Diseño, síntesis, fabricación y prueba de un *Hasher* SHA-256 en tecnología CMOS de 180 nm

Fernando Aguirre, Octavio Alpago, Jerónimo Atencio, Alejandro Furfaro, Sebastián Pazos[1]

Universidad Tecnológica Nacional, Facultad Regional Buenos Aires, Departamento de Ingeniería Electrónica, Medrano 951, (C1179AAQ) Ciudad Autónoma de Buenos Aires, Argentina

faguirre@electron.frba.utn.edu.ar

Recibido el 30 de marzo de 2015, aprobado el 10 de junio de 2015

Resumen

Este reporte surge de un proyecto de cooperación con ingenieros y científicos del departamento de *Computer Engineering de la Universidad de Utah*, Estados Unidos y el Laboratorio de Microelectrónica de la Facultad Regional Buenos Aires de la Universidad Tecnológica Nacional (FRBA-UTN). En el mismo se exponen las técnicas utilizadas para la síntesis de un Circuito Integrado (CI)[2] que representa un módulo hasher del algoritmo *Secure Hash Algorithm* (SHA-256) lo cual es una función criptográfica de 256 bits de longitud. Este algoritmo adquirió popularidad con el auge de los *BitCoins* como criptomonedas. Las técnicas asincrónicas son las técnicas metodológicas de diseño más prometedoras. En este trabajo se describe lo realizado en las áreas de lógica, síntesis, *layout, testing* y manufactura. Cabe destacar que un componente esencial de su síntesis física fue el desarrollo de un *kit* de Diseño Físico Interoperable (iPDK) para el cual se utilizaron las herramientas de diseño automático de *Synopsys®* y *Mentor Graphics®*. Por otra parte, la manufactura se realizó a través de MOSIS en un proceso de fabricación CMOS de 180 nm de 6 niveles de metal, provisto por IBM.

PALABRAS CLAVE: CIRCUITOS INTEGRADOS DIGITALES – SINCRÓNICO – ASINCRÓNICO – VERILOG – CMOS - iPDK

Abstract

This work describes the initial steps of a joint collaboration with engineers and scientists of the Computer Engineering Department of the University of Utah, USA. It presents the techniques used in the design and synthesis of an integrated digital circuit (IC). The target IC is an embedded SHA-256 hasher module, which became popular with the noticeable advance of BitCoins as crypto currencies. The asynchronous design methodology is one of the most promising methodologies of IC design. In this article we simply describe our work in the areas of logic and physical synthesis, testing and manufacturing. One essential component for physical synthesis was the creation and adaptation of an Interoperable Physical Design Kit (iPDK) for the Synopsys® CAD environment and Mentor Graphics®. Manufacturing was done through MOSIS, on a 180 nm CMOS process with 6 metal layers from IBM.

KEYWORDS: INTEGRATED DIGITAL CIRCUITS – SYNCHRONOUS – ASYNCHRONOUS – VERILOG – CMOS - iPDK

1 También han participado de la elaboración de este trabajo: Roberto Simone y Roberto Suaya.

2 Al final de este artículo se encuentra un glosario con todas las expresiones y sus acrónimos.

Introducción

En el Laboratorio de Microelectrónica de la FRBA-UTN se estudia el analisis y desarrollo de circuitos integrados digitales y analógicos. Históricamente, sus primeros pasos se centraron en el diseño de circuitos digitales cuya evolución vertiginosa en las cinco últimas décadas, dio como resultado un crecimiento exponencial. Las dos componentes básicas que predominaron durante este lapso y en este área son la existencia de un *clock* maestro global que regula y sincroniza todas las comunicaciones y operaciones dentro del CI, y la metodología de diseño: *top-down*, seguido por una implementación *bottom-up*. Este último requerimiento simple en apariencia, es la llave del éxito frente al trabajo de diseñar circuitos integrados cada vez más complejos. El método permite esconder la complejidad del diseño al separar su funcionamiento en distintos niveles de abstracción. En cada uno de los cuales se examinan unos pocos objetos cuya descripción detallada está en un nivel inferior. A mayor abstracción menor detalle visible, ergo menor complejidad.

