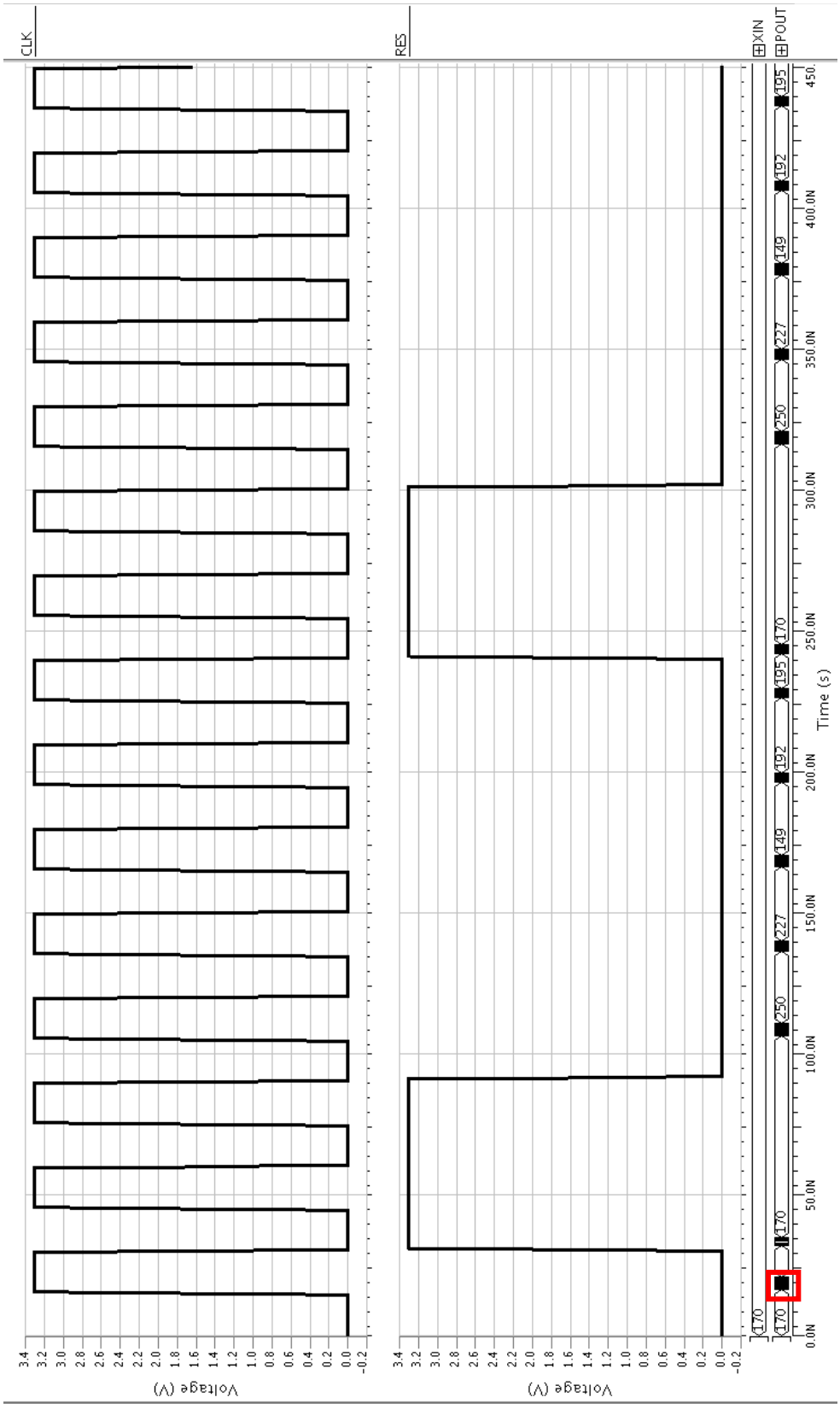


Figura 3.33. Resultados de la simulación estructural del PRNG.

Durante los flancos de subida de la señal *clk* o *res*, al estar habilitado el proceso de generación de números pseudoaleatorios, en la salida se crearon un conjunto de oscilaciones asíncronas (recuadro rojo de la figura 3.33) de una duración promedio de 4ns, antes de estabilizarse a un valor concreto. Aunque no son parte del comportamiento lógico, eran de esperarse debido a las múltiples celdas que conforman las 9 etapas de acumuladores, lo cual se traduce en un aumento del retardo de propagación de la salida y un mayor tiempo para estabilizarse.

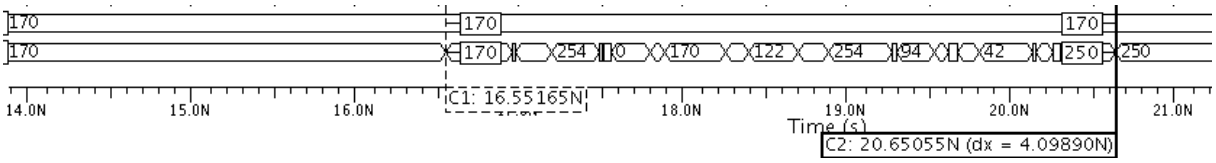


Figura 3.34. Duración de tiempo del primer conjunto de oscilaciones asíncronas del PRNG, de aproximadamente 4ns.

De la imagen anterior, el software interpreta dichas oscilaciones del orden de picosegundos como valores generados, pero en realidad no lo son. Dicho fenómeno, al producirse durante y después del flanco de subida de la señal *clk* o *res*, resta tiempo al periodo de cada número pseudoaleatorio. No obstante, puede ser despreciable si el periodo de la señal de reloj es mayor o igual a 10 veces la duración promedio de dichas señales asíncronas. Para ello y en base a los datos de la simulación, este periodo de reloj es de 40ns como mínimo, es decir 25MHz como frecuencia máxima para tales fines.

Al considerar nuevamente las oscilaciones asíncronas de la figura 3.34 es posible definir un rango de valores para el periodo de la señal de reloj, aunque en algunos casos no implique volver despreciables tales oscilaciones. Son dos las condiciones que debe cumplir la señal de reloj para la emisión de un nuevo valor pseudoaleatorio: el flanco ascendente y posteriormente retener el nivel lógico 1 una duración de tiempo mayor al periodo de oscilaciones asíncronas. Por lo cual la duración de tiempo del reloj en 1 lógico debe ser mayor a 4ns y aunado a que los tiempos en alto y bajo son iguales entonces se deduce que el periodo de tal señal debe ser mayor a 8ns. A su vez también implica un rango de frecuencias de reloj y por lo tanto de velocidades del PRNG, siendo menor a 125MHz.

A continuación se presenta una comparación de resultados de las simulaciones comportamental y estructural.

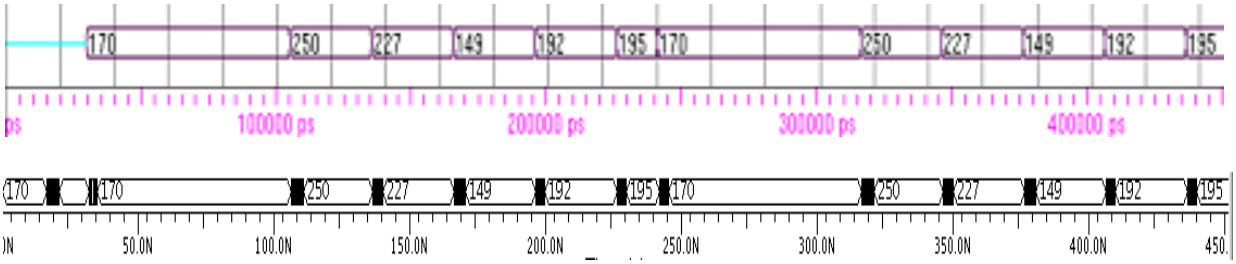


Figura 3.35. Secuencia de números pseudoaleatorios generada de la simulación: comportamental (superior) y estructural (inferior).

Las secuencias anteriores son idénticas, de donde se deduce que las oscilaciones asíncronas no afectan el comportamiento del circuito. Durante los primeros 30ns, en la simulación estructural sí existen valores definidos en la salida, el primero de ellos es el valor semilla de donde se genera el resto de la cadena, situación no ocurrida en la comportamental. Con lo anterior se comprueba la correcta funcionalidad del PRNG a nivel de celdas y demuestra la correcta ejecución del proceso de síntesis.

Finalmente el diseño físico del PRNG se presenta a continuación, bajo las siguientes dimensiones y características.

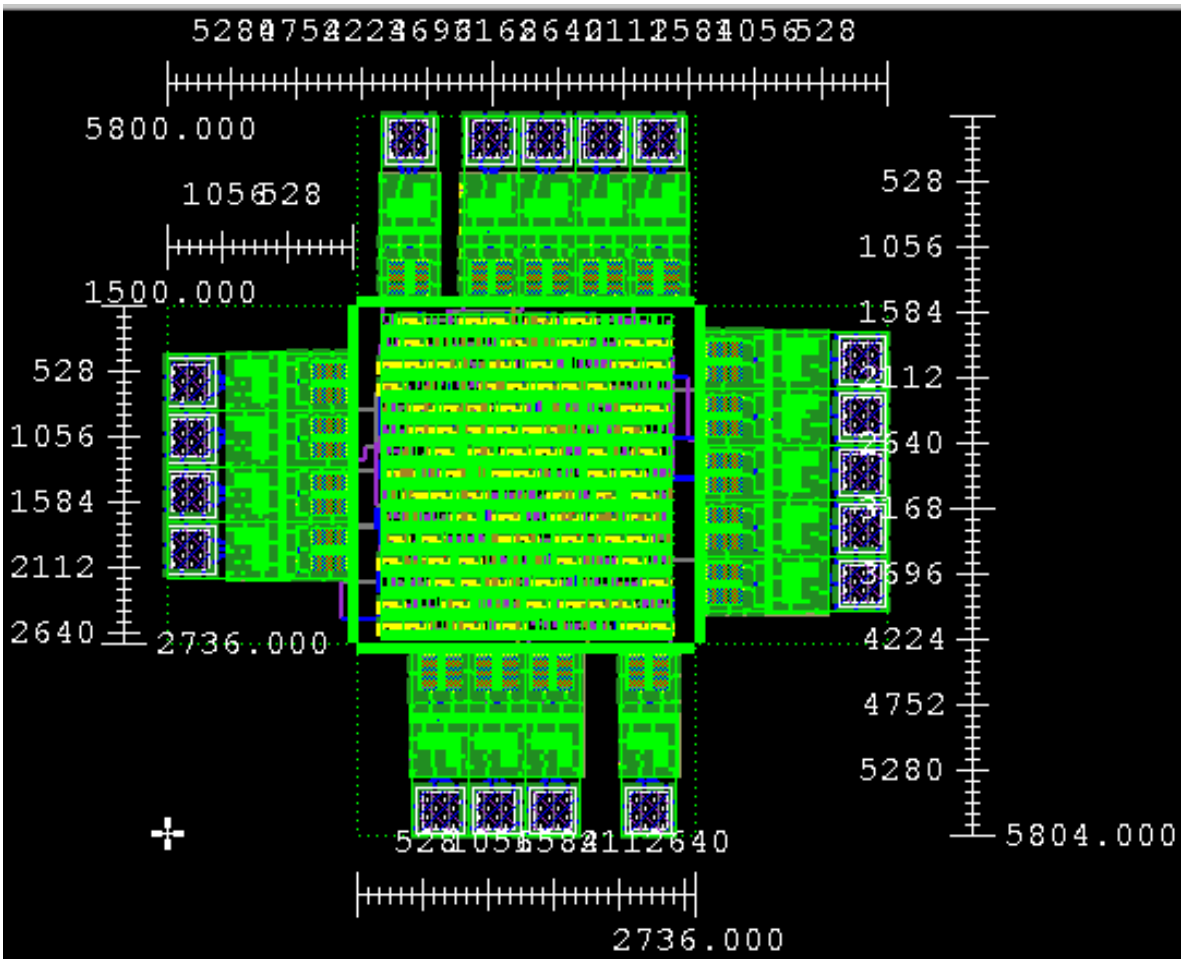


Figura 3.36. Circuito integrado final del PRNG (Escala: 1µm/div).

Con base en la figura 3.36, la altura y anchura del circuito integrado, incluyendo los *pads*, son de 5.804mm y 5.800mm respectivamente, logrando una proporción más cercana a la cuadrada que a una rectangular. Las zonas de color morado representan las terminales de conexión de los *pads* hacia las del encapsulado, tal que son de mayor robustez para lograr una conexión resistente. La estructura física de los *pads* es analizada con mayores detalles en la figura 3.38.

Una ampliación del núcleo del circuito se presenta en la siguiente figura.

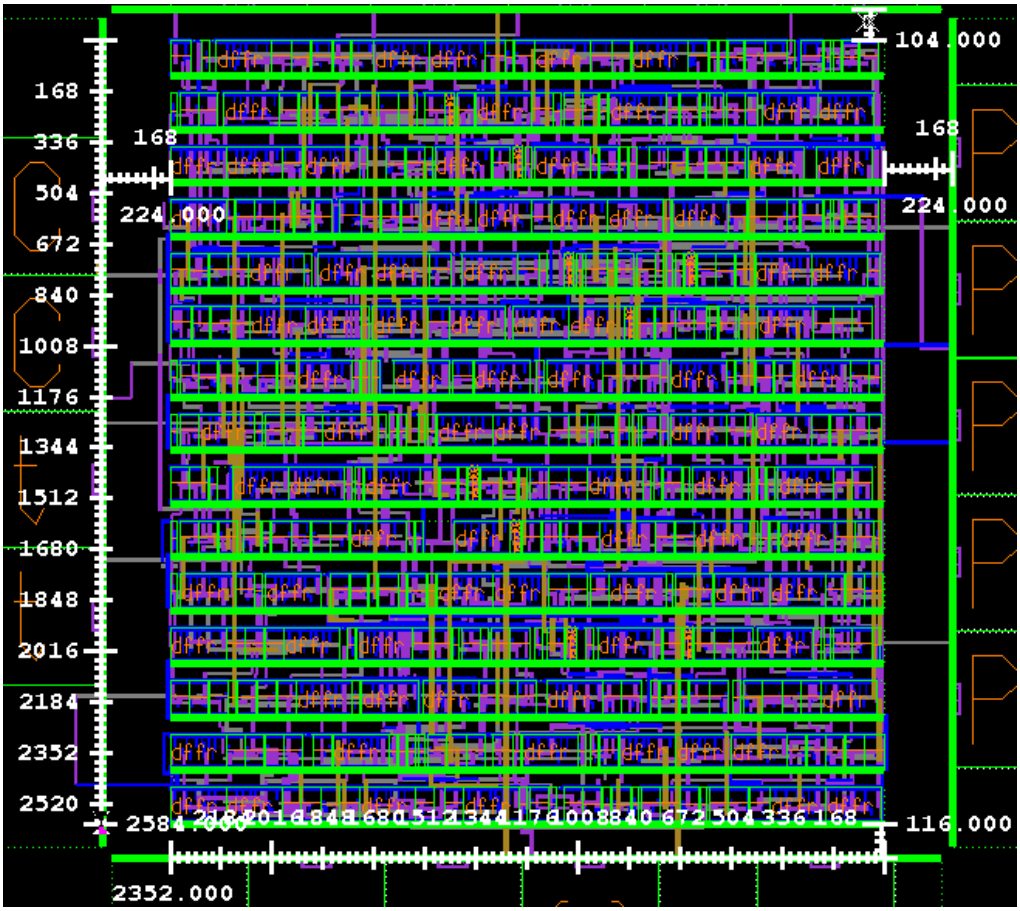


Figura 3.37. Núcleo del circuito integrado (Escala: 1µm/div).

Según la figura 3.37, la altura y ancho del núcleo son de 2,520µm y 2,352µm respectivamente. Los espacios entre el núcleo y los *pads* son: laterales de 224µm, superior de 104µm e inferior de 116µm. Estos espacios fueron empleados para construir pistas entre celdas y de estas a los *pads*. Debido a la forma de colocación de estos últimos, las conexiones de estos con las celdas son de longitudes cortas y se intersecan en menor cantidad. Los espacios entre núcleo y *pads* también son empleados para la construcción de conexiones globales, como son el VDD (voltaje de alimentación) y GND (tierra), ya que son comunes a todas las celdas y requieren estar distribuidos a lo largo y ancho del núcleo.

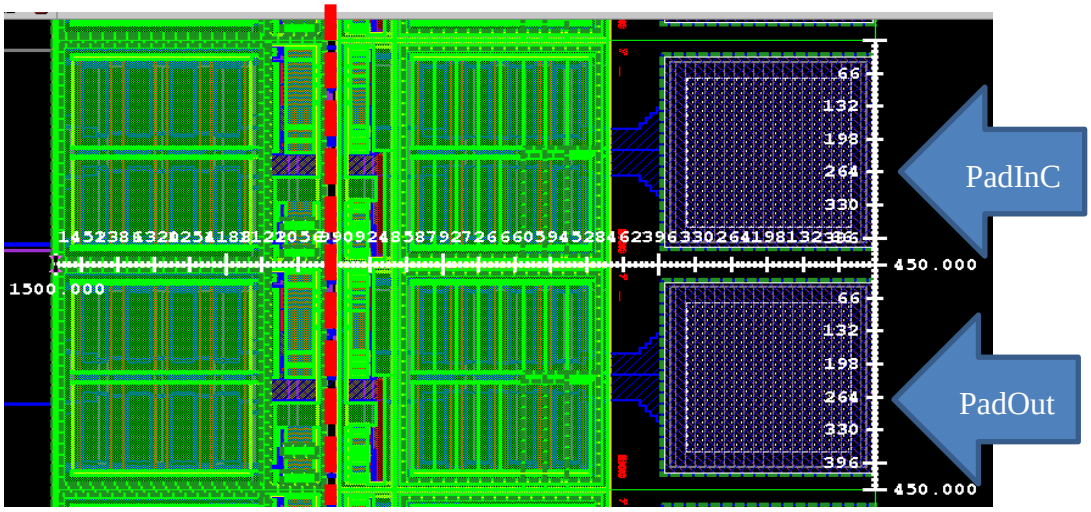


Figura 3.38. ***Pad*** **de entrada y salida (Escala: 1µm/div).**

La figura 3.38 muestra un circuito *pad* de entrada y uno de salida. Las regiones de color morado son áreas metalizadas donde se crearán las conexiones hacia el encapsulado. Sobre la región verde, del lado izquierdo de la línea roja punteada se encuentran los *buffers* los cuales acondicionan la señal a transmitir y del lado derecho se ubican los diodos de protección antiestática con el objetivo de reducir los efectos de posibles descargas.

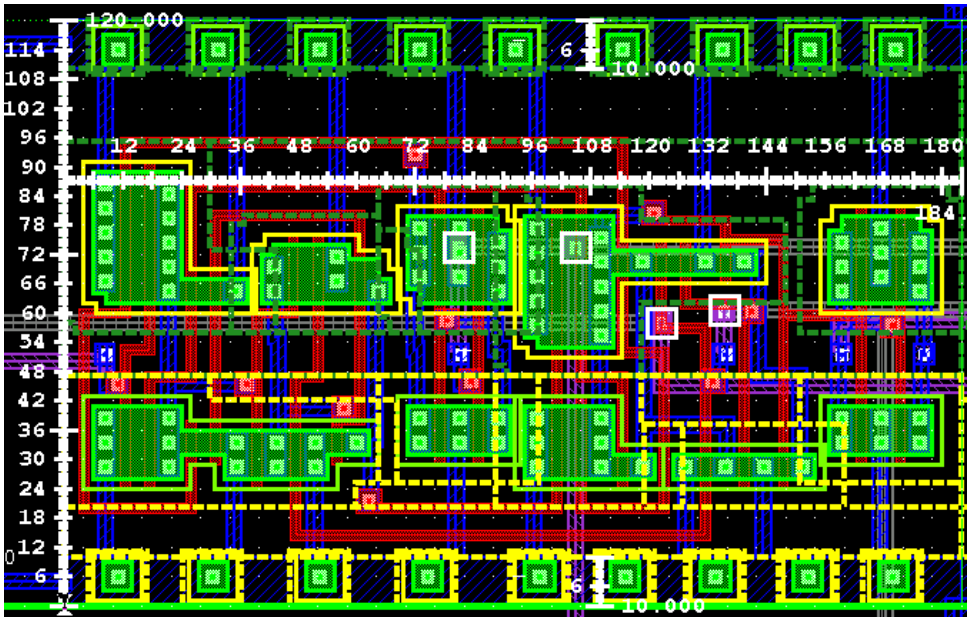


Figura 3.39. Medidas de la celda dffr (flip - flop D) con la cualidad de ser la de mayor area. (Escala: 1µm/div).