El uso de herramientas de diseño automático (DA) provistos por las empresas que desarrollan sistemas automáticos de diseño electrónico *Electronic Design Automation* (EDA) incorporan los dos conceptos formulados y ayudan a diseñar con éxito, circuitos integrados de alta complejidad. Otra virtud que introduce la tecnología de DA es la reutilización de componentes (es decir: el uso de un número limitado de primitivas en el diseño de distintas funcionalidades). Esto simplifica profundamente el proceso de diseño. El avance en la escala de integración permite que miles de millones de transistores coexistan en un solo CI, lo cual pone en evidencia el nivel alcanzado en los desarrollos tecnológicos y a su vez, plantea el desafío de desarrollar una metodología que permita disipar el calor concentrado en el mismo.

La energía consumida por los circuitos integrados de alta complejidad es de cientos de Watts por centímetro cuadrado. Si bien los requerimientos funcionales del CI no se alteran, se le exige una mayor funcionalidad (más transistores) y mayor performance para cada nueva generación de los mismos.
El Prof. Dr. Ken Stevens, Director del Departamento de *Computer Engineering* de la Universidad de Utah, ha desarrollado una alternativa metodológica para lograr dicho cometido que consiste en eliminar el *clock* en el método de diseño. Para resolver la comunicación[3] dentro del CI se utiliza el tiempo de arribo de las señales a la entrada de un circuito lógico, sumada al tiempo de propagación de la señal por la lógica. Por lo tanto, el diseño debe preocuparse de asegurar la preservación de las relaciones temporales secuenciales dentro del CI. Este es el centro de la metodología del profesor, denominado "*Relative timing methodology*" y se distingue de otros trabajos en el área asincrónica (Stevens, 2003).

En conjunto con el Grupo de VLSI dirigido por el Prof. Stevens se tomó como punto de partida el diseño de un hasher *SHA-256*, popular en el mundo de los *BitCoins*, considerando la implementación de una versión sincrónica del mismo en una tecnología CMOS de 180 nm. La elección de este nodo tecnológico para comenzar la experiencia en el área de diseño y manufactura está guiada por la practicidad y la ausencia de complejidades inherentes a los procesos más avanzados y su disponibilidad sin costo alguno para diseños académicos. De allí que se proyectó trabajar con procesos de menor litografía para aumentar la velocidad de funcionamiento de los diseños y poder realizar comparaciones de rendimiento entre las técnicas sincrónicas y las asincrónicas en una misma tecnología (el profesor Stevens trabaja en un nodo 65 nm) y con arquitecturas comparables. Esta comparación de pros y contras de dos arquitecturas semejantes diferenciadas por el carácter sincrónico ó asincrónico de su implementación, es un área poco explorada en la literatura.

El equipo de Utah ha estado trabajando durante los dos últimos años entre otras cosas, en una versión asincrónica del *hasher* SHA-256 y tiene hoy en día fabricado un CI en un proceso CMOS de 65 nm de muy alta *performance* y con un consumo eléctrico muy respetable.

El diseño que se describe en este trabajo en un proceso de 180 nm es distinto al diseño del equipo antes mencionado, más allá de la diferencia de *timing*. Cabe destacar que el diseño bajo análisis no copia a este trabajo, puesto que es una síntesis propia de la misma arquitectura, pero mantiene una funcionalidad equivalente a las del equipo de Utah. El tra·bajo ha

[3] Función realizada por el *clock*

sido utilizado en sus diferentes instancias del desarrollo para familiarizarse con los innumerables problemas, nuevos para el equipo, que pueden presentarse durante el diseño de un CI digital. A futuro se proyecta avanzar de forma incremental: primero superar escollos básicos de diseño; luego, optimizar la implementación para avanzar sobre el área de metodología de *testing* y, posteriormente, implementar una arquitectura equivalente a las del equipo de Utah. Así se podrá realizar una comparación cuantitativa de rendimiento entre la versión asincrónica desarrollada por ellos y la desarrollada con la versión sincrónica, pero con idénticos procesos de fabricación. El objetivo para 2016 consiste en la implementación física de un CI sincrónico (del *hasher SHA-256*) que opere a la máxima frecuencia en 65 nm. Esto permitirá realizar una comparación válida entre las mencionadas metodologías sincrónicas y asincrónicas.

En el apartado siguiente se discuten las distintas técnicas utilizadas para el diseño del CI sincrónico, desde el nivel de comportamiento, pasando por la síntesis en silicio y la integración de este circuito adoptado como dispositivo de ensayo. Se inicia con una breve descripción del algoritmo *hasher* sintetizado y luego la proyección del circuito desde un diagrama en bloques. Con posterioridad, se describe funcionalmente en Verilog, para pasar a su síntesis y verificación las simulaciones de rendimiento y funcionamiento. Finalmente, se realiza la implementación en silicio y se efectúan técnicas de *testing*. Se cierra con las conclusiones y proyecciones a futuro sobre la continuidad de este trabajo.

Diseño de CI: Etapas analógica y digital – Flujo de diseño

El proceso de diseño de circuitos integrados digitales está organizado dentro de un flujo de diseño (*design-flow*). El mismo implica una secuencia de pasos genéricos que se reproducirán siempre, independientemente del circuito. Los mismos pueden resumirse en:

1. Diseño conceptual: definición de especificaciones, y establecimiento de los lineamientos generales del sistema.

2. Descripción de comportamiento: en circuitos digitales, el sistema se plantea mediante lenguajes de descripción de *Hardware (Verilog, VHDL*). Obtenido un primer diseño (código verilog) se somete a una iteración de simulaciones hasta obtener un sistema que cumpla con las especificaciones estipuladas.

3. Síntesis y *layout*: el código diseñado es procesado por el *software* especializado (*Synopsys IC Compiler®, Synopsys Design Compiler ®, Synopsys VCS®*), que determinará las celdas básicas estándar con las que se lo implementará. Como paso siguiente, conociendo las celdas a utilizar y cómo conectarlas, el *software* realiza el diseño físico en silicio del sistema (*layout*).

4. Design Rule Check (DRC), Layout Vs. Schematic (LVS) y Parasitic Extraction (PEX): corresponden a la etapa de verificación, en la cual el *layout* obtenido es analizado en busca de violaciones de las reglas de diseño (DRC). A continuación se compara con el diseño esquemático asociado a cada celda estándar (LVS) y finalmente, se realiza una simulación considerando los componentes parásitos no deseados que aparecen en el diseño físico (PEX). Para esto se utilizan herramientas de *Mentor Graphics, Calibre Division (Mentor Graphics,* 2015). En el presente caso, para la extracción de parásitos se utilizan los útiles de *Synopsys.*

5. Iteraciones: en el caso de que alguno de los pasos 2 a 4 no cumpla con lo estipulado, se itera desde el paso 2 hasta conseguir las especificaciones propuestas.

El diseño que se propone en este trabajo es un bloque predominantemente digital. El mismo representa un circuito capaz de encriptar mensajes de longitud arbitraria empleando un código de encriptación previamente definido (SHA-256). Esta idea es utilizada en el estampado digital de documentos para otorgarles un alto nivel de seguridad (Massias, 1999) y da sustento a la moneda virtual que ha tomado importancia en los últimos tiempos: el *BitCoin.* Esta divisa auto-regulada se promociona como un mecanismo de transferencia monetaria segura entre personas sin la necesidad de la intervención de una institución financiera (Nakamoto, 2013). Dicha seguridad radica en que la única forma de quebrar el código es por fuerza bruta (es decir, probando todas las combinaciones posibles, hasta encontrar una que al ser encriptada, produzca un *hash* que reúna ciertos requisitos que varían con el tiempo). Por ende, los algoritmos de creación y validación

utilizados en la cadena de *BitCoin* deberán ser capaces de procesar un gran número de combinaciones por unidad de tiempo.

A continuación, se describen los bloques más significativos utilizados en el sistema: el núcleo de procesamiento (el cual se encarga de encriptar los mensajes), el oscilador que proporciona la frecuencia de reloj al sistema y la *Universal Asynchronous Receiver Transmitter* (UART) para comunicar el *chip* con el mundo exterior.

SHA-256: un *Hasher* Criptográfico de *BitCoins*

Un *hash*, o función *hash*, consiste en el mapeo de mensajes de longitudes arbitrarias a valores de *n bits.* Para que un *hash* sea considerado criptográfico, debe cumplir con la condición de que el mapeo sea unidireccional, de esta forma se requerirán 2^n operaciones para encontrar un mensaje que pueda ser encriptado al *hash* considerado. Por otro lado, se dice que un *hash* es resistente a colisiones si la obtención de un mismo resultado a partir de dos mensajes diferentes es computacionalmente inviable. Estas propiedades inherentes al algoritmo SHA-256 explican su amplia utilización en el firmado digital y la protección de contraseñas.