La figura 3.39 muestra el diseño físico de un flip – flop D, con un ancho de 184µm y una altura de 120µm. Esta última siempre es la misma para todas las celdas, ya que pertenecen al mismo kit de diseño y la finalidad es permitir la perfecta alineación horizontal de las celdas a lo largo de las filas en el circuito. Los transistores son representados por las regiones color verde (excepto los cuadros en los extremos superior e inferior). Esta celda emplea 34 transistores, 17 PMOS ubicados dentro de los contornos punteados verdes que simbolizan los perímetros de los pozos tipo N. De igual manera son 17 NMOS dentro de los contornos punteados amarillos que delimitan los pozos tipo P. Las pistas color rojo son conexiones de polisilicio y los cuadros verdes alineados y colocados a lo largo de los extremos superior e inferior de la celda corresponden a contactos de VDD y GND para su conexión con otros nodos. La nomenclatura de colores anterior se aplica para el resto de las celdas que a continuación se presentan.

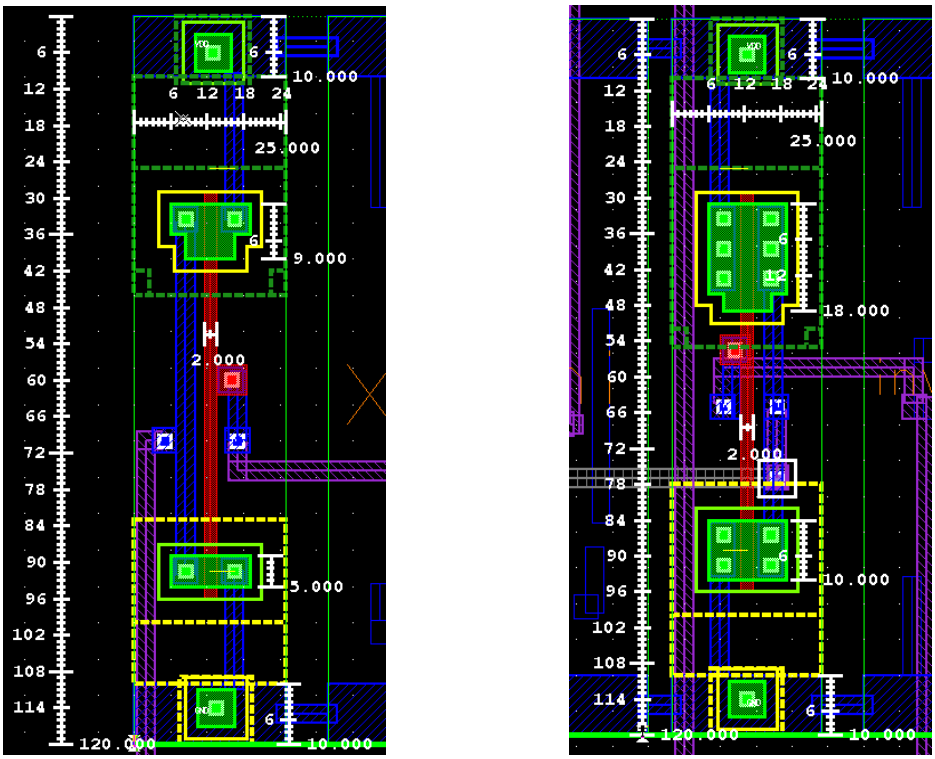


Figura 3.40. Medidas de los inversores tipo 1 (izquierda) y tipo 2 (derecha), con las cualidades de ser las de menor area (Escala: 1µm/div).

La figura 3.40 muestra el layout de dos inversores, ambos con un ancho de 25µm, cada uno con un transistor PMOS y un NMOS. La diferencia radica en el ancho de canal de los transistores que los conforman, donde es mayor para el inversor tipo 2, de 18µm para el PMOS y 10µm para el NMOS. El aumento del ancho de canal implica una menor resistencia en los transistores, lo cual ocasiona tiempos de retardo menores. Por lo cual el inversor tipo 2 es más veloz que el tipo 1.

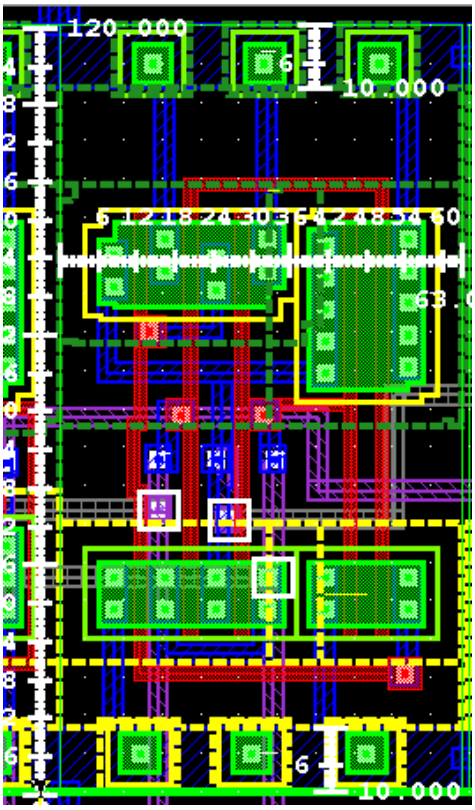


Figura 3.41. Medidas de la compuerta xnor02, siendo la de mayor cantidad de instancias en el circuito (Escala: 1µm/div).

La figura 3.41 muestra el layout de una compuerta XNOR de dos entradas, con 5 transistores tipo PMOS y 5 NMOS. Algunos de ellos se sobreponen en las terminales drain para lograr la conexión requerida y ahorrar la construcción extra de una pista metálica. Es por ello que en apariencia se observan solo 2 transistores de cada tipo.

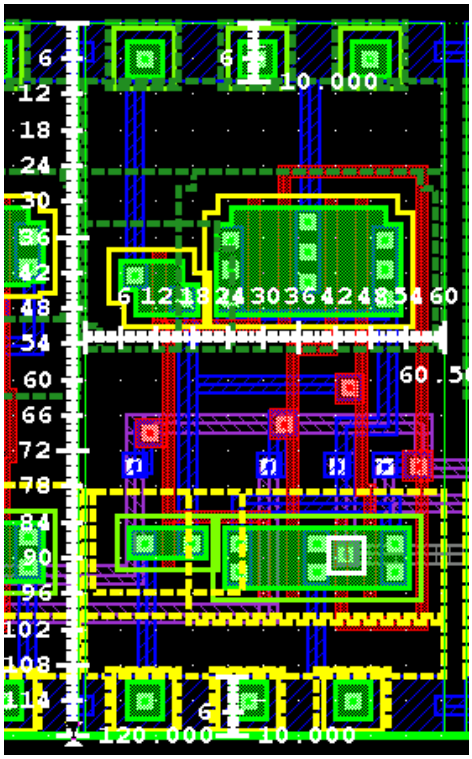


Figura 3.42. Medidas del multiplexor mux21, siendo la de menor cantidad de instancias en el circuito (Escala: 1µm/div).

La figura 3.42 muestra el layout de un multiplexor de dos entradas, con 5 transistores tipo PMOS y 5 NMOS. Los nodos de VDD y GND se ubican en los extremos superior e inferior de la celda y son regiones metálicas de nivel 1 (color azul) a lo ancho de las celdas. Su altura es de 10µm y siempre es la misma para todas las celdas anteriores. La razón es la misma que la de la altura de las celdas, mencionada anteriormente.

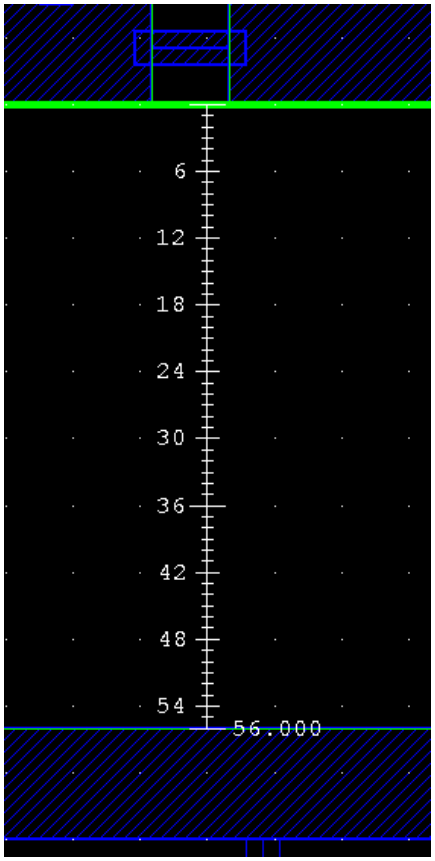


Figura 3.43. Separación entre filas de celdas (Escala: 1µm/div).

La figura 3.43 expone la distancia de separación entre filas de celdas, de 56µm. Esta magnitud es calculada por el software durante el trazado automático del área del chip y depende del número de celdas a emplear y las conexiones a construir. El valor de tal separación puede modificarse si es requerido. Al considera que el ancho promedio de cada pista es de 3µm y el distanciamiento mínimo entre aquellas del mismo nivel metálico es de 6µm entonces es posible la construcción máxima de 6 pistas del mismo nivel metálico dentro del espaciado entre filas de celdas, suficiente para mantener un compromiso entre la reducción del congestionamiento de conexiones y el menor tamaño posible del circuito integrado.

La altura y ancho del núcleo del circuito son de 2,520µm y 2,352 µm respectivamente, esto originó un área de 5, 927, 040 μm^2 = 5.92704 mm^2. Con base en el reporte de área y en las medidas de las celdas anteriores, se realiza el siguiente análisis.

Tabla 3.4. Análisis de elementos implementados y de área empleada para el PRNG en ASIC.

Celda	**Área por unidad (μm^2)**	**Número de instancias en el circuito**	**Área por instancias en el circuito (μm^2)**	**Porcentaje de instancias en el circuito (%)**	**Porcentaje de área por instancias en el circuito (%)**
flip – flop D (dffr)	22,080	88	1,943,040	20.09	32.78
inversor tipo 1 (inv01)	3000	69	207,000	15.75	3.49
inversor tipo 2 (inv02)	3000	30	90,000	6.84	1.51
compuerta xnor (xnor2)	7560	132	997,920	30.13	16.83
multiplexor (mux21)	7260	4	29,040	0.91	0.48
TOTAL	42,900	323	3,267,000	74	55

Las celdas con mayor porcentaje de ocupación de área por instancias son el *flip - flop D* (dffr) de 32.78% seguido de la compuerta *xnor* (xnor2) de 16.83%. En conjunto emplean el 49.61% del área total, aproximadamente el 50%. Tal porcentaje también coincide con el de instancias en el circuito por parte de ambas celdas. Aunque no necesariamente es así, pues la suma de los porcentajes de instancias de inv01 e inv02 es de 22.59%, cuando en área emplean solo el 5%. De igual forma ocurre con el *flip – flop D* y la compuerta *xnor*, con un mayor porcentaje de unidades para el segundo pero una mayor ocupación de superficie para el primero. El conjunto de multiplexores, por ser los menos empleados, ocupan la menor área de todas.

3.9 Resumen

Debido a las prestaciones lógicas y de recursos consumidos, se implementó el PRNG enfatizado en el capítulo 2 en un ASIC para ejemplificar el flujo de diseño de dichos circuitos, importantes hoy día con aplicaciones en telefonía, telecomunicaciones, computadoras, cámaras fotográficas, automóviles, entre otras. Fueron cinco los pasos ejecutados, los cuales fueron: descripción y simulación en HDL, síntesis digital, generación automática del circuito esquemático, simulación estructural y generación automática del *layout*, todos ellos desarrollados con el paquete de software *Menthor Graphics* y el Kit de Diseño de ASIC (ADK). En el primero se empleó *VSIM* para describir el PRNG en lenguaje VerilogA con un estilo comportamental. Posteriormente se compiló y simuló exitosamente. Para el proceso de síntesis se utilizó *Leonardo Spectrum* donde se generó un archivo *netlist* para importarlo al editor de esquemáticos de ADK. En él se trazó automáticamente el PRNG a nivel estructural con un proceso tecnológico de 0.35µm y se inició una simulación. Una vez comprobada su funcionalidad se crearon los *viewpoints*, archivo necesario para la creación automática del *layout*. Dicho archivo se importó al editor de *layout* de ADK, donde se realizaron automáticamente las siguientes tareas: trazado de área del chip, colocación de celdas, *pads*, puertos y conexión de componentes. Los resultados arrojados demostraron la correcta funcionalidad del PRNG en forma comportamental y eléctrica, así como también la información de retardos y de celdas empleadas. A partir del reporte de celdas empleadas se analizaron aquellas con mayores y menores instancias en el circuito. En la implementación física se estudiaron las dimensiones de las celdas con mayores y menores instancias, así como aquellas de mayor y menor tamaño. A partir de ello se obtuvieron porcentajes de ocupación del área del núcleo para cada una.

Capítulo 4

Propuesta de implementación de un PRNG mejorado

En el capítulo 2 se estudió el PRNG basado en una cadena de acumuladores digitales con coeficientes variantes en el tiempo. Con base a los resultados del capítulo 3, los números emitidos tienen un tiempo de duración igual al periodo de la señal de reloj, además de repetirse la secuencia de valores después de un pulso de reinicio (*res*). Partiendo de dicho punto, surge la posibilidad de cambiar el comportamiento del generador, de pseudoaleatorio a aleatorio. Si el reloj es sustituido por una señal digital aperiódica, la duración de cada número emitido será diferente e igual al tiempo entre los dos flancos de subida que lo delimitan. Así, a pesar de generar numéricamente la misma secuencia, las diferentes duraciones de cada valor convierten el proceso a una forma aperiódica, y por lo tanto aleatorio.

El nuevo sistema fue conformado por el PRNG original en su modelo comportamental en conjunto con un Oscilador Aleatorio Booleano (OAB) a nivel transistor que generó la señal digital aleatoria hacia el puerto del reloj en el PRNG. Esta implementación se caracterizó en modo de señal mixta, lo que permite una mayor velocidad de simulación y un menor tiempo de desarrollo. A continuación se presenta el esquema general del sistema.

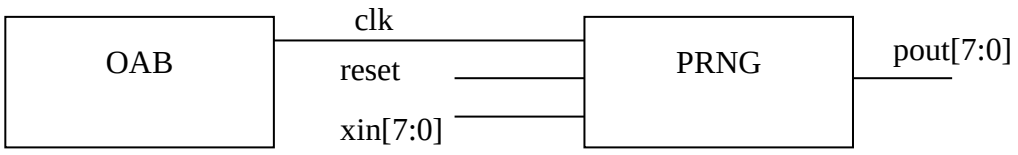


Figura 4.1. Esquema general del PRNG mejorado.

4.1 Diseño del Oscilador Aleatorio Booleano (OAB)

El principio de funcionamiento del OAB es la propagación asíncrona de señales digitales. Tal comportamiento se logra al existir lazos de retroalimentación entre módulos combinacionales como pueden ser las compuertas lógicas, en ausencia de una señal de control. Sin embargo, un sistema asíncrono no necesariamente es aperiódico, por lo cual se requiere diseñar una arquitectura con la facultad de aprovechar tal propagación y emitir un comportamiento aleatorio. Lo anterior es completamente implementado con elementos digitales que implica una mayor facilidad de integración con otros sistemas de la misma índole a nivel hardware. Por tales razones se eligió desarrollar al OAB con las características antes mencionadas.

El diagrama a bloques del OAB propuesto es el siguiente.

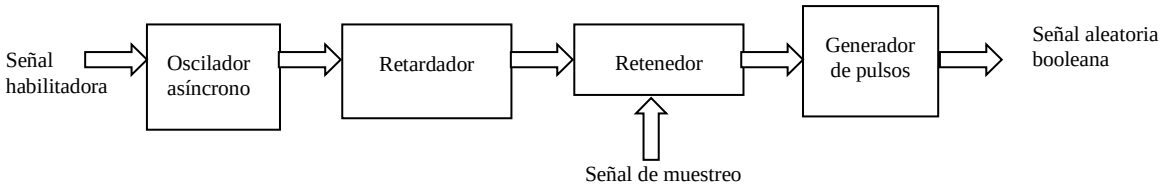


Figura 4.2. Diagrama a bloques del Oscilador Aleatorio Booleano.

Las funciones generales de cada etapa son:

- **Oscilador asíncrono:** emite una señal digital asíncrona periódica.

- **Retardador:** procesa la señal asíncrona periódica de la etapa anterior para convertirla en aleatoria.

- **Retenedor:** retiene valores digitales de la salida del retardador conforme a una señal de muestreo para formar un tren aperiódico de pulsos con anchos muy reducidos, en el orden de picosegundos.

- **Generador de pulsos:** a partir del tren de pulsos emitido por el retenedor construye la señal digital aleatoria final, con periodos variables entre flancos ascendentes y duraciones del orden de nanosegundos para ambos niveles lógicos.

El sistema fue implementado en el flujo de diseño *IC Nanometer* y el editor de esquemáticos de *Menthor Graphics* con el empleo de celdas digitales genéricas del kit de diseño ADK para un proceso de 0.35µm. En cada etapa diseñada se ejecutó un proceso de simulación de 1µs de duración para comprobar el funcionamiento de cada una y su acoplamiento con las demás. A continuación se explica el procedimiento de diseño para cada etapa.

4.1.1 Oscilador asíncrono

Se conectó una compuerta AND de 2 entradas con un inversor como retroalimentación entre la salida y una entrada, la restante (ENABLE) se estableció como habilitadora del oscilador, con lo cual se obtiene un tren de pulsos periódico con un ciclo de trabajo del 50%, cuya duración depende de los retardos de propagación de las compuertas.

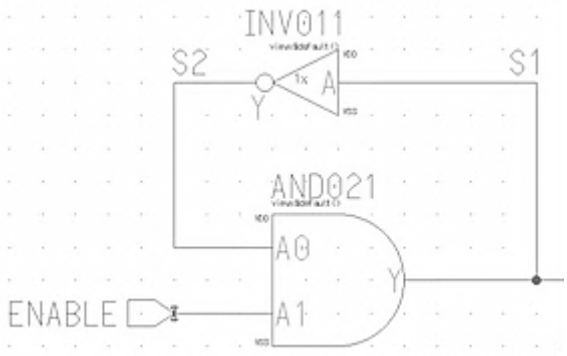


Figura 4.3. Oscilador asíncrono.

Así, el inversor complementa la salida actual (S1) para preparar el próximo valor a emitir (S2) y ser arrojado por medio de la compuerta AND, tarea que se llevará a cabo siempre y cuando ENABLE mantenga un valor lógico 1. La tabla de verdad correspondiente se muestra en la tabla 4.1.

Tabla 4.1. Tabla de verdad del oscilador asíncrono.

ENABLE	S2	S1
0	X	0
1	0	0
1	1	1

Sin embargo en la práctica, debido a la alta frecuencia de la propagación asíncrona, la salida presenta un comportamiento sinusoidal.

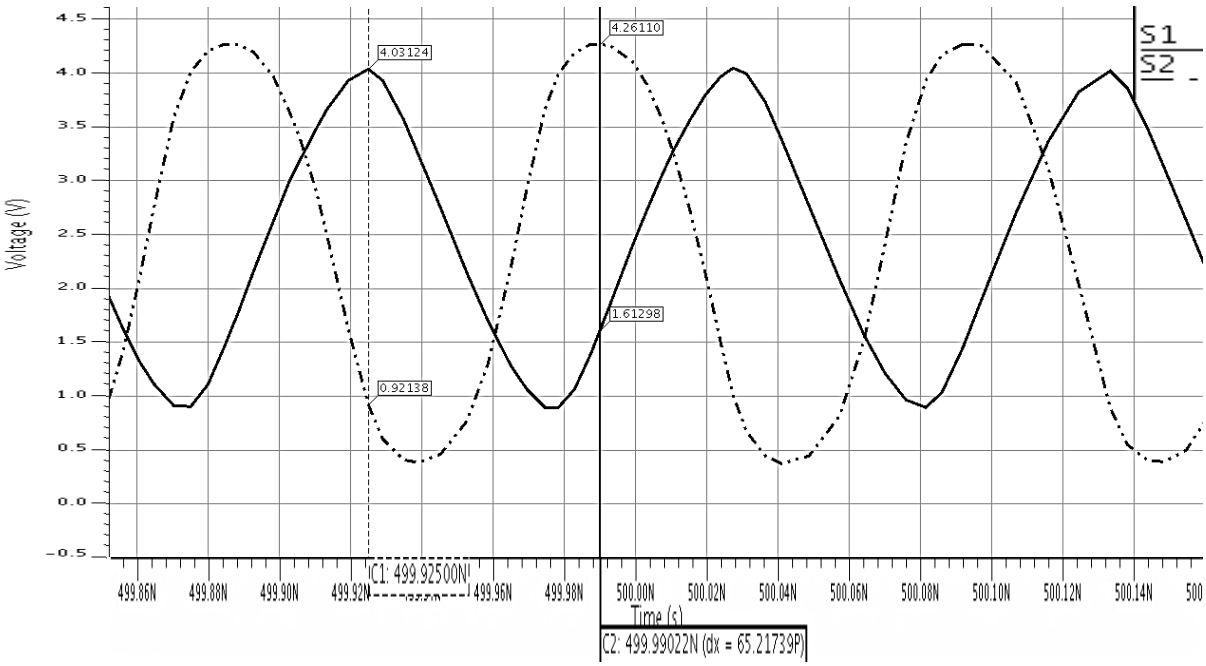


Figura 4.4. Señales S1 y S2 del oscilador asíncrono.

De la imagen anterior, ambas señales presentan un periodo de 104ps, es decir de 9.6GHz. También muestran un desfasamiento de 65ps equivalente a 225º generado por el inversor. El voltaje pico – pico de la salida S1 se atenúa en un 38% respecto al ideal de 0V – 5V debido a los retardos de propagación conjuntos de ambas compuertas que impiden la formación de pulsos completamente definidos ante una oscilación de tales características. Dichos factores se tomaron en cuenta para diseñar la siguiente etapa: el retardador.

4.1.2 Retardador

El objetivo principal de este sistema consistió en generar una señal aleatoria a partir de la oscilación asíncrona de la etapa previa, la cual es determinística. La propuesta para lograrlo fue implementar un circuito que trabaje bajo el principio de los retardos de propagación. Aunque todas las compuertas a nivel transistor presentan tales retardos, estos son despreciables en bajas frecuencias, no así para oscilaciones asíncronas. El objetivo de tal circuito es retardar la señal asíncrona hasta llegar a un punto en el cual la salida de bits no concuerde con lo esperado en un nivel puramente comportamental. De esta manera la salida está más en función de elementos analógicos que de la combinación de entradas digitales. Estos elementos analógicos son las capacitancias y resistencias equivalentes presentes en cada uno de los transistores activados dada una combinación de las entradas, los cuales son precisamente los causantes de los retardos de propagación.

Tal circuito se diseñó en dos fases debido a que en cada una se analizaron los resultados a fin de determinar la necesidad o no de una nueva etapa.

4.2.1.1 Retardador fase 1

Esta consistió en implementar un circuito conformado por las compuertas XOR e inversor, tal como se muestra a continuación.

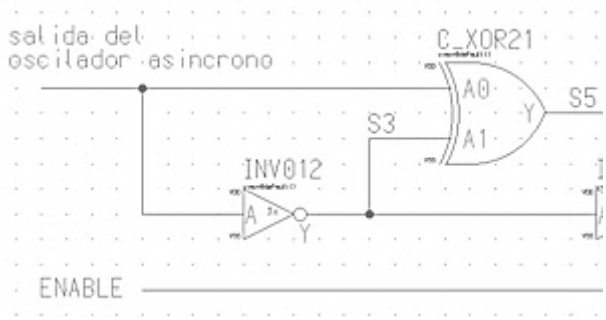


Figura 4.5. Circuito retardador fase 1.

Se eligió una compuerta XOR debido a que emite una señal de bits con distribución uniforme a partir de las 4 posibles combinaciones de entrada, de esta manera no predomina más un valor digital sobre otro. A esta compuerta ingresa la salida del oscilador asíncrono y el desfasamiento de esta (S3) por medio del mismo modelo de inversor usado en la etapa previa, donde se dedujo genera un retardo de 65ps equivalente a 225º. Así fue posible comparar dos señales distintas para producir una salida variable en el tiempo, además de únicamente trabajar con la señal del oscilador asíncrono sin requerir otras adicionales que implicarían un aumento de componentes digitales.

Para enfatizar los efectos de los retardos de propagación en una señal asíncrona y su importancia para generar una señal aleatoria, se simuló el circuito anterior en *Matlab R2009a* (ver anexo C) a nivel comportamental incluyendo el desfasamiento del inversor. Los resultados que se presentan a continuación son los que deberían de existir sin la presencia de tales retardos.

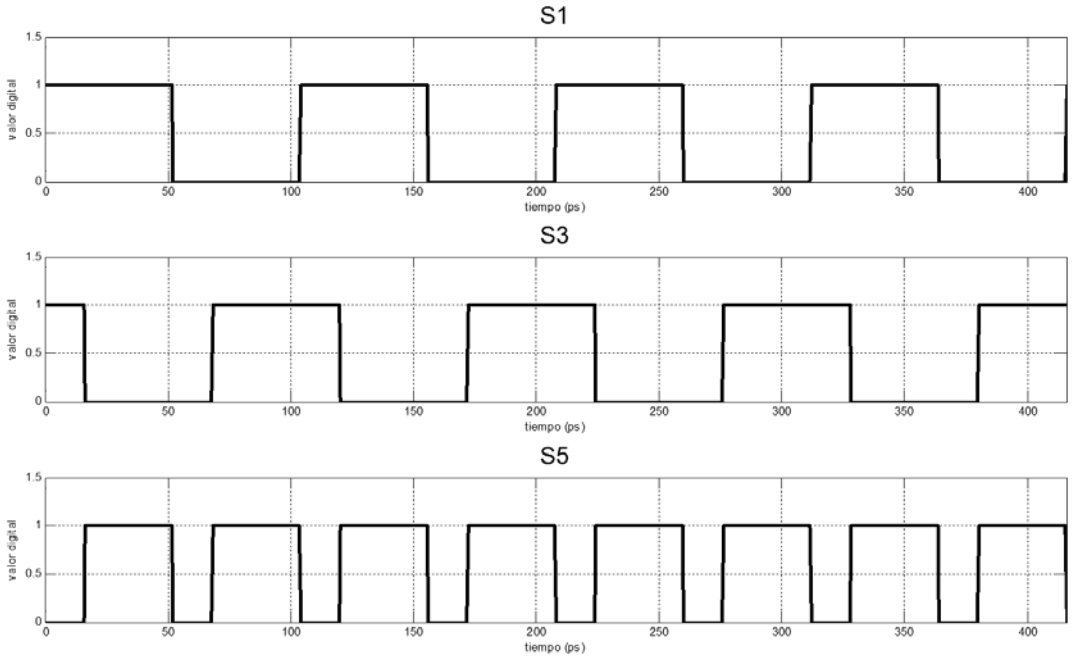


Figura 4.6. Resultados de la simulación comportamental en *Matlab* del circuito retardador fase 1.

De acuerdo a la imagen anterior, lo ideal es una salida (S5) periódica de 52ps con un ciclo de trabajo de aproximadamente 70%.

Los resultados de simulación en *Menthor Graphics* se plasman a continuación.

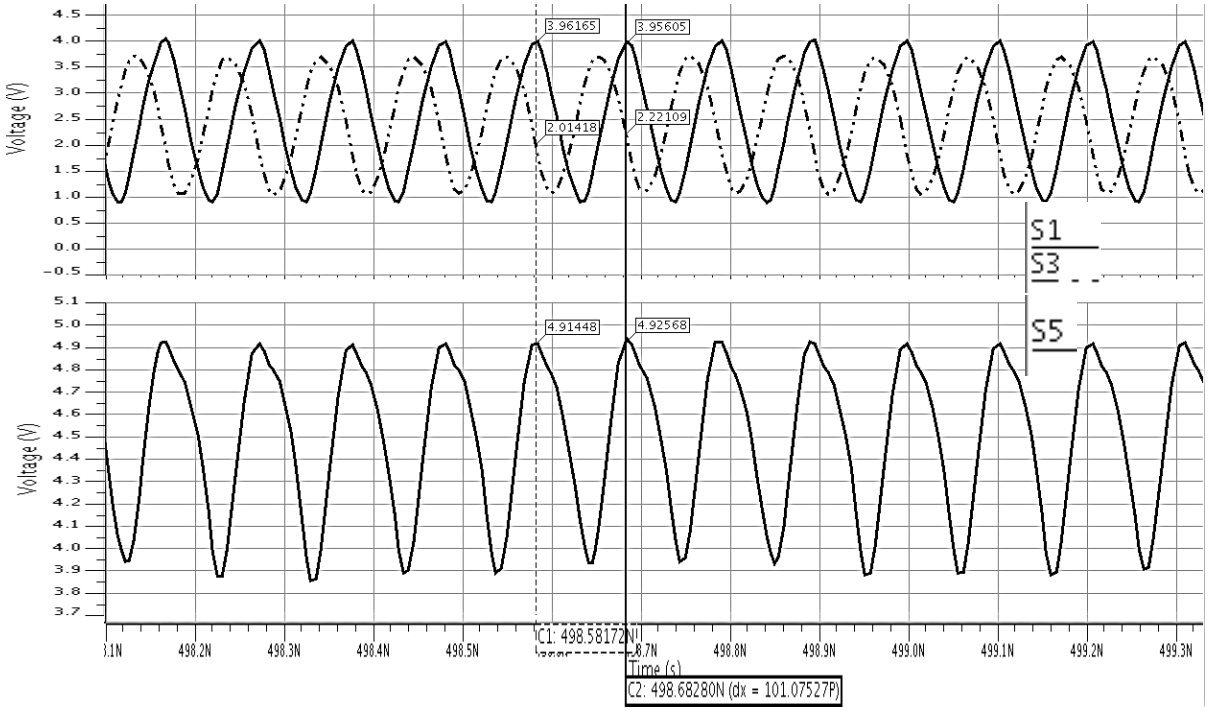


Figura 4.7. Resultados de la simulación del circuito retardador fase 1.

Como era de esperarse, los retardos de propagación generan una salida muy distinta respecto a lo idealmente esperado. Las características más sobresalientes de S5 son la atenuación del voltaje pico – pico de aproximadamente 1V entre 4.9V y 3.85V y el nivel de DC de 4.37V. Lo anterior significa que tal señal únicamente oscila en el umbral de nivel alto y podría considerarse como un 1 lógico constante. Las altas frecuencias de S1 y S3 aunado con los retardos de propagación impiden que la salida S5 logre los voltajes digitales, imposibles de alcanzar por las condiciones de carga. Por ello, son las oscilaciones de baja amplitud en el umbral de alto. En un nivel de diseño más inferior, la explicación está en que el capacitor equivalente en cada compuerta no termina de cargarse y descargarse por completo, de ahí que almacena energía y esta es precisamente la causante del nivel de DC a la que se encuentra sujeta.

4.2.1.1 Retardador fase 2

Con S5 en el umbral de alto, aunque oscilando en forma periódica, no tiene la característica aleatoria, por lo que se requirió de una segunda fase en el diseño total del circuito. Se propuso integrar nuevamente una compuerta XOR y un inversor como se plasma a continuación.

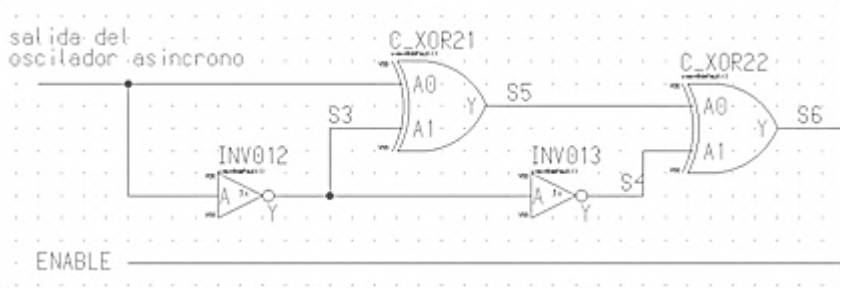


Figura 4.8. Circuito retardador fase 2.

Las razones de emplear nuevamente las compuertas de la primera fase son las mismas expuestas anteriormente. La salida de la fase 1 (S5) se compara con S4 que es el desfasamiento de S3 a través del mismo modelo de inversor. Esto significa un retardo total de 130ps, equivalente a 450º ≡ 90º respecto a la señal asíncrona original (S1). Como se observó previamente, S5 puede idealizarse constante en nivel alto, por lo que un desfasamiento con un inversor solo arrojaría una señal constante ahora en nivel bajo, tal que si se ingresa esta señal junto con S5 a una compuerta XOR el resultado sería 1 lógico constante, es decir no habría oscilación. Esta es la razón de no desfasar S5 en forma similar a la fase 1.

Nuevamente, se simuló el circuito anterior en *Matlab R2009a* con las mismas condiciones y razones expuestas en la fase 1.

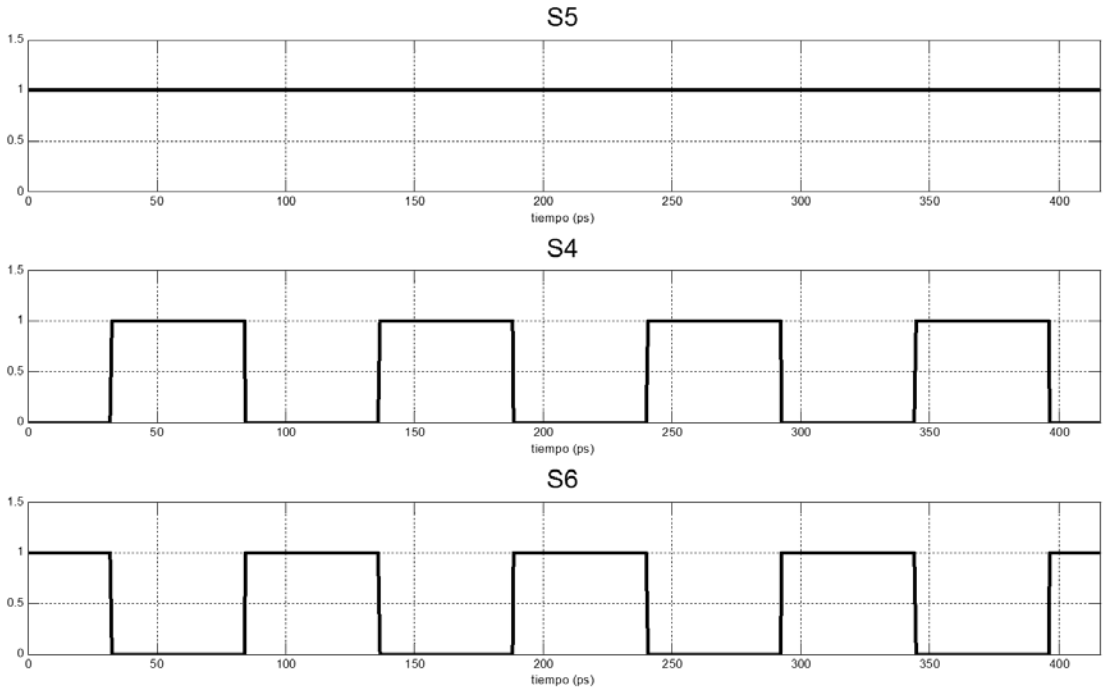


Figura 4.9. Resultados de la simulación comportamental en *Matlab* del circuito retardador fase 2.

De los resultados anteriores, S6 es el complemento de S4, con un periodo de 104ps y un ciclo de trabajo del 50%, no así ocurre en la simulación en *Menthor Graphics* donde se toman en cuenta los retardos de propagación.

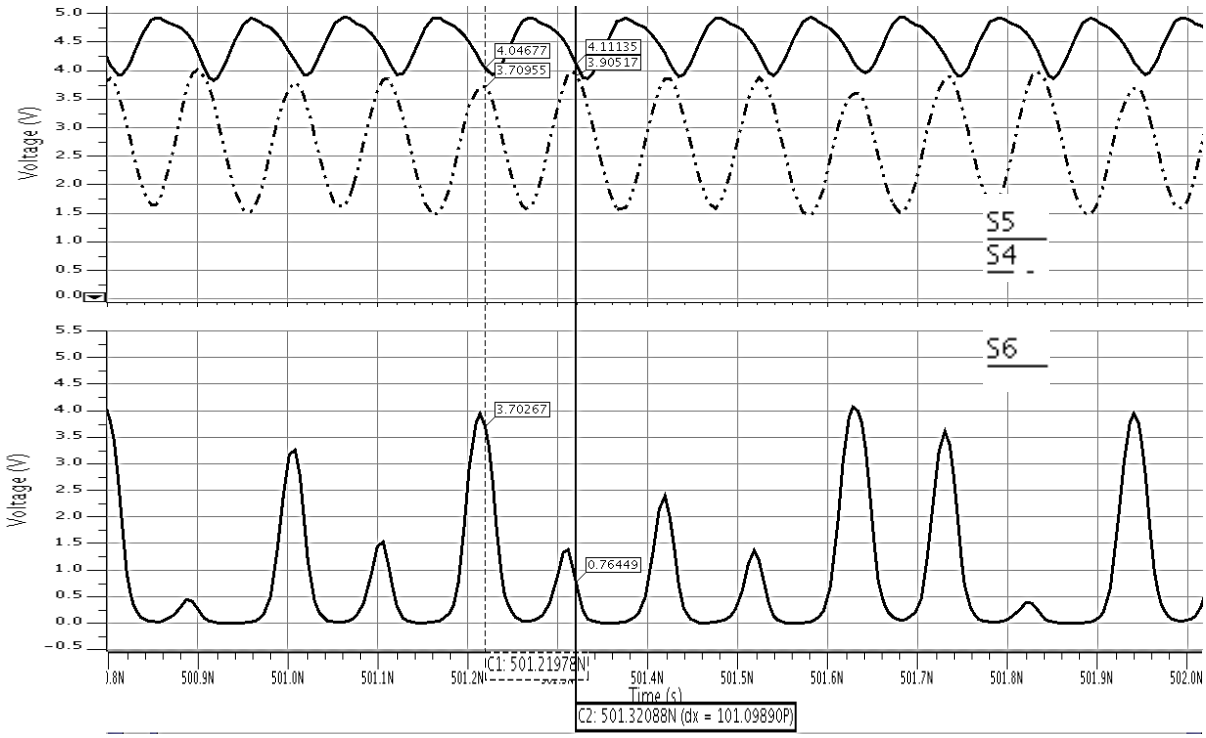


Figura 4.10. Resultados de la simulación del circuito retardador fase 2.

De acuerdo a lo anterior, es evidente que a pesar de existir una operación XOR sobre dos señales determinísticas, el resultado (S6) es una secuencia de pulsos en voltajes e instantes de tiempo impredecibles, inconsistente con la simulación en *Matlab*. Éste es el grado donde los retardos de propagación afectan significativamente el comportamiento lógico del circuito, quedando más en función de elementos analógicos que de digitales. El tiempo mínimo entre pulsos sigue conservando aproximadamente el periodo de la señal asíncrona original, de 104ps.