SHA-256 es una función criptográfica de 256 bits de longitud sin clave, también llamada *Manipulation Detection Code* (MDC). Para generar el *hash* de un mensaje de longitud arbitraria, éste debe ser llevado inicialmente a una longitud múltiplo de 512 *bits*, proceso denominado *padding* y luego llevado a bloques de mensaje $M^{(1)}$, $M^{(2)}$, $M^{(N)}$, pudiendo representar algebraicamente al *hash* $H^{(i)}$ del mensaje $M^{(i)}$ como en la expresión (1), donde *C* es la *función compresión* que responde a la representación en bloques de la Figura 1 y el signo ⊕ a la suma módulo 2^{32}. Las funciones lógicas realizadas en el diagrama en bloques son presentadas en las expresiones (1-7), donde *rot* representa la rotación de *bits* hacia la derecha y *sh* el desplazamiento lógico de *bits* en la misma dirección.

$$H^{(i)} = H^{(i-1)} + C_M{}^{(i)}(H^{(i-1)}) \quad (1)$$

$$Ch(x,y,z) = (X.Y) \oplus (\overline{X}.Z) \quad (2)$$

$$Maj(X,Y,Z) = (X.Y) \oplus (X.Z) \oplus (Y.Z) \quad (3)$$

$$\sum_0(X) = rotR(X,2) \oplus rotR(X,13) \oplus rotR(X,22) \quad (4)$$

$$\sum_1(X) = rotR(X,6) \oplus rotR(X,11) \oplus rotR(X,25) \quad (5)$$

$$\sigma_0(X) = rotR(X,7) \oplus rotR(X,18) \oplus shR(X,3) \quad (6)$$

$$\sigma_1(X) = rotR(X,17) \oplus rotR(X,19) \oplus shR(X,10) \quad (7)$$

La función compresión es solo una parte del algoritmo del hasher, cuyo diagrama en bloques final puede apreciarse en la Figura 2 y está compuesto por la máquina de estados de la Figura 3 que gobierna todo el circuito y que se encarga de: contar 64 iteraciones, proveer señalización e inicializar los registros mediante una tabla de W_j valores constantes iniciales, complementada por un contador de 64 iteraciones.

Cabe destacar que cada registro es de 32 *bits*, tanto para la compresión como para la sección del mensaje y los datos se van trasladando a lo largo de las cadenas de registro de manera similar a una cola circular pero con puntos o taps en los cuales la información es modificada mediante las funciones lógicas (1-7). Una vez completado el ciclo, el resultado se obtiene sumando una constante al resultado de los registros *a-h*.

Diseño de oscilador en anillo digitalmente sintonizable

La necesidad de efectuar pruebas sobre el circuito bajo distintas condiciones de funcionamiento sugiere la posibilidad de establecer distintas frecuencias de *clock.* Una topología de oscilador muy usual en circuitos integrados es la del Oscilador en Anillo, que consiste en una cadena impar de inversores con su entrada conectada a su salida, como se muestra en la Figura 4. De esta forma, cada inversor introduce una demora en la propagación de la señal de tensión a la entrada del primer inversor, generando transiciones de alto a bajo y viceversa cuyo período de ocurrencia es directamente proporcional al número de inversores que componen la cadena. Esta topología se destaca por su sencillez pero tiene el inconveniente de que el período de la oscilación depende fuertemen-