Algunos pulsos existentes en S6 no alcanzan completamente los 5V. De la imagen anterior muchos no rebasan el voltaje de 2.5V que limita el umbral de nivel bajo y alto. Para definir mejor los pulsos generados, se anexó como elemento final una compuerta AND como se muestra en la siguiente figura, donde se muestra el circuito retardador final.

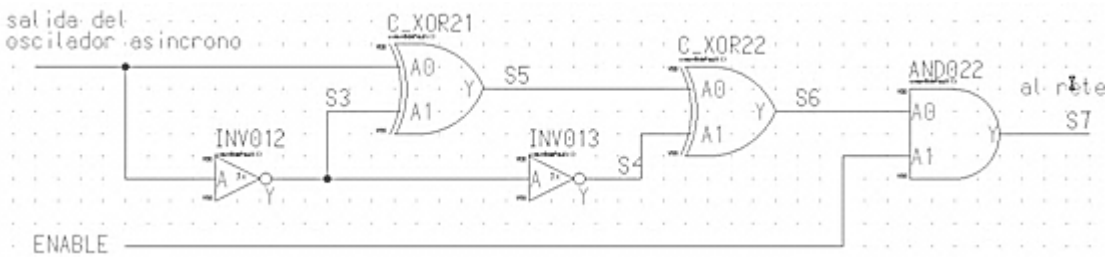


Figura 4.11. Circuito retardador final.

Son dos las funciones de la compuerta AND: definir mejor la secuencia de pulsos (S6) y habilitar el paso de la salida final (S7) cuando ENABLE = 1, la misma señal del oscilador asíncrono. Los resultados se muestran a continuación.

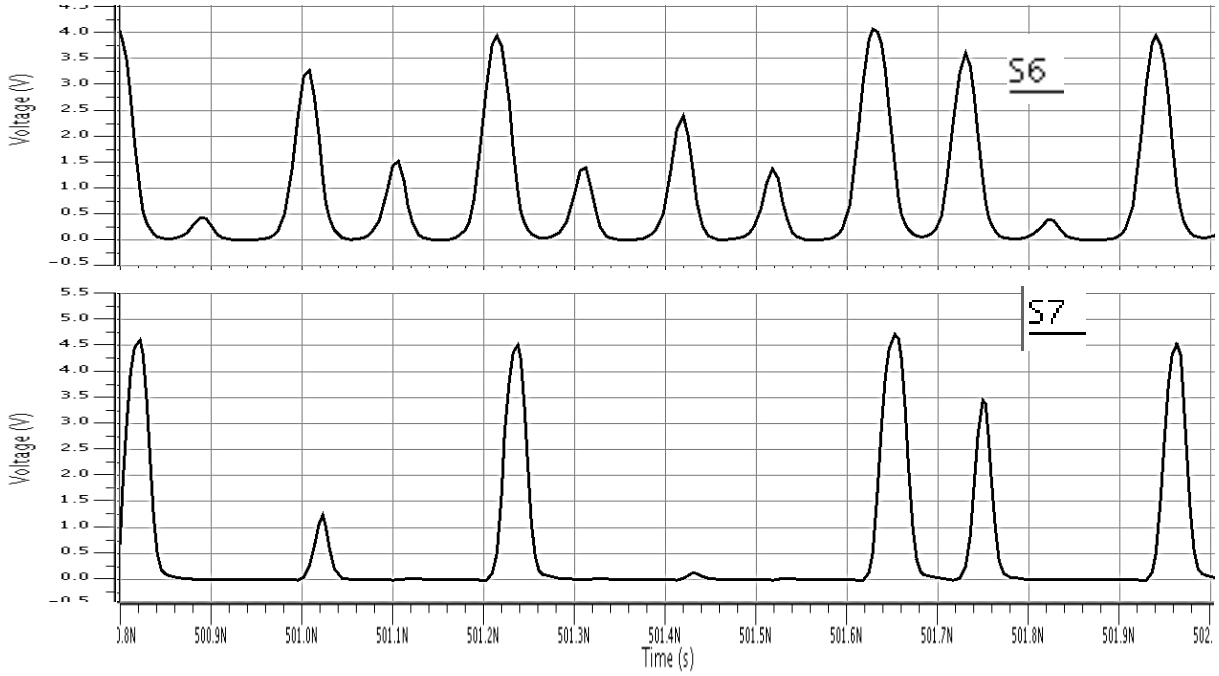


Figura 4.12. Resultados de simulación del circuito retardador final.

Puede observarse claramente como la compuerta AND suprime aquellos pulsos que no alcanzan a rebasar el umbral de nivel bajo sin cambiar su naturaleza lógica, por lo que puede considerarse como eliminación de ruido. La importancia de definir adecuadamente las señales se apreciará más adelante con las siguientes etapas. Es finalmente S7 la señal aleatoria esperada, por lo cual el objetivo planteado al comienzo de esta etapa fue alcanzado.

4.1.3 Retenedor

La señal emitida por el circuito retardador puede considerarse como aleatoria. Sin embargo, aún no puede ser empleada como señal de control del PRNG. Existen dos razones, la primera es el ancho reducido de los pulsos menor a 1ns y la segunda es el breve tiempo mínimo entre flancos ascendentes, menores también a 1ns. Estas condiciones no brindan al PRNG el tiempo suficiente para generar un valor pseudoaleatorio, es decir su velocidad de respuesta es muy lenta para una señal de control del orden de picosegundos, esto conforme al análisis de resultados del capítulo 3 referente al estudio de las oscilaciones asíncronas generadas por el PRNG a nivel estructural.

La siguiente etapa diseñada consistió en un circuito de retención de pulsos al azar por medio de una señal de muestreo de frecuencia Fs que es a la cual opera originalmente el PRNG dentro de los rangos establecidos para despreciar o no las oscilaciones asíncronas, acorde al análisis de resultados del capítulo 3. El resultado es una señal con anchos de pulsos aún reducidos pero con un tiempo mínimo entre flancos ascendentes igual a 1/2Fs, por lo tanto con la posibilidad de ser establecido a un valor preferido y ser mayor a 1ns.

Al tratarse de pulsos asíncronos del orden de picosegundos, se optó también por un elemento digital de retención asíncrono, como un *latch SR* de compuertas NOR.

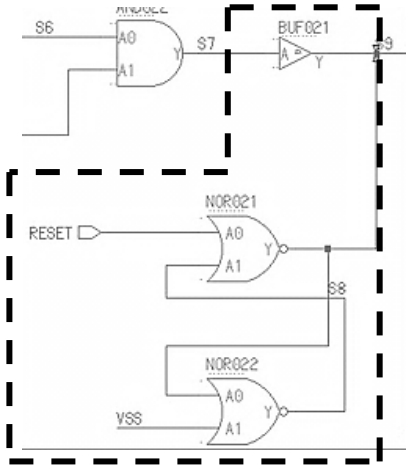


Figura 4.13. Circuito retenedor, enmarcado en líneas punteadas.

Tabla 4.2. Tabla de verdad del retenedor.

SET	RESET	S8[i]	Modo de operación
0	0	S8[i-1]	Memoria
0	1	0	Reset
1	0	1	Set
1	1	Estado no deseado	No deseado

El funcionamiento consiste en hacer operar el *latch* en dos modos: memoria para habilitar la posibilidad de retener un pulso y *reset* para borrar el dato y preparar nuevamente el *latch* hacia el estado de memoria. Para ello SET = 0 = VSS y RESET es una señal de reloj de frecuencia Fs = 50MHz, es decir un periodo de muestreo de 20ns. Aunque dicho lapso de muestreo es menor al mínimo propuesto de 40ns para despreciar los periodos de oscilaciones asíncronas según los resultados del capítulo anterior, para el PRNG modelo Verilog AMS no ocurre dicho fenómeno. Por lo tanto, con el propósito de caracterizar el PRNG mejorado con un *testbench* en señal mixta en un tiempo menor y con una cantidad de muestras suficientes para analizar su distribución de datos, se optó por fijar tal valor de Fs. Junto a las etapas anteriores, la salida del sistema es la mostrada en la figura 4.14.

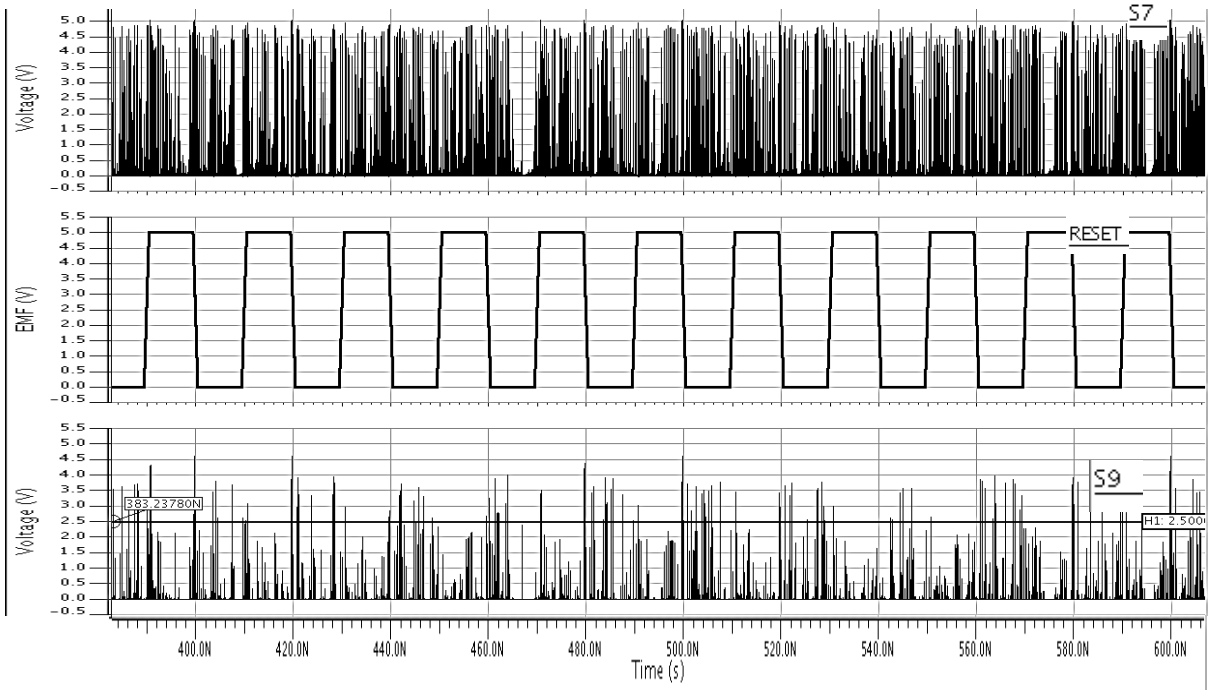


Figura 4.14. Señal del retardador (S7), señal de muestreo (RESET) y salida del retenedor (S9).

De la imagen anterior se aprecia como los pulsos retenidos, es decir aquellos que superan la línea horizontal de 2.5V, ocurren en cada transición de la señal de muestreo (RESET), ya sea de bajo a alto o viceversa.

En un inicio se esperó la posible retención de un pulso digital cada 20ns, es decir en cada flanco de subida de la señal RESET. Sin embargo, de acuerdo a los resultados esta posibilidad se dio también durante algunos flancos de bajada. Por lo tanto, el lapso entre una posible retención de un pulso digital se realizó cada 10ns, circunstancia no prevista pero benéfica, pues es posible muestrear al doble de la frecuencia de RESET.

El *buffer* entre la señal del retardador y el *latch* sirve para mantener el funcionamiento independiente de estos bloques, pues evita que el retardador detenga sus oscilaciones mientras se ejecuta la posible retención del pulso por parte del *latch*. Esta implementación evita la necesidad de un interruptor analógico para realizar la misma tarea. Finalmente el retenedor emite una señal con mucho ruido por debajo de los 2.5V, situación no perjudicial ya que al no superar el límite es considerado como nivel bajo.

4.1.4 Generador de pulsos

El retenedor emite una señal aleatoria de pulsos con un tiempo mínimo entre ellos igual a 1/2Fs, es decir la mitad del periodo de la señal RESET. Sin embargo, dichos pulsos poseen un ancho muy reducido, menor a 1ns. El objetivo final fue aumentar tal ancho al orden de nanosegundos. No obstante, al ser los pulsos tan reducidos de aproximadamente 50ps en realidad podrían considerarse más como flancos de subida. Si en el tiempo entre pares de estos flancos, que es siempre variable, se retiene un valor booleano y se alterna por su complemento en el próximo par se obtendrá un tren de bits con tiempos de duración mayores a 1ns para ambos valores. Esta propuesta fue posible por medio de un flip - flop D con un inversor como retroalimentación.

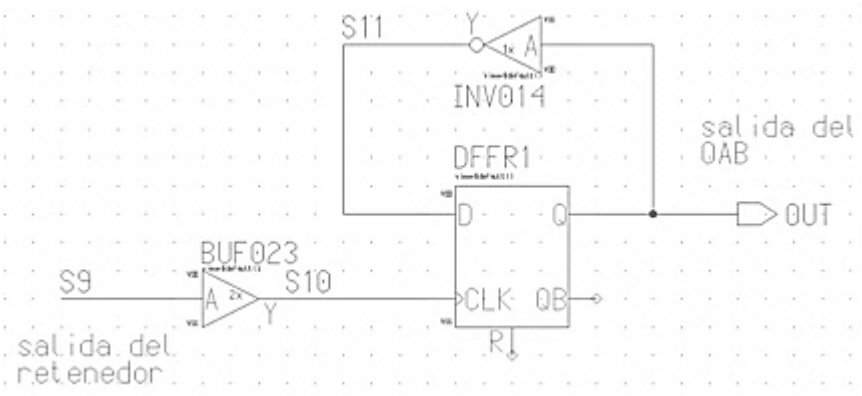


Figura 4.15. Circuito generador de pulsos.

La tabla de verdad respectiva es la siguiente.

Tabla 4.3. Tabla de verdad del generador de pulsos.

Reloj	OUT[i]	OUT[i+1]
Flanco ascendente	0	1
	1	0
Flanco descendente	X	OUT[i]

El flip - flop D transmite el dato de su entrada a la salida en el momento de recibir un flanco de subida por parte del retenedor. El inversor complementa la salida y la retroalimenta a la entrada para efectuar el cambio alternado de niveles lógicos entre transiciones. El *buffer* entre la señal del retenedor y el puerto de reloj del flip - flop D sirve para eliminar el ruido existente en el umbral del nivel bajo y transmitir una secuencia de pulsos correctamente definidos.

La salida de esta etapa, junto con la de entrada y de muestreo (RESET), es la siguiente.

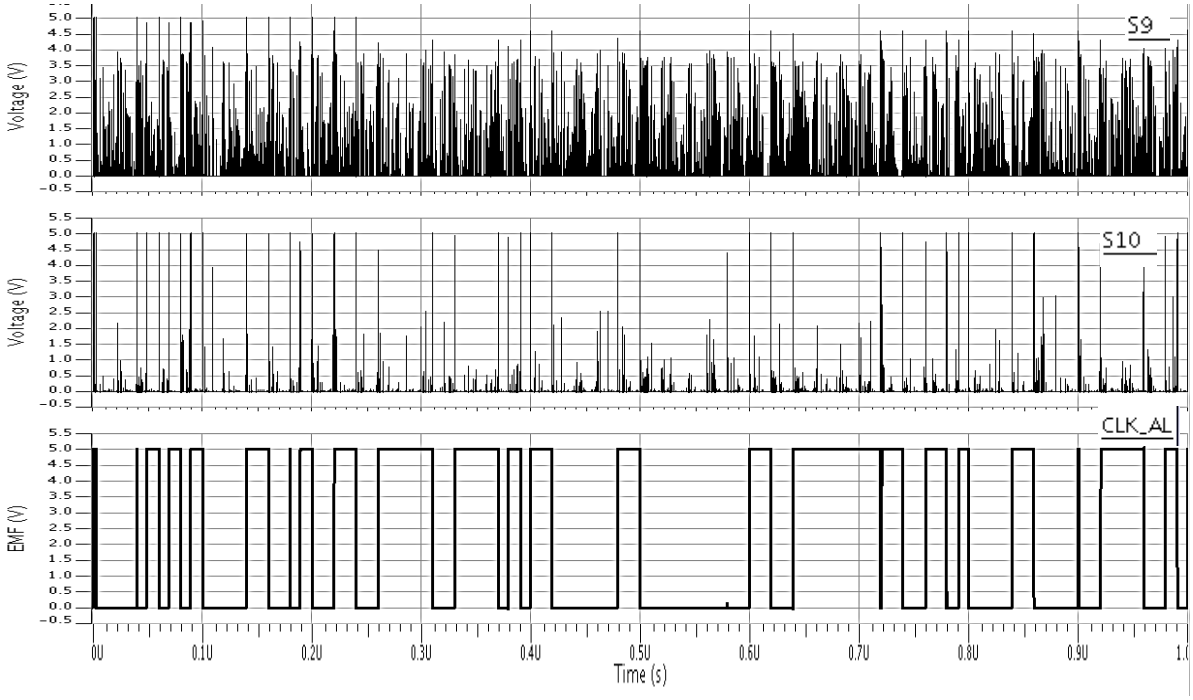


Figura 4.16. Señal del retenedor (S9), salida del buffer (S10) y señal aleatoria booleana (CLK_AL).

Es finalmente la salida de esta etapa la señal aleatoria booleana esperada. Por fenómenos analógicos impredecibles del circuito, llegan a formarse algunos pulsos de anchos menores a 1ns como los ocurridos en 0.18µs, 0.9µs por mencionar algunos. No obstante esto no afecta la correcta emisión de números por parte del PRNG en su modelo comportamental cuando se integra con el OAB. Como se verá más tarde en el análisis de resultados de este capítulo, tales pulsos son vistos como flancos ascendentes y pueden surgir cada 1/2Fs como mínimo, de tal modo que el periodo mínimo posible de cada número pseudoaleatorio es también 1/2Fs, con la opción de ser mayor a 1ns.

Al integrarse cada una de las etapas anteriores se conformó el OAB requerido, como se muestra a continuación.

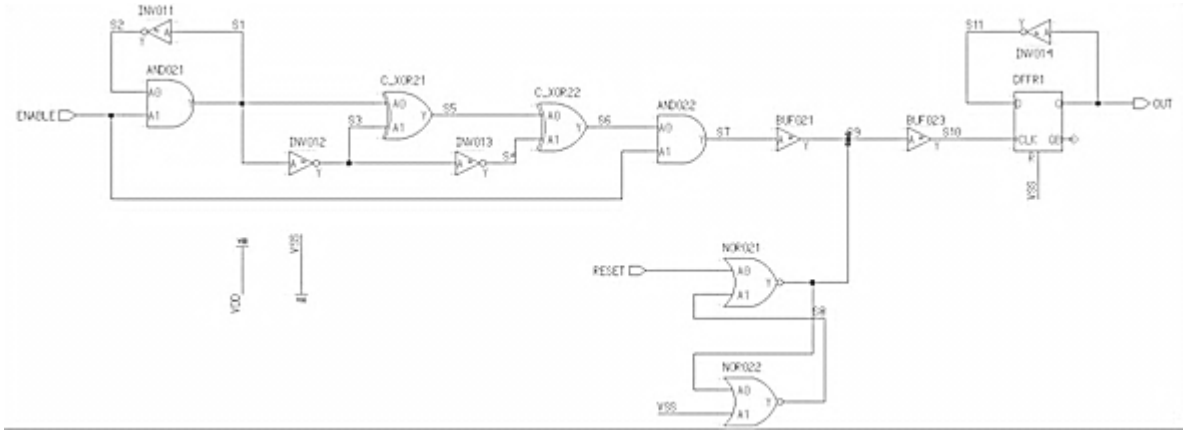


Figura 4.17. Diseño final del OAB.

Con base en la figura 4.17, el OAB fue conformado por 13 celdas digitales, 12 de ellas de tipo combinacional y únicamente el flip - flop D como tipo secuencial, sin requerir fuentes analógicas de aleatoriedad. Tal sistema emplea únicamente dos señales de entrada tipo bit: ENABLE para habilitar el OAB y RESET para fijar el periodo mínimo posible de los bits aleatorios generados.

Figura 4.18. Símbolo representativo del OAB.

La figura 4.18 muestra el símbolo representativo del OAB para fines de simulación con otros sistemas, creado en automático por el editor de símbolos de *Menthor Graphics*.

4.2 Implementación del PRNG mejorado

Al ser el OAB un sistema caótico a integrarse con el PRNG en su modelo comportamental, este último se compiló en el entorno *Language Interface* de *Menthor Graphics* en lenguaje Verilog AMS destinado para sistemas en señal mixta, usando el mismo código fuente de *VSIM*. Posteriormente en el mismo entorno se generó el símbolo representativo.

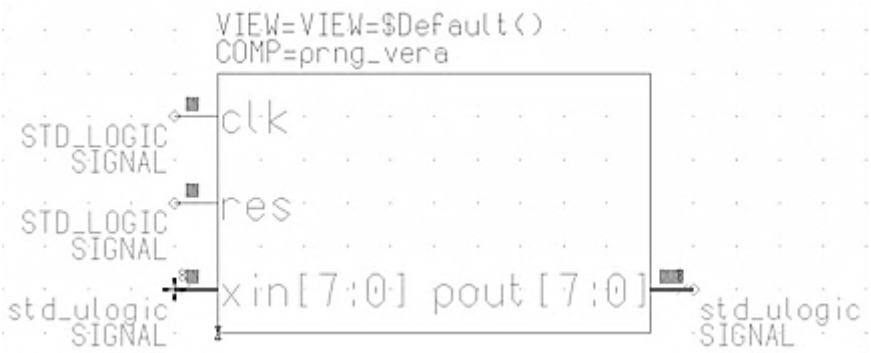


Figura 4.19. Símbolo representativo del PRNG en su modelo comportamental.

Seguidamente se integraron el OAB y el PRNG en el editor de esquemáticos de *Menthor Graphics*, conectados conforme a lo especificado en la figura 4.1. Esto se demuestra en la figura 4.20.

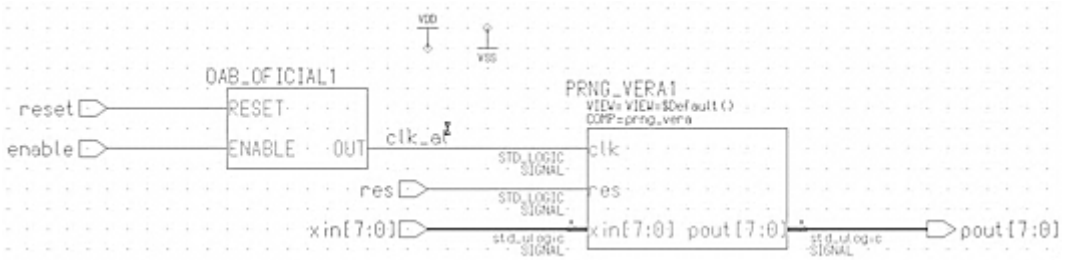


Figura 4.20. Circuito del PRNG mejorado para fines de simulación.

Posteriormente se procedió con la simulación. El entorno empleado fue ADMS para pruebas en señal mixta.

Figura 4.21. Cuadro de dialogo para especifica el modo de simulación, donde se eligió ADMS.

Las configuraciones para simular se presentan a continuación.

a) tipo de análisis: transitorio, con tiempo de corrimiento de 20µs.

b) tecnología empleada: 0.35µm

c) forzamiento de señales:

- **VDD:** señal de DC de 5 V/A, para fines de simulación en señal mixta
- **GND:** señal de DC de 0 V/A
- **RESET:** señal de reloj con periodo de 20ns, tiempo de elevación y caída de 1ns. Valor en alto y bajo de 5 y 0 V/A, respectivamente.
- **ENABLE:** escalón unitario en 1ns, con valor en nivel bajo y alto de 0 y 5 V/A respectivamente y un tiempo de elevación y caída de 1ns.
- **RES:** pulso único de 1ns de ancho, inicio en 1ns, valor en nivel alto y bajo de 0 y 5 V/A respectivamente y un tiempo de elevación y caída de 1ns.
- **XIN:** señal vector de bits constante de valor $(10101010)_2 = (170)_{10}$, nivel alto y bajo de 5 y 0 V/A, a partir de 0ns.

d) Visualización de las salidas: voltajes de las señales *clk_al, reset y pout[7:0*].

Al ingresar señales analógicas al PRNG digital, donde una de ellas es la salida del OAB (*clk_al*), se emplearon ADC configurados conforme a [17] y mostrados en la siguiente imagen.

Model Name	Boundary	Type	Object	Info
MODEL__A2D_STD_LO...	A2D	builtin	/clk_al	Mode=STD_LOGIC Str=STRONG Vth=2.5
MODEL__A2D_STD_LO...	A2D	builtin	/res	Mode=STD_LOGIC Str=STRONG Vth=2.5
MODEL__A2D_STD_LO...	A2D	builtin	/xin[0]	Mode=STD_LOGIC Str=STRONG Vth=2.5
MODEL__A2D_STD_LO...	A2D	builtin	/xin[1]	Mode=STD_LOGIC Str=STRONG Vth=2.5
MODEL__A2D_STD_LO...	A2D	builtin	/xin[2]	Mode=STD_LOGIC Str=STRONG Vth=2.5
MODEL__A2D_STD_LO...	A2D	builtin	/xin[3]	Mode=STD_LOGIC Str=STRONG Vth=2.5
MODEL__A2D_STD_LO...	A2D	builtin	/xin[4]	Mode=STD_LOGIC Str=STRONG Vth=2.5
MODEL__A2D_STD_LO...	A2D	builtin	/xin[5]	Mode=STD_LOGIC Str=STRONG Vth=2.5
MODEL__A2D_STD_LO...	A2D	builtin	/xin[6]	Mode=STD_LOGIC Str=STRONG Vth=2.5
MODEL__A2D_STD_LO...	A2D	builtin	/xin[7]	Mode=STD_LOGIC Str=STRONG Vth=2.5

Figura 4.22. Configuración de ADC para el PRNG mejorado.

Donde Vth = 2.5V es el voltaje de umbral entre el nivel alto y bajo. La configuración Mode = STD_LOGIC refiere al uso de un modelo de ADC de lógica estándar, es decir puede emitir tres posibles estados lógicos: alto, bajo y desconocido (*unknown*). El estado *unknown* es emitido en la salida de un sistema cuando se desconocen las condiciones iniciales.

Después de correr la simulación se graficó la Transformada Rápida de Fourier (FFT) de la salida del OAB y el PRNG mejorado con la finalidad de visualizar mejor el carácter aleatorio de estas señales. Las configuraciones de la FFT fueron automáticamente calculadas por el software acorde a los datos obtenidos de la simulación.

- Tiempo de inicio: 0s
- Tiempo de detención: 20µs
- Frecuencia de muestreo: 102MHz
- Número de puntos: 1024

Adicionalmente se trazó un histograma de las señales anteriores para determinar la distribución de datos.

4.3 Análisis de resultados

A continuación se plasman los valores emitidos por el PRNG mejorado y el OAB (señal CLK_AL) en un tiempo de 20µs.

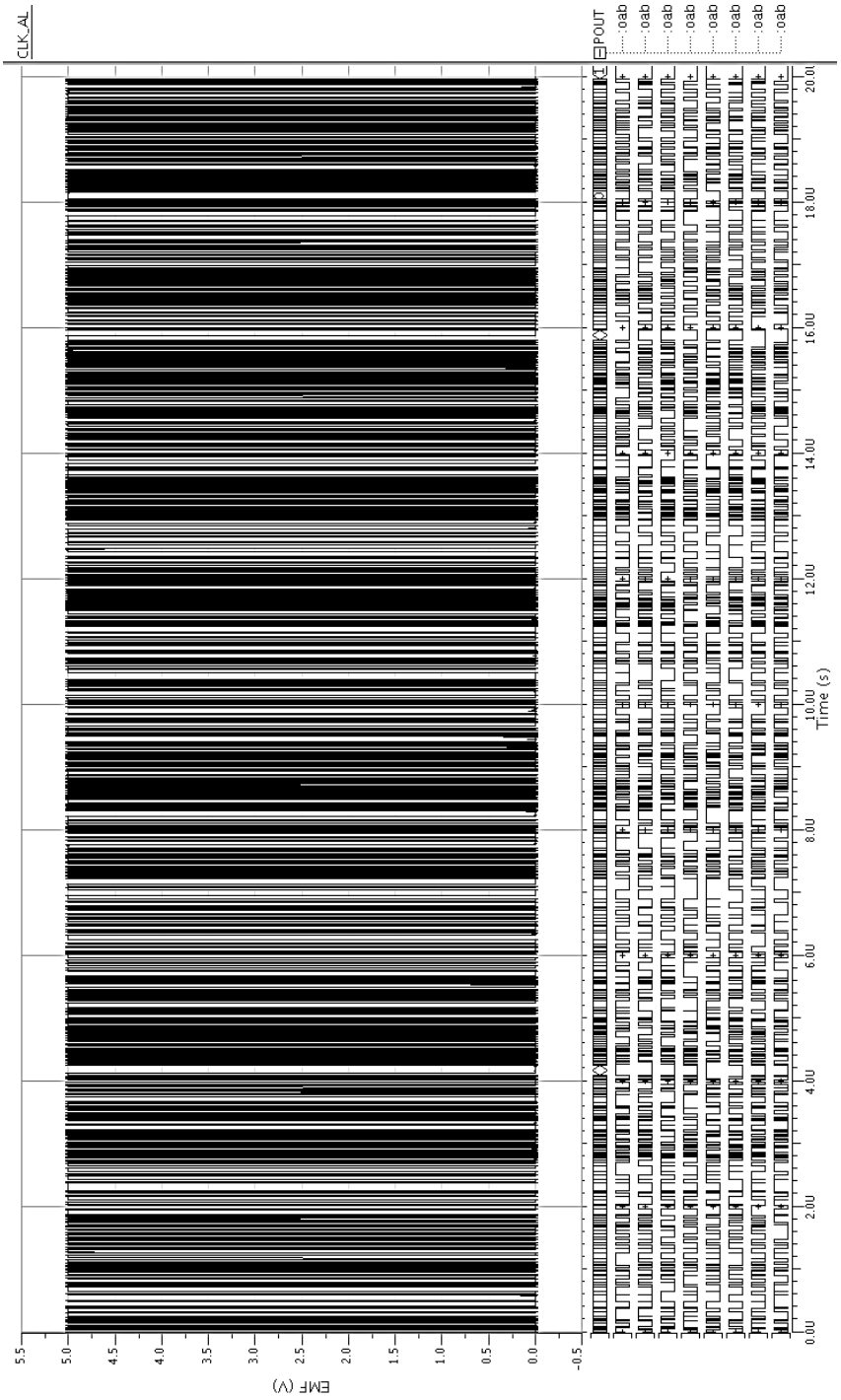


Figura 4.23. Resultados de simulación del PRNG mejorado y del OAB.

De la figura anterior se han desplegado cada uno de los bits que conforman la salida del PRNG mejorado (POUT). Al observar a simple vista la salida del OAB (CLK_AL) y la del PRNG mejorado no se detecta un comportamiento periódico en un tiempo de 20µs. Para analizar mejor los instantes de generación de dichos valores, la siguiente imagen plasma una ampliación de los resultados anteriores.

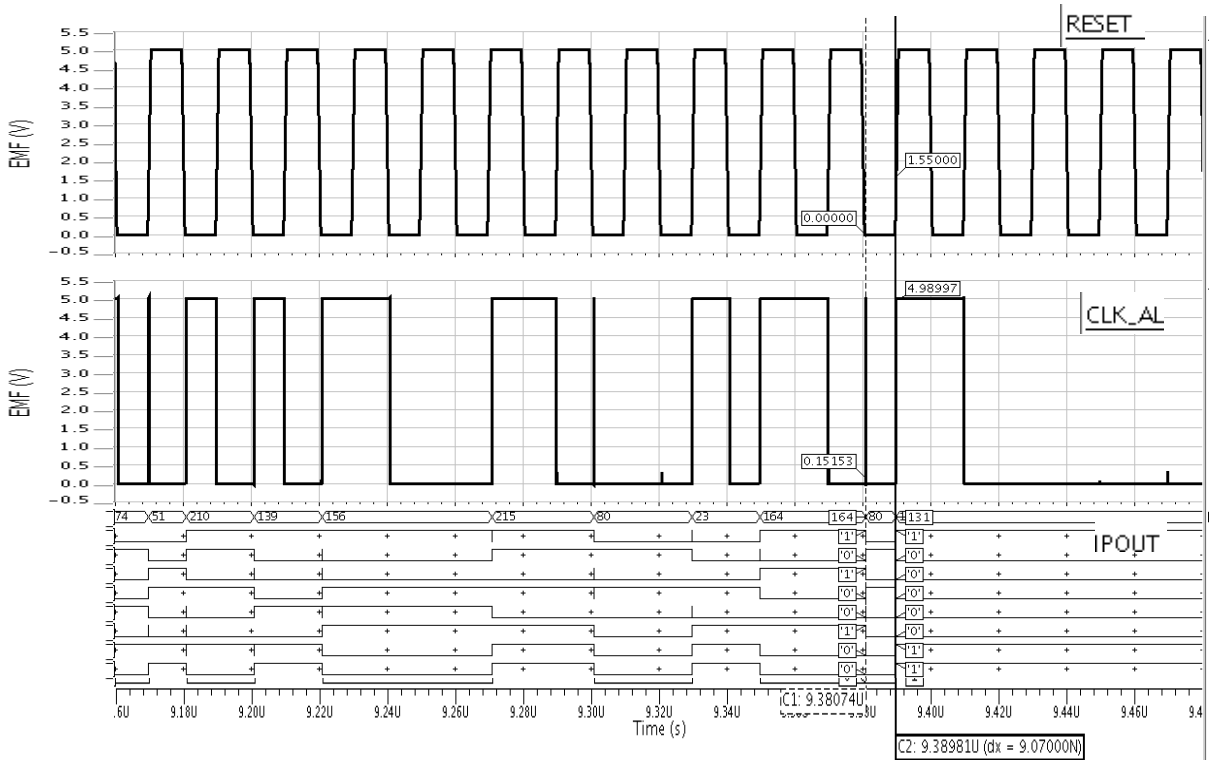


Figura 4.24. Ampliación visual de los resultados de simulación del PRNG mejorado.

Acorde a los resultados de la figura 4.24, si se descartan los pulsos menores a 1ns ya mencionados en la parte final de la sección 4.1, los diferentes periodos de los niveles lógicos de la señal del OAB van desde los 10ns hasta los 70ns en pasos de 10ns, es decir el conjunto de lapsos posibles es discreto. Por lo cual se deduce que el periodo de cada nivel lógico de la señal del OAB es un múltiplo entero de 1/2Fs, donde Fs es la frecuencia de muestreo (RESET). Para este caso al ser Fs = 50MHz el mínimo periodo posible es de 10ns como se demuestra en la figura anterior. De acuerdo a estos resultados, se deducen las siguientes características del OAB:

Si Fs es la frecuencia de muestreo (RESET) en la etapa del retenedor, entonces:

- El periodo mínimo posible entre flancos de subida es igual a 1/2Fs.
- El periodo mínimo posible de cada nivel lógico es igual a 1/2Fs, exceptuando los pulsos menores a 1ns.

De esta forma, los pulsos menores a 1ns son los causantes del periodo mínimo posible entre flancos de subida de 1/2Fs y no representan un error de funcionamiento.

Conforme a la señal de salida del PRNG mejorado en la figura 4.24, los diferentes periodos de cada número van desde los 10ns hasta los 90ns en pasos de 10ns, es decir al igual que ocurre con la señal del OAB el conjunto de lapsos posibles es discreto. Así se comprueba que también el periodo de cada número generado es un múltiplo entero de 1/2Fs. Al observar el valor emitido del PRNG mejorado entre los tiempos 9.38µs y 9.389µs, su duración de aproximadamente 10ns se debe al pulso menor a 1ns. Esta prueba demuestra cómo tales pulsos menores originan el periodo mínimo posible de 1/2Fs y por ello son factores contribuyentes. Un aspecto muy importante es que a pesar de ser un proceso probabilístico se tiene un control sobre la mínima duración de cada número con únicamente una señal periódica de frecuencia Fs (RESET) y componentes completamente digitales. Esto además conlleva a una facilidad de integración con otros sistemas similares, sin el riesgo de alterar su funcionalidad por efectos de polarización [1].

Como es sabido, la FFT de una onda continua periódica es un conjunto de impulsos característicos ubicados en las frecuencias que componen la señal. Para el caso de las salidas del OAB y el PRNG mejorado no mostraron impulsos sobresalientes que representen cierto conjunto de frecuencias. Por lo cual y considerando los resultados de la figura 4.23, el comportamiento del OAB y el PRNG mejorado es aleatorio. Las gráficas de magnitud en dB de la FFT se exponen a continuación.

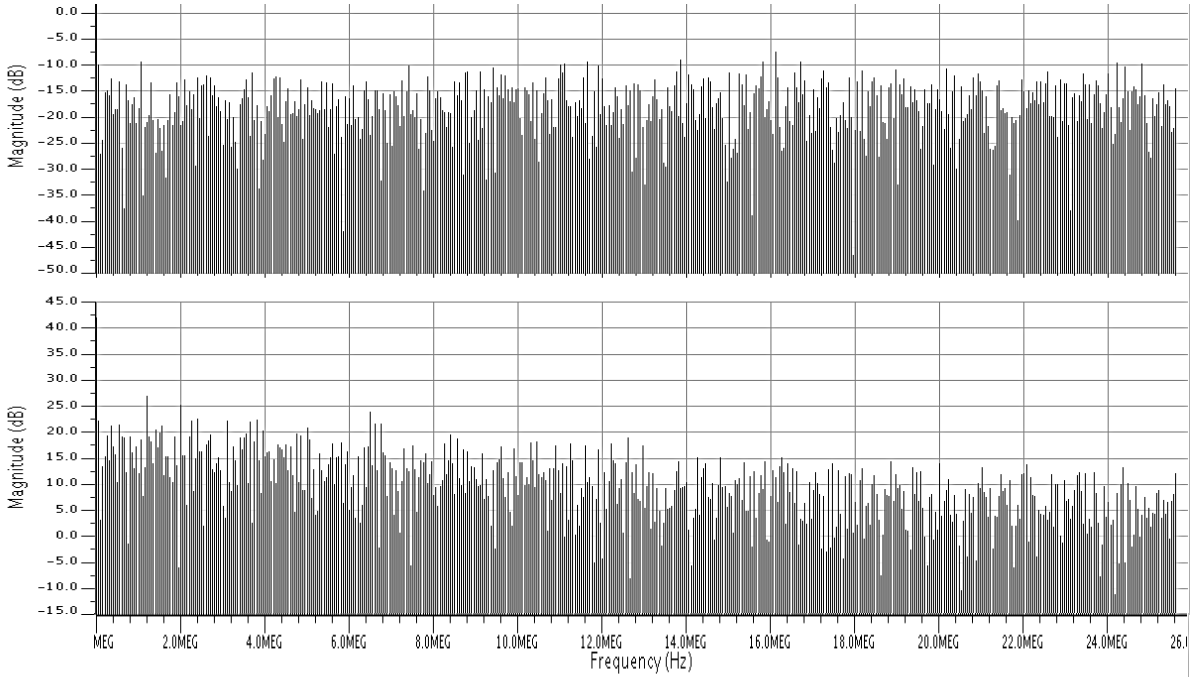


Figura 4.25. Magnitud en dB de la FFT de las señales del OAB y el PRNG mejorado.

Un histograma es una representación gráfica que ofrece información sobre la distribución de la población o la muestra dentro de un conjunto de valores posibles, es decir indica la cantidad de muestras pertenecientes a conjuntos específicos. A continuación se presenta el histograma de los valores emitidos por el OAB y el PRNG mejorado.

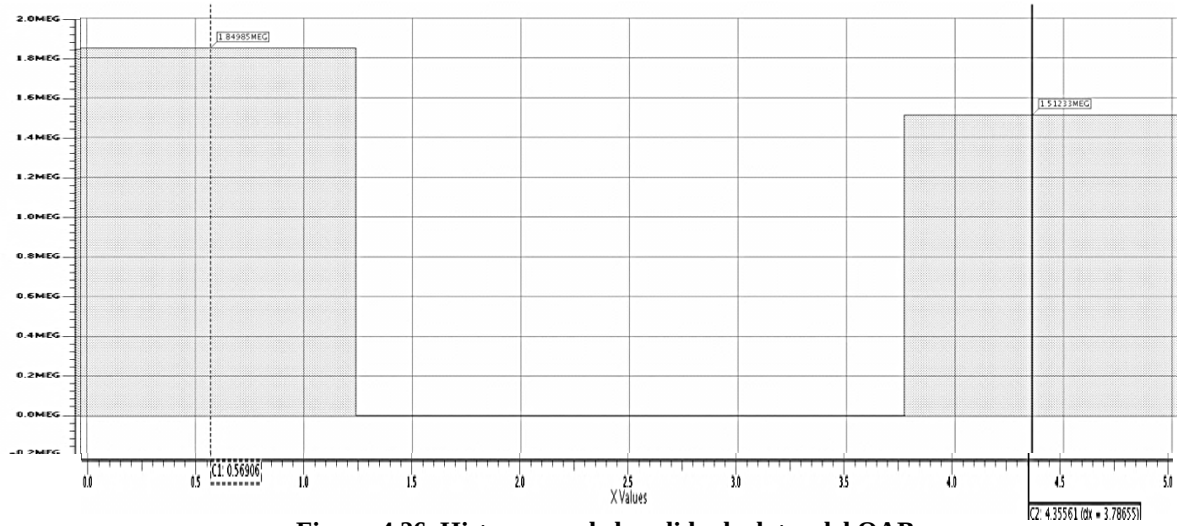


Figura 4.26. Histograma de la salida de datos del OAB.

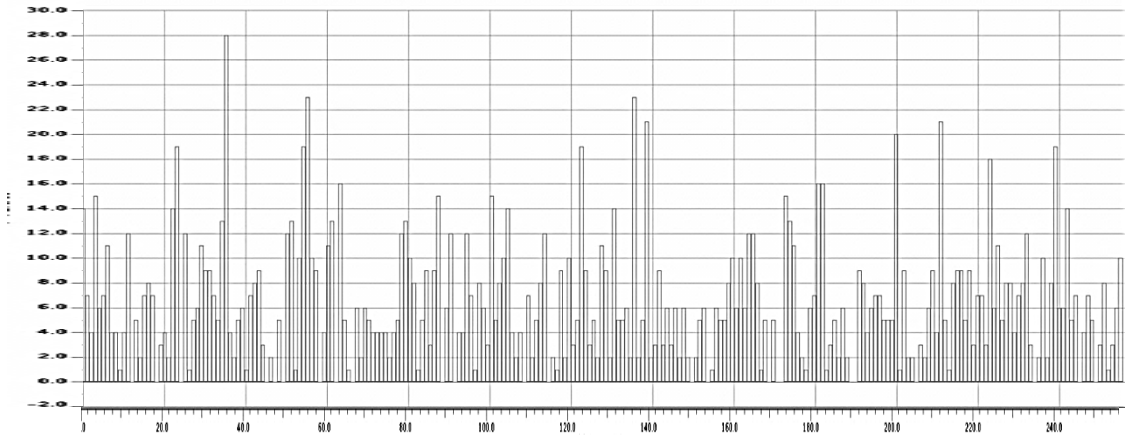


Figura 4.27. Histograma de la salida de datos del PRNG mejorado.

Conforme al histograma del OAB, de un total de 3 362 180 muestras, el 55% corresponden a ceros lógicos y el 45% a unos lógicos, por lo cual se considera una distribución uniforme. Respecto al histograma del PRNG mejorado, con tal cantidad de muestras no presenta una distribución uniforme, esto se debe a los distintos periodos de cada valor emitido, es decir cuando en alguno su duración es mayor a 1/2Fs se considera como una repetición del mismo y por ello una mayor concurrencia. Sin embargo, conforme se aumenta el número de muestras el histograma tiende a ser una distribución uniforme, característica de un proceso aleatorio.

4.4 Resumen

Con base en el análisis de resultados del PRNG del capítulo 3 surgió una propuesta para mejorar sus características aleatorias. Esta consistió en controlar la generación de números por medio de una señal aleatoria booleana y así producir valores con duraciones variables, para lo cual se diseñó un Oscilador Aleatorio Booleano (OAB) que consiste en un circuito completamente digital cuyo principio de operación es la oscilación asíncrona. Tal circuito se conformó por 4 etapas: un oscilador asíncrono para generar una señal de tal naturaleza; un retardador para convertir la señal asíncrona periódica a aleatoria; un retenedor para muestrear posibles pulsos de la salida del retardador con una señal de muestreo de frecuencia Fs y finalmente un generador de pulsos para construir la señal digital aleatoria definitiva. Posteriormente el OAB se integró con el PRNG en modelo Verilog AMS para permitir la implementación y simulación en señal mixta. De acuerdo a los resultados en el tiempo, tanto la señal del OAB como la del PRNG mejorado no presentaron un comportamiento periódico, esto también fue comprobado por medio de la FFT de ambas señales. En tal análisis no se hallaron frecuencias características que sugirieran periodicidad. Se dedujo la duración de cada nivel lógico y número emitido por el OAB y el PRNG mejorado respectivamente, siendo esta un múltiplo entero de 1/2Fs. Al crear un histograma de las señales anteriores se demostró la distribución uniforme del OAB y no uniforme del PRNG mejorado.

Capítulo 5

Conclusiones y trabajo a futuro

De acuerdo a los análisis de resultados obtenidos en esta investigación, se ha demostrado la posibilidad de implementar un sistema digital conformado por bloques descritos en HDL y a nivel transistor, sin tareas intermedias, junto con la ejecución de un proceso de síntesis digital por medio del paquete de software *Menthor Graphics* y sus diferentes plataformas: *VSIM*, *Leonardo Spectrum*, editor de esquemáticos y de *layout* del Kit de Diseño de ASIC (ADK).

De acuerdo al capítulo 3, se logró describir un PRNG en lenguaje VerilogA, por medio de la plataforma *VSIM*. En él se describió, compiló y simuló exitosamente, con resultados concordantes a los esperados conforme a [2]. El estilo comportamental demostró ser el más eficaz para este tipo de arquitecturas complicadas, sobre todo por permitir con éxito el proceso de la síntesis digital.

Conforme al mismo capítulo del párrafo anterior, se logró diseñar un PRNG en un ASIC, concretado en sus tres modelos respectivos: comportamental, estructural y físico. De la tabla 3.4, las celdas con mayores instancias fueron la compuerta xnor (xnor2) y el flip - flop D (dffr) por ser unidades básicas en la implementación de sumadores y registros de entrada – salida paralela, las cuales conformaron el 50% de toda la arquitectura. A su vez estas celdas abarcan también el mismo porcentaje en área del núcleo del circuito integrado.

Según el análisis de resultados del capítulo 3 y a la tabla 2.1, el rango de velocidad del PRNG del presente trabajo es menor a 125MSa/s lo que conlleva a ser mayor a la del PRNG sintetizado en FPGA de 90.98 MSa/s. Así, la velocidad del PRNG en ASIC puede ser superior a la del FPGA en un porcentaje no mayor al 37%. En comparación a las implementaciones de [11], [12] y [13], la velocidad del PRNG de este trabajo es mayor en una razón no mayor a 5.1 veces.

Respecto al número de celdas empleadas, el diseño en ASIC con herramientas de *Menthor Graphics* requirió 438, cantidad menor al desarrollado en [2] en *Cadence Silicon Ensemble*, de 474, ambos con el mismo proceso tecnológico CMOS de 0.35µm. Lo anterior implicó un ahorro del 7.5% de recursos.

En el capítulo 4 se diseñó un Oscilador Aleatorio Booleano a nivel transistor para mejorar las características aleatorias del PRNG estudiado en el capítulo 2. De dicho oscilador se comprobó la generación de secuencias aleatorias de bits con distribución uniforme por medio de una arquitectura completamente digital, un funcionamiento asíncrono y el aprovechamiento de los retardos de propagación. Se dedujo el periodo de cada nivel lógico, siendo éste igual a un múltiplo entero de 1/2Fs, exceptuando los pulsos menores a 1ns, donde Fs es la frecuencia de muestreo de la señal RESET en la etapa del retenedor.

Así mismo, se logró caracterizar el PRNG mejorado por medio de un *testbench* en señal mixta en el entorno de simulación ADMS. Conforme a los resultados mostrados, a lo largo de 20µs no se encontraron patrones repetitivos además de la inexistencia de frecuencias características que sugirieran periodicidad, por lo cual se demostró su naturaleza aleatoria. De manera similar al OAB, se comprobó el periodo de cada valor emitido, siendo éste un múltiplo entero de 1/2Fs. Para el número de muestras obtenido, el histograma del PRNG mejorado demostró una distribución de datos no uniforme, sin embargo al aumentar el número de muestras la distribución tiende a ser uniforme, característica de un proceso aleatorio.

A lo largo de la investigación se presentaron algunas limitaciones que prolongaron más el tiempo de desarrollo, como fueron la existencia de escasas referencias para manejar las herramientas de *Menthor Graphics*, sin embargo todas ellas en desactualización respecto a la versión actual del software. En la generación del *layout*, la construcción manual de algunas conexiones también demoró el proceso.

Conforme a los resultados obtenidos y las limitaciones presentadas, se propone como trabajo a futuro:

- Redactar un manual sobre el uso exclusivo del sintetizador *Leonardo Spectrum*, puesto que las referencias existentes son escasas y desactualizadas. El proceso de síntesis es muy importante ya que desde dicho entorno pueden mejorarse en gran medida algunos parámetros del circuito como el número de celdas empleadas y la velocidad.

- Mejorar el algoritmo para trazar las conexiones en el *layout* para disminuir el número de pistas a construir manualmente.

- Utilizar baterías de prueba como las mencionadas en [2] para la caracterización del TRNG.

Referencias

[1] Piotr Wieczorek and Krzysztof Golofit, "Dual-Metastability Time-Competitive True Random Number Generator", *IEEE Trans. Circuits Syst. I, Exp. Briefs*, vol. 16, no. 1, January 2014, p. 134.

[2] Victor R. Gonzalez-Díaz *et al*, "A Pseudorandom Number Generator Based on Time-Variant Recursion of Accumulators", *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 58, no. 9, September 2011, pp. 580-584.

[3] Grover Brown and Patrick Robert Y.C. Hwang, *"Introduction to random signals and applied Kalman filtering: with Matlab exercises"*, 4th ed. John Wiles & Sons, Inc. USA 2012, pp. 128.

[4] Ahmed B. Mahmood and Robert D. Dony, "Adaptive Encryption Using Pseudo-Noise Sequences for Medical Images", *The 3rd International Conference on Communications and Information Technology (ICCT-2013), Digital Information Management & Security*, Beirut, pp. 39.

[5] Jan M. Rabaey, Anantha Chandrakasan and Borivoje Nikolić. *"Digital integrated circuits. A design perspective"*, 2nd ed. Prentice Hall Electronics and VLSI series, Charles G. Sodini series editor, Upper Saddle River, New Jersey, USA 2003. pp. 383-385, 397, 435.

[6] Peter J. Ashenden, Gregory D. Peterson and Darrel A. Teegarden, *"The system designer´s guide to VHDL-AMS. Analog, mixed-signal, and mixed-technology modeling"*, Morgan Kaufman Publishers, San Francisco, California, USA 2003. pp. 2-5.

[7] Mànuel Antoni *et al*, *"Instrumentación virtual. Adquisición de datos, procesado y análisis de señales*", Alfaomega grupo editor S.A. de C.V. México DF 2002, pp. 20-21.

[8] Xiaole Fang *et al*, "Noise and Chaos Contributions in Fast Random Bit Sequence Generated from Broadband Optoelectronic Entropy Sources", *IEEE Trans. Circuits. Syst. I, Regular Papers*, vol. 0, no. , "to be published", p. 1-2.

[9] Harald Niederreiter, *"Random number generation and Quasi-Monte Carlo methods"*, Society for Industrial and Applied Mathematics, Philadelphia, Pennsylvania, 1992, pp. 161-165.

[10] Viktor Fischer, Florent Bernard and Patrick Haddad, "An Open-Source Multi-FPGA Modular System for Fair Benchmarking of True Random Number Generators". 1-2.

[11] Xiang Tian and Khaled Benkrid, "Mensernne Twister Random Number Generation of FPGA, CPU and GPU", in *Proc. NASA/ESA Conf. AHS*, August 2009, pp. 460-464.

[12] S. Chandrasekaran and A. Amira, "High performance FPGA implementation of the Mersenne Twister", in *Proc. IEEE Int. Symp. Electron. Des., Test Appl. (DELTA)*, January 2008, pp. 482-485.

[13] V. Sriram and D. Kearney, "An área time efficient field programable Mersenne Twister uniform random number generator", in *Proc. Int. Conf. Eng. Reconfigurable Syst.*, August 2006, pp. 244-246.

[14] Antonio Rubio *et al*, "*Diseño de circuitos y sistemas integrados"*, Alfaomega, México 2005, p. 448.

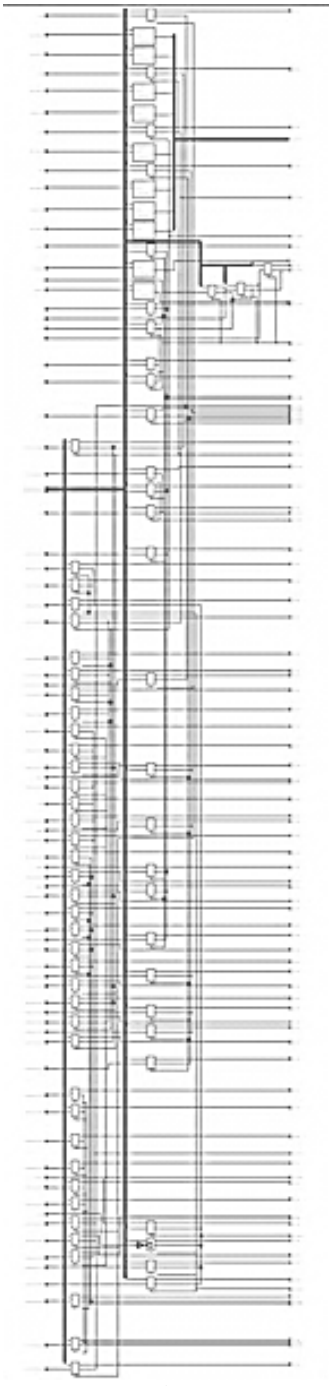
[15] Jean-Pierre Deschamps, *"Diseño de Circuitos Integrados de Aplicación Específica ASIC"*, Paraninfo, España 1994, p. 1-5.

[16] Benemérita Universidad Autónoma de Puebla, *"Manual de diseño de sistemas digitales en ASIC, con Menthor Graphics"*, Facultad de Ciencias de la Electrónica, Puebla, México, 2014.

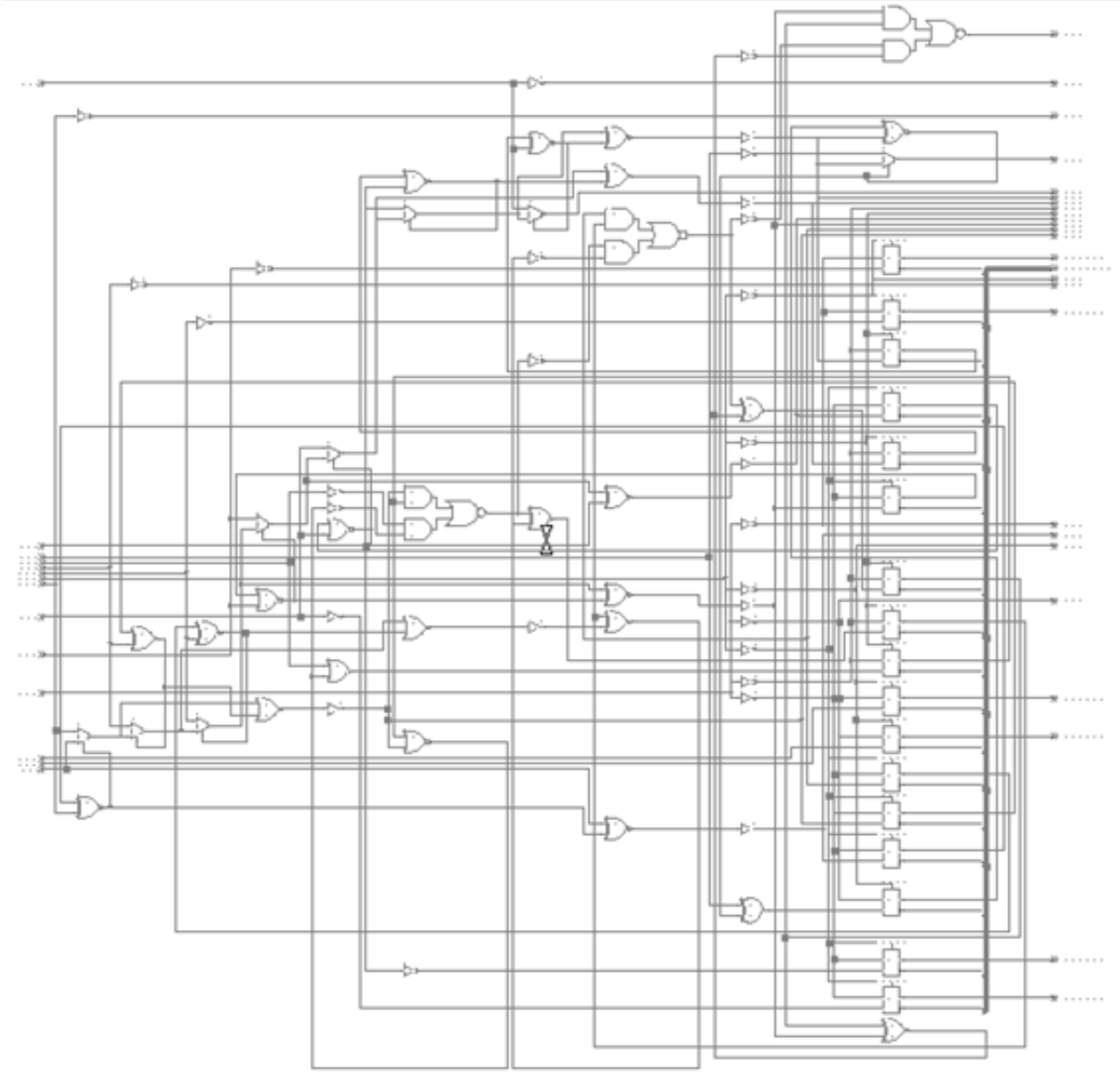
[17] Menthor Graphics Corp., *Pyxis Self Paced Tutorial*, Menthor Graphics, February 2012.

Anexos

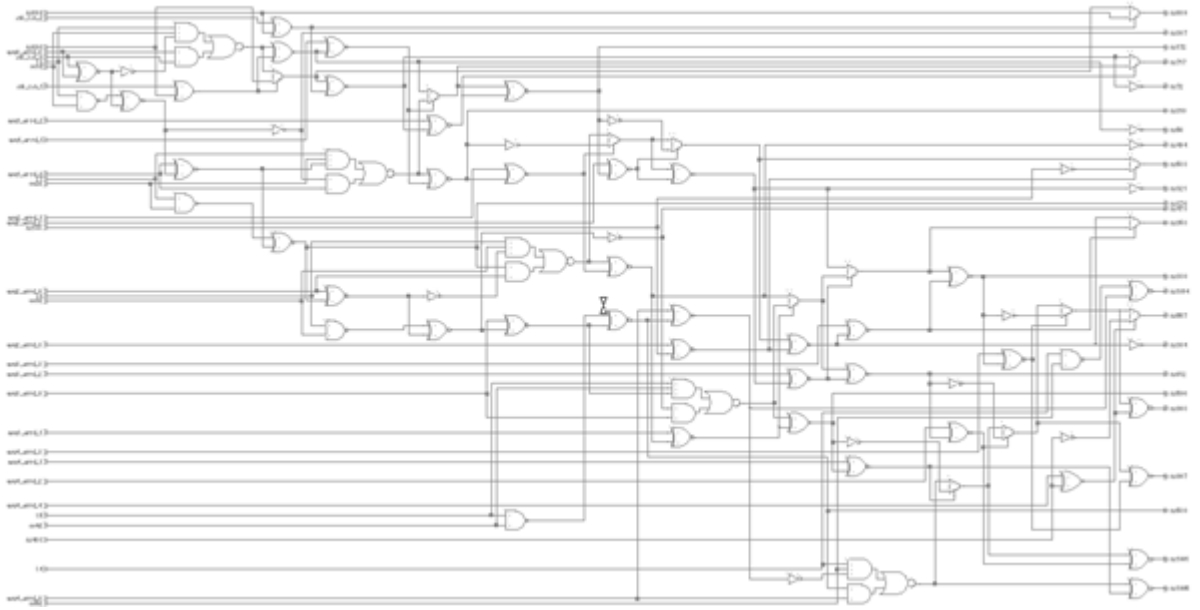
Anexo A. Hojas esquemáticas correspondientes al PRNG a nivel transistor



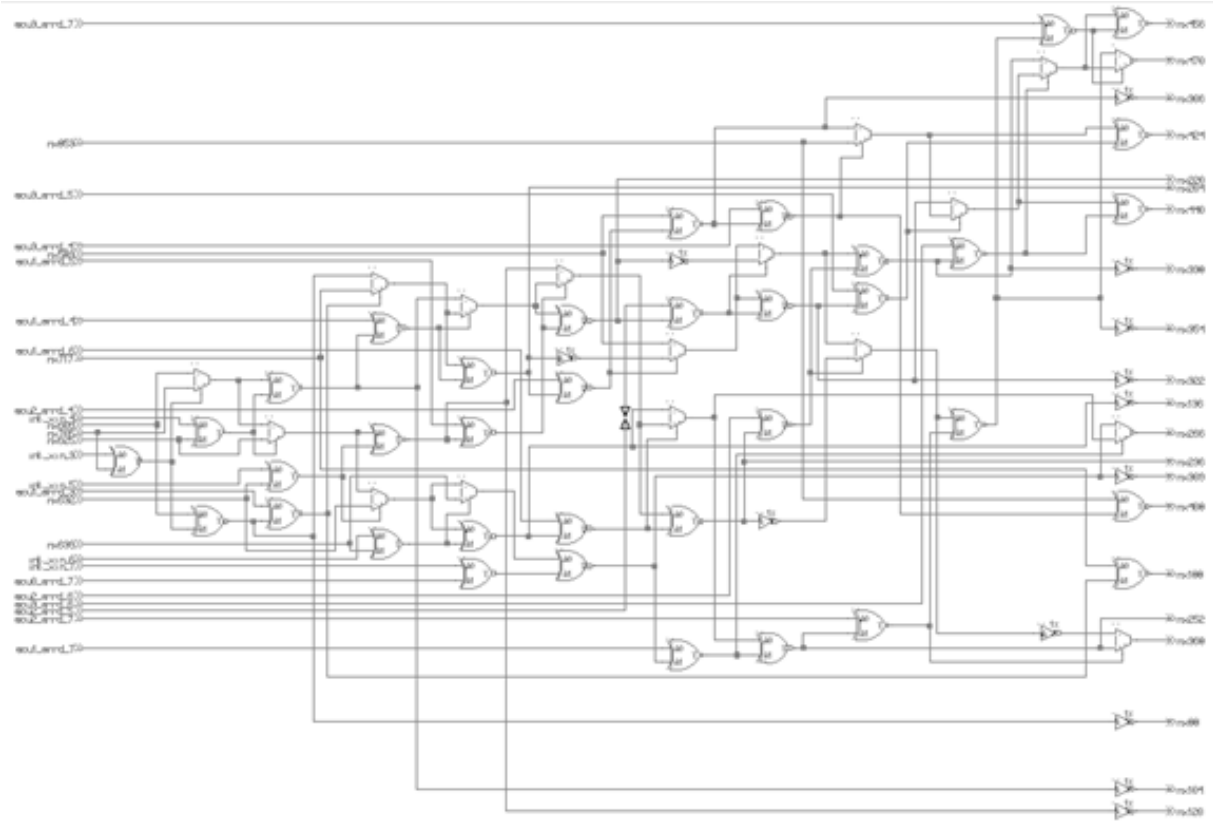
Hoja 1



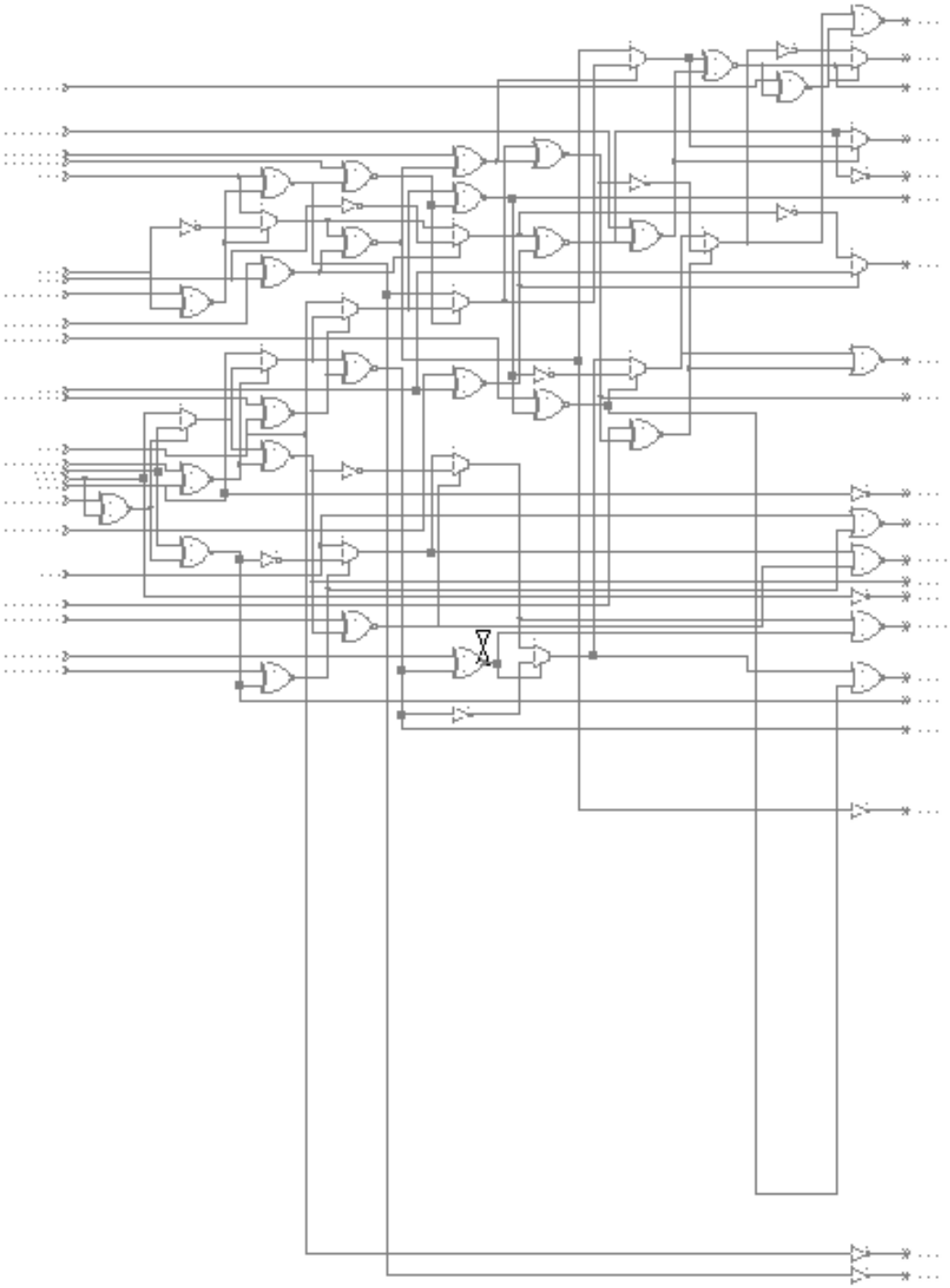
Hoja 2



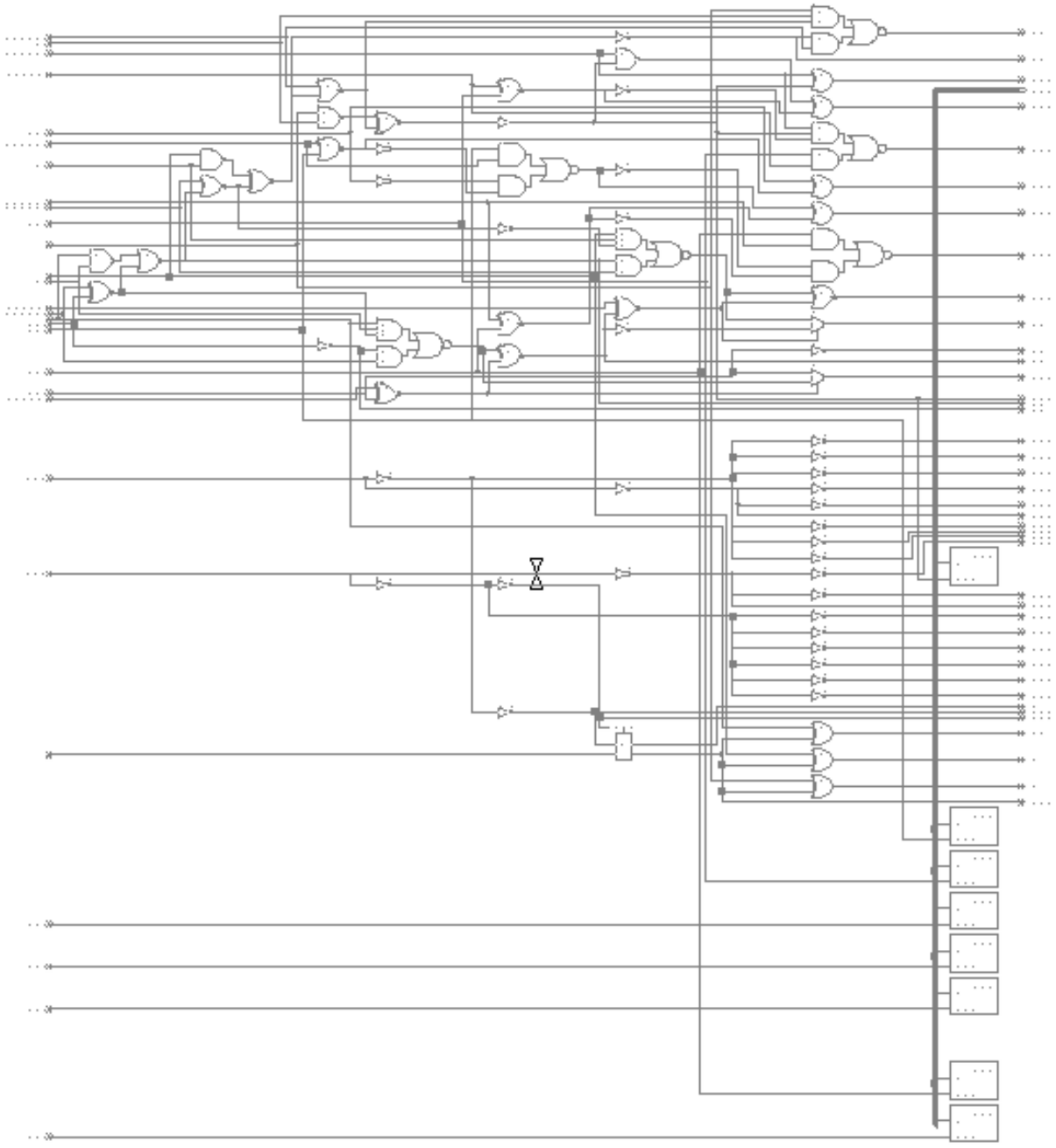
Hoja 3



Hoja 4



Hoja 5



Hoja 6

Anexo B. Caminos críticos de la síntesis digital

```
                    Critical Path Report

Critical path #1, (unconstrained path)
NAME                              GATE          ARRIVAL              LOAD
-------------------------------------------------------------------------
clock information not specified
delay thru clock network                        0.00 (ideal)


acu0_reg1_reg_q(0)/Q              dffr      0.00  0.54 dn             0.03
ix688/Y                           xnor2     0.20  0.74 up             0.03
ix45/Y                            inv01     0.11  0.85 dn             0.01
ix614/Y                           aoi32     0.52  1.38 up             0.04
ix724/Y                           xnor2     0.36  1.74 dn             0.05
ix163/Y                           xnor2     0.31  2.04 up             0.04
ix393/Y                           xnor2     0.28  2.33 dn             0.05
ix816/Y                           xnor2     0.31  2.64 up             0.04
ix588/Y                           mux21_ni  0.50  3.13 dn             0.04
ix586/Y                           mux21_ni  0.41  3.54 dn             0.04
ix856/Y                           xnor2     0.28  3.83 up             0.05
ix399/Y                           xnor2     0.26  4.08 dn             0.04
ix854/Y                           mux21_ni  0.51  4.59 up             0.03
ix409/Y                           xnor2     0.31  4.90 dn             0.05
ix971/Y                           xnor2     0.29  5.20 up             0.04
ix558/Y                           mux21_ni  0.50  5.70 dn             0.04
ix990/Y                           xnor2     0.28  5.98 up             0.05
ix635/Y                           xnor2     0.26  6.24 dn             0.04
ix865/Y                           xnor2     0.31  6.55 up             0.05
ix1100/Y                          xnor2     0.26  6.81 dn             0.04
ix524/Y                           mux21_ni  0.53  7.34 up             0.04
ix1116/Y                          xnor2     0.30  7.64 dn             0.05
ix1085/Y                          xnor2     0.29  7.93 up             0.04
ix1114/Y                          mux21_ni  0.49  8.42 dn             0.04
ix1095/Y                          xnor2     0.18  8.61 up             0.01
ix1240/Y                          buf04     0.50  9.10 up             0.30
io_pout(7)/Pad                    PadOut    1.56  10.66 up             0.00
pout(7)/                                    0.00  10.66 up             0.00
data arrival time                                 10.66

data required time                              not specified
-------------------------------------------------------------------------
data required time                              not specified
data arrival time                                  10.66
                                                ----------
                                              unconstrained path
```

```
Critical path #2, (unconstrained path)
NAME                              GATE              ARRIVAL               LOAD
-------------------------------------------------------------------------------
clock information not specified
delay thru clock network                            0.00 (ideal)


acu0_reg1_reg_q(0)/Q              dffr        0.00  0.54 dn              0.03
ix688/Y                           xnor2       0.20  0.74 up              0.03
ix45/Y                            inv01       0.11  0.85 dn              0.01
ix614/Y                           aoi32       0.52  1.38 up              0.04
ix724/Y                           xnor2       0.36  1.74 dn              0.05
ix163/Y                           xnor2       0.31  2.04 up              0.04
ix393/Y                           xnor2       0.28  2.33 dn              0.05
ix816/Y                           xnor2       0.31  2.64 up              0.04
ix588/Y                           mux21_ni    0.50  3.13 dn              0.04
ix860/Y                           xnor2       0.28  3.42 up              0.05
ix619/Y                           xnor2       0.26  3.68 dn              0.04
ix858/Y                           mux21_ni    0.51  4.19 up              0.03
ix629/Y                           xnor2       0.30  4.13 up              0.05
ix962/Y                           xnor2       0.26  4.39 dn              0.04
ix560/Y                           mux21_ni    0.53  4.92 up              0.04
ix558/Y                           mux21_ni    0.41  5.70 dn              0.04
ix990/Y                           xnor2       0.28  5.98 up              0.05
ix635/Y                           xnor2       0.26  6.24 dn              0.04
ix865/Y                           xnor2       0.31  6.55 up              0.05
ix1100/Y                          xnor2       0.26  6.81 dn              0.04
ix524/Y                           mux21_ni    0.53  7.34 up              0.04
ix1116/Y                          xnor2       0.30  7.64 dn              0.05
ix1085/Y                          xnor2       0.29  7.93 up              0.04
ix1114/Y                          mux21_ni    0.49  8.42 dn              0.04
ix1095/Y                          xnor2       0.18  8.60 up              0.01
ix1240/Y                          buf04       0.50  9.10 up              0.30
io_pout(7)/Pad                    PadOut      1.56  10.66 up              0.00
pout(7)/                                      0.00  10.66 up              0.00
data arrival time                                   10.66

data required time                                not specified
-------------------------------------------------------------------------------
data required time                                not specified
data arrival time                                   10.66
                                                  ----------
                                                  unconstrained path
-------------------------------------------------------------------------------
```

```
Critical path #3, (unconstrained path)
NAME                                GATE              ARRIVAL               LOAD
-------------------------------------------------------------------------------
clock information not specified
delay thru clock network                              0.00 (ideal)


acu0_reg1_reg_q(0)/Q                dffr        0.00  0.54 dn              0.03
ix688/Y                             xnor2       0.20  0.74 up              0.03
ix45/Y                              inv01       0.11  0.85 dn              0.01
ix614/Y                             aoi32       0.52  1.38 up              0.04
ix724/Y                             xnor2       0.36  1.74 dn              0.05
ix163/Y                             xnor2       0.31  2.04 up              0.04
ix393/Y                             xnor2       0.28  2.33 dn              0.05
ix816/Y                             xnor2       0.31  2.64 up              0.04
ix864/Y                             xnor2       0.27  2.90 dn              0.05
ix603/Y                             xnor2       0.31  3.21 up              0.04
ix862/Y                             mux21_ni    0.48  3.69 dn              0.03
ix613/Y                             xnor2       0.30  3.99 up              0.05
ix954/Y                             xnor2       0.26  4.25 dn              0.04
ix562/Y                             mux21_ni    0.53  4.78 up              0.04
ix998/Y                             xnor2       0.30  5.08 dn              0.05
ix839/Y                             xnor2       0.29  5.37 up              0.04
ix996/Y                             mux21_ni    0.48  5.85 dn              0.03
ix849/Y                             xnor2       0.30  6.15 up              0.05
ix1090/Y                            xnor2       0.26  6.41 dn              0.04
ix526/Y                             mux21_ni    0.53  6.94 up              0.04
ix1120/Y                            xnor2       0.30  7.24 dn              0.05
ix1069/Y                            xnor2       0.29  7.53 up              0.04
ix1118/Y                            mux21_ni    0.48  8.02 dn              0.03
ix1114/Y                            mux21_ni    0.40  8.41 dn              0.04
ix1095/Y                            xnor2       0.18  8.59 up              0.01
ix1240/Y                            buf04       0.50  9.09 up              0.30
io_pout(7)/Pad                      PadOut      1.56  10.65 up              0.00
pout(7)/                                        0.00  10.65 up              0.00
data arrival time                                     10.65

data required time                                 not specified
-------------------------------------------------------------------------------
data required time                                 not specified
data arrival time                                     10.65
                                                   ----------
                                                   unconstrained path
-------------------------------------------------------------------------------
```

```
Critical path #4, (unconstrained path)
NAME                                GATE              ARRIVAL               LOAD
-------------------------------------------------------------------------------
clock information not specified
delay thru clock network                              0.00 (ideal)


acu0_reg1_reg_q(0)/Q                dffr        0.00  0.54 dn              0.03
ix688/Y                             xnor2       0.20  0.74 up              0.03
ix45/Y                              inv01       0.11  0.85 dn              0.01
ix614/Y                             aoi32       0.52  1.38 up              0.04
ix724/Y                             xnor2       0.36  1.74 dn              0.05
ix163/Y                             xnor2       0.31  2.04 up              0.04
ix393/Y                             xnor2       0.28  2.33 dn              0.05
ix816/Y                             xnor2       0.31  2.64 up              0.04
ix588/Y                             mux21_ni    0.50  3.13 dn              0.04
ix860/Y                             xnor2       0.28  3.42 up              0.05
ix619/Y                             xnor2       0.26  3.68 dn              0.04
ix858/Y                             mux21_ni    0.51  4.19 up              0.03
ix629/Y                             xnor2       0.30  4.13 up              0.05
ix962/Y                             xnor2       0.26  4.39 dn              0.04
ix998/Y                             xnor2       0.27  5.06 dn              0.05
ix839/Y                             xnor2       0.29  5.35 up              0.04
ix996/Y                             mux21_ni    0.48  5.83 dn              0.03
ix849/Y                             xnor2       0.30  6.13 up              0.05
ix1090/Y                            xnor2       0.26  6.39 dn              0.04
ix526/Y                             mux21_ni    0.53  6.92 up              0.04
ix1120/Y                            xnor2       0.30  7.22 dn              0.05
ix1069/Y                            xnor2       0.29  7.51 up              0.04
ix1118/Y                            mux21_ni    0.48  7.99 dn              0.03
ix1114/Y                            mux21_ni    0.40  8.39 dn              0.04
ix1095/Y                            xnor2       0.18  8.57 up              0.01
ix1240/Y                            buf04       0.50  9.07 up              0.30
io_pout(7)/Pad                      PadOut      1.56  10.63 up              0.00
pout(7)/                                        0.00  10.63 up              0.00
data arrival time                                     10.63

data required time                                 not specified
-------------------------------------------------------------------------------
data required time                                 not specified
data arrival time                                     10.63
                                                   ----------
                                                   unconstrained path
-------------------------------------------------------------------------------
```

```
Critical path #5, (unconstrained path)
NAME                                GATE              ARRIVAL                LOAD
-------------------------------------------------------------------------------
clock information not specified
delay thru clock network                              0.00 (ideal)


acu0_reg1_reg_q(0)/Q                dffr        0.00  0.54 dn                0.03
ix688/Y                             xnor2       0.20  0.74 up                0.03
ix45/Y                              inv01       0.11  0.85 dn                0.01
ix614/Y                             aoi32       0.52  1.38 up                0.04
ix724/Y                             xnor2       0.36  1.74 dn                0.05
ix163/Y                             xnor2       0.31  2.04 up                0.04
ix393/Y                             xnor2       0.28  2.33 dn                0.05
ix816/Y                             xnor2       0.31  2.64 up                0.04
ix588/Y                             mux21_ni    0.50  3.13 dn                0.04
ix860/Y                             xnor2       0.28  3.42 up                0.05
ix619/Y                             xnor2       0.26  3.68 dn                0.04
ix858/Y                             mux21_ni    0.51  4.19 up                0.03
ix629/Y                             xnor2       0.30  4.13 up                0.05
ix962/Y                             xnor2       0.26  4.39 dn                0.04
ix560/Y                             mux21_ni    0.53  4.92 up                0.04
ix994/Y                             xnor2       0.30  5.22 dn                0.05
ix855/Y                             xnor2       0.29  5.51 up                0.04
ix992/Y                             mux21_ni    0.48  6.00 dn                0.03
ix865/Y                             xnor2       0.30  6.29 up                0.05
ix1100/Y                            xnor2       0.26  6.56 dn                0.04
ix1120/Y                            xnor2       0.27  7.21 dn                0.05
ix1069/Y                            xnor2       0.29  7.50 up                0.04
ix1118/Y                            mux21_ni    0.48  7.99 dn                0.03
ix1114/Y                            mux21_ni    0.40  8.38 dn                0.04
ix1095/Y                            xnor2       0.18  8.56 up                0.01
ix1240/Y                            buf04       0.50  9.06 up                0.30
io_pout(7)/Pad                      PadOut      1.56  10.62 up                0.00
pout(7)/                                        0.00  10.62 up                0.00
data arrival time                                     10.62

data required time                                 not specified
-------------------------------------------------------------------------------
data required time                                 not specified
data arrival time                                     10.62
                                                   ----------
                                                  unconstrained path
-------------------------------------------------------------------------------
```

```
Critical path #6, (unconstrained path)
NAME                                GATE              ARRIVAL                LOAD
-------------------------------------------------------------------------------
clock information not specified
delay thru clock network                              0.00 (ideal)


acu0_reg1_reg_q(0)/Q                dffr        0.00  0.54 dn              0.03
ix688/Y                             xnor2       0.20  0.74 up              0.03
ix45/Y                              inv01       0.11  0.85 dn              0.01
ix614/Y                             aoi32       0.52  1.38 up              0.04
ix724/Y                             xnor2       0.36  1.74 dn              0.05
ix163/Y                             xnor2       0.31  2.04 up              0.04
ix722/Y                             mux21_ni    0.48  2.52 dn              0.03
ix173/Y                             xnor2       0.30  2.82 up              0.05
ix824/Y                             xnor2       0.26  3.08 dn              0.04
ix586/Y                             mux21_ni    0.50  3.28 dn              0.04
ix856/Y                             xnor2       0.28  3.56 up              0.05
ix399/Y                             xnor2       0.26  3.82 dn              0.04
ix854/Y                             mux21_ni    0.51  4.33 up              0.03
ix409/Y                             xnor2       0.31  4.64 dn              0.05
ix971/Y                             xnor2       0.29  4.93 up              0.04
ix558/Y                             mux21_ni    0.50  5.43 dn              0.04
ix990/Y                             xnor2       0.28  5.72 up              0.05
ix635/Y                             xnor2       0.26  5.97 dn              0.04
ix865/Y                             xnor2       0.31  6.28 up              0.05
ix1100/Y                            xnor2       0.26  6.54 dn              0.04
ix1120/Y                            xnor2       0.27  7.21 dn              0.05
ix1069/Y                            xnor2       0.29  7.50 up              0.04
ix1118/Y                            mux21_ni    0.48  7.98 dn              0.03
ix1114/Y                            mux21_ni    0.40  8.38 dn              0.04
ix1095/Y                            xnor2       0.18  8.56 up              0.01
ix1240/Y                            buf04       0.50  9.06 up              0.30
io_pout(7)/Pad                      PadOut      1.56  10.61 up              0.00
pout(7)/                                        0.00  10.61 up              0.00
data arrival time                                     10.61

data required time                                 not specified
-------------------------------------------------------------------------------
data required time                                 not specified
data arrival time                                     10.61
                                                   ----------
                                                   unconstrained path
-------------------------------------------------------------------------------
```

```
Critical path #7, (unconstrained path)
NAME                                GATE              ARRIVAL               LOAD
-------------------------------------------------------------------------------
clock information not specified
delay thru clock network                              0.00 (ideal)


ff0_reg_q/Q                         dffr        0.00  0.52 dn              0.02
ix686/Y                             nand02      0.19  0.71 up              0.02
ix749/Y                             xnor2       0.25  0.96 dn              0.03
ix383/Y                             xnor2       0.26  1.22 up              0.04
ix377/Y                             xnor2       0.28  1.50 dn              0.05
ix810/Y                             xnor2       0.26  1.76 up              0.03
ix485/Y                             inv01       0.11  1.88 dn              0.01
ix590/Y                             aoi32       0.52  2.40 up              0.04
ix864/Y                             xnor2       0.36  2.76 dn              0.05
ix603/Y                             xnor2       0.31  3.07 up              0.04
ix862/Y                             mux21_ni    0.48  3.55 dn              0.03
ix613/Y                             xnor2       0.30  3.85 up              0.05
ix954/Y                             xnor2       0.26  4.11 dn              0.04
ix562/Y                             mux21_ni    0.53  4.64 up              0.04
ix998/Y                             xnor2       0.30  4.94 dn              0.05
ix839/Y                             xnor2       0.29  5.23 up              0.04
ix996/Y                             mux21_ni    0.48  5.71 dn              0.03
ix849/Y                             xnor2       0.30  6.01 up              0.05
ix1090/Y                            xnor2       0.26  6.27 dn              0.04
ix526/Y                             mux21_ni    0.53  6.80 up              0.04
ix524/Y                             mux21_ni    0.40  7.20 up              0.04
ix1116/Y                            xnor2       0.30  7.50 dn              0.05
ix1085/Y                            xnor2       0.29  7.79 up              0.04
ix1114/Y                            mux21_ni    0.49  8.28 dn              0.04
ix1095/Y                            xnor2       0.18  8.46 up              0.01
ix1240/Y                            buf04       0.50  8.96 up              0.30
io_pout(7)/Pad                      PadOut      1.56  10.52 up              0.00
pout(7)/                                        0.00  10.52 up              0.00
data arrival time                                     10.52

data required time                                 not specified
-------------------------------------------------------------------------------
data required time                                 not specified
data arrival time                                     10.52
                                                   ----------
                                                   unconstrained path
-------------------------------------------------------------------------------
```

```
Critical path #8, (unconstrained path)
NAME                                GATE              ARRIVAL               LOAD
-------------------------------------------------------------------------------
clock information not specified
delay thru clock network                              0.00 (ideal)


ff0_reg_q/Q                         dffr        0.00  0.52 dn              0.02
ix686/Y                             nand02      0.19  0.71 up              0.02
ix749/Y                             xnor2       0.25  0.96 dn              0.03
ix383/Y                             xnor2       0.26  1.22 up              0.04
ix377/Y                             xnor2       0.28  1.50 dn              0.05
ix810/Y                             xnor2       0.26  1.76 up              0.03
ix485/Y                             inv01       0.11  1.88 dn              0.01
ix590/Y                             aoi32       0.52  2.40 up              0.04
ix864/Y                             xnor2       0.36  2.76 dn              0.05
ix603/Y                             xnor2       0.31  3.07 up              0.04
ix862/Y                             mux21_ni    0.48  3.55 dn              0.03
ix613/Y                             xnor2       0.30  3.85 up              0.05
ix954/Y                             xnor2       0.26  4.11 dn              0.04
ix562/Y                             mux21_ni    0.53  4.64 up              0.04
ix560/Y                             mux21_ni    0.40  5.03 up              0.04
ix994/Y                             xnor2       0.30  5.33 dn              0.05
ix855/Y                             xnor2       0.29  5.63 up              0.04
ix992/Y                             mux21_ni    0.48  6.11 dn              0.03
ix865/Y                             xnor2       0.30  6.40 up              0.05
ix1100/Y                            xnor2       0.26  6.67 dn              0.04
ix524/Y                             mux21_ni    0.53  7.20 up              0.04
ix1116/Y                            xnor2       0.30  7.50 dn              0.05
ix1085/Y                            xnor2       0.29  7.79 up              0.04
ix1114/Y                            mux21_ni    0.49  8.28 dn              0.04
ix1095/Y                            xnor2       0.18  8.46 up              0.01
ix1240/Y                            buf04       0.50  8.96 up              0.30
io_pout(7)/Pad                      PadOut      1.56  10.52 up              0.00
pout(7)/                                        0.00  10.52 up              0.00
data arrival time                                     10.52

data required time                                 not specified
-------------------------------------------------------------------------------
data required time                                 not specified
data arrival time                                     10.52
                                                   ----------
                                                   unconstrained path
-------------------------------------------------------------------------------
```

```
Critical path #9, (unconstrained path)
NAME                                GATE              ARRIVAL                LOAD
-------------------------------------------------------------------------------
clock information not specified
delay thru clock network                              0.00 (ideal)


ff0_reg_q/Q                         dffr        0.00  0.52 dn               0.02
ix686/Y                             nand02      0.19  0.71 up               0.02
ix749/Y                             xnor2       0.25  0.96 dn               0.03
ix383/Y                             xnor2       0.26  1.22 up               0.04
ix377/Y                             xnor2       0.28  1.50 dn               0.05
ix810/Y                             xnor2       0.26  1.76 up               0.03
ix485/Y                             inv01       0.11  1.88 dn               0.01
ix590/Y                             aoi32       0.52  2.40 up               0.04
ix864/Y                             xnor2       0.36  2.76 dn               0.05
ix603/Y                             xnor2       0.31  3.07 up               0.04
ix862/Y                             mux21_ni    0.48  3.55 dn               0.03
ix613/Y                             xnor2       0.30  3.85 up               0.05
ix954/Y                             xnor2       0.26  4.11 dn               0.04
ix562/Y                             mux21_ni    0.53  4.64 up               0.04
ix998/Y                             xnor2       0.30  4.94 dn               0.05
ix839/Y                             xnor2       0.29  5.23 up               0.04
ix996/Y                             mux21_ni    0.48  5.71 dn               0.03
ix849/Y                             xnor2       0.30  6.01 up               0.05
ix1090/Y                            xnor2       0.26  6.27 dn               0.04
ix526/Y                             mux21_ni    0.53  6.80 up               0.04
ix1120/Y                            xnor2       0.30  7.10 dn               0.05
ix1069/Y                            xnor2       0.29  7.39 up               0.04
ix1118/Y                            mux21_ni    0.48  7.87 dn               0.03
ix1114/Y                            mux21_ni    0.40  8.27 dn               0.04
ix1095/Y                            xnor2       0.18  8.45 up               0.01
ix1240/Y                            buf04       0.50  8.95 up               0.30
io_pout(7)/Pad                      PadOut      1.56  10.51 up               0.00
pout(7)/                                        0.00  10.51 up               0.00
data arrival time                                     10.51

data required time                                 not specified
-------------------------------------------------------------------------------
data required time                                 not specified
data arrival time                                     10.51
                                                   ----------
                                                  unconstrained path
-------------------------------------------------------------------------------
```

```
Critical path #10, (unconstrained path)
NAME                                GATE              ARRIVAL               LOAD
-------------------------------------------------------------------------------
clock information not specified
delay thru clock network                              0.00 (ideal)


lfsr0_ff7_reg_q/Q                   dffr        0.00  0.50 dn              0.01
ix686/Y                             nand02      0.16  0.66 up              0.02
ix749/Y                             xnor2       0.25  0.92 dn              0.03
ix383/Y                             xnor2       0.26  1.18 up              0.04
ix377/Y                             xnor2       0.28  1.46 dn              0.05
ix810/Y                             xnor2       0.26  1.72 up              0.03
ix485/Y                             inv01       0.11  1.83 dn              0.01
ix590/Y                             aoi32       0.52  2.36 up              0.04
ix864/Y                             xnor2       0.36  2.72 dn              0.05
ix603/Y                             xnor2       0.31  3.03 up              0.04
ix862/Y                             mux21_ni    0.48  3.51 dn              0.03
ix613/Y                             xnor2       0.30  3.81 up              0.05
ix954/Y                             xnor2       0.26  4.07 dn              0.04
ix562/Y                             mux21_ni    0.53  4.60 up              0.04
ix998/Y                             xnor2       0.30  4.90 dn              0.05
ix839/Y                             xnor2       0.29  5.19 up              0.04
ix996/Y                             mux21_ni    0.48  5.67 dn              0.03
ix849/Y                             xnor2       0.30  5.97 up              0.05
ix1090/Y                            xnor2       0.26  6.23 dn              0.04
ix526/Y                             mux21_ni    0.53  6.76 up              0.04
ix524/Y                             mux21_ni    0.40  7.15 up              0.04
ix1116/Y                            xnor2       0.30  7.45 dn              0.05
ix1085/Y                            xnor2       0.29  7.74 up              0.04
ix1114/Y                            mux21_ni    0.49  8.24 dn              0.04
ix1095/Y                            xnor2       0.18  8.42 up              0.01
ix1240/Y                            buf04       0.50  8.92 up              0.30
io_pout(7)/Pad                      PadOut      1.56  10.47 up              0.00
pout(7)/                                        0.00  10.47 up              0.00
data arrival time                                     10.47

data required time                                 not specified
-------------------------------------------------------------------------------
data required time                                 not specified
data arrival time                                     10.47
                                                   ----------
                                                   unconstrained path
-------------------------------------------------------------------------------
```

Anexo C. Códigos en *Matlab R2009A* para la simulación del retardador

Código correspondiente a la figura 4.6

```
clear all                                   %borrar variables
clc                                         %limpiar pantalla
ti = 0;                                     %tiempo inicial (ps)
tf = 416;                                   %tiempo final (ps)
muestras = 1000;                            %numero de muestras
%señal S1
A1 = 0.5;                                   %amplitud (adimensional)
f1 = 1/104;                                 %frecuencia (GHz)
rho1 = 0;                                   %desfasamiento (ps)
offset1 = 0.5;                              %nivel de offset (adimensional)
%señal S3
A3 = 0.5;                                   %amplitud (adimensional)
f3 = 1/104;                                 %frecuencia (GHz)
rho3 = 65;                                  %desfasamiento (ps)
offset3 = 0.5;                              %desfasamiento (adimensional)
%-------------------------------------
t = ti:(tf-ti)/muestras:tf;                 %dominio de tiempo
w1 = 2*pi*f1;                               %frecuencia de S1 (rad/s)
w3 = 2*pi*f3;                               %frecuencia de S3 (rad/s)
S1 = A1*square(w1*t+rho1) + offset1;        %señal S1
S3 = A3*square(w3*t+rho3) + offset3;        %señal S3
S5 = xor(S1,S3);                            %señal S5
%---------------Gráficas de S1, S3 y S5----------------------
subplot(3,1,1)
plot(t,S1,'linewidth',3)
axis([ti tf 0 1.5])
title('S1','FontSize',20)
xlabel('tiempo (ps)')
ylabel('valor digital')
grid
subplot(3,1,2)
plot(t,S3,'linewidth',3)
axis([ti tf 0 1.5])
title('S3','FontSize',20)
xlabel('tiempo (ps)')
ylabel('valor digital')
grid
subplot(3,1,3)
plot(t,S5,'linewidth',3)
axis([ti tf 0 1.5])
title('S5','FontSize',20)
xlabel('tiempo (ps)')
ylabel('valor digital')
grid
```

Código correspondiente a la figura 4.9

```
clear all                                   %borrar variables
clc                                         %limpiar pantalla
ti = 0;                                     %tiempo inicial (ps)
tf = 416;                                   %tiempo final (ps)
muestras = 1000;                            %número de muestras
%señal S4
A4 = 0.5;                                   %amplitud (adimensional)
f4 = 1/104;                                 %frecuencia (GHz)
rho4 = 2*65;                                %desfasamiento (ps)
offset4 = 0.5;                              %nivel de offset (adimensional)
%-----------------------------------------------------------
t = ti:(tf-ti)/muestras:tf;                 %rango de tiempo (ps)
w4 = 2*pi*f4;                               %frecuencia de S4 (rad/s)
S4 = A4*square(w4*t+rho4) + offset4;        %señal S4
S5 = ones(1,muestras+1);                    %señal S5
S6 = xor(S5,S4);                            %operación XOR entre S4 y S5
%--------------------Gráficas de S4, S5 y S6-----------------
subplot(3,1,1)
plot(t,S5,'linewidth',3)
axis([ti tf 0 1.5])
title('S5','Fontsize',20)
xlabel('tiempo (ps)')
ylabel('valor digital')
grid
subplot(3,1,2)
plot(t,S4,'linewidth',3)
axis([ti tf 0 1.5])
title('S4','FontSize',20)
xlabel('tiempo (ps)')
ylabel('valor digital')
grid
subplot(3,1,3)
plot(t,S6,'linewidth',3)
axis([ti tf 0 1.5])
title('S6','fontSize',20)
xlabel('tiempo (ps)')
ylabel('valor digital')
grid
```