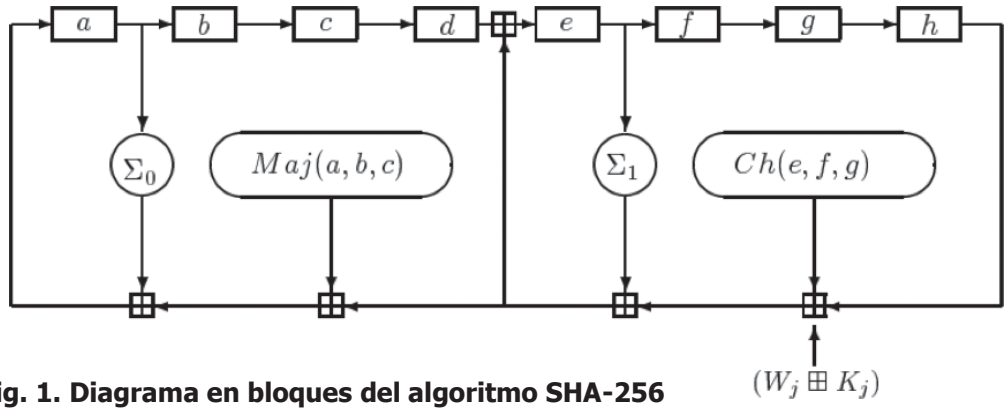

Fig. 1. Diagrama en bloques del algoritmo SHA-256

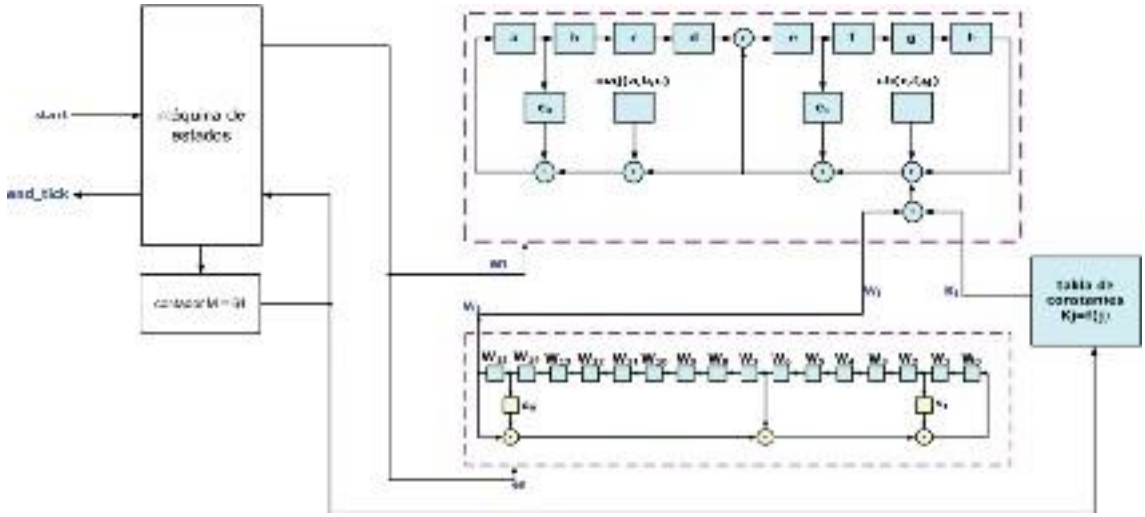

Fig. 2. Diagrama en bloques del SHA-256 completo

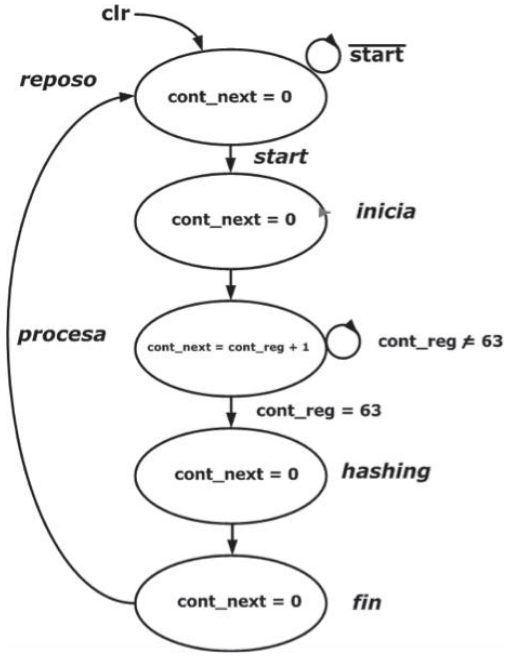

Fig. 3. Máquina de estados encargada de gobernar el *hasher*

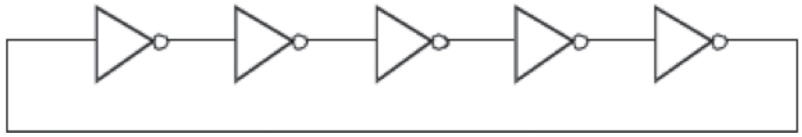

Fig. 4. Cadena de inversores impar para obtener un oscilador en anillo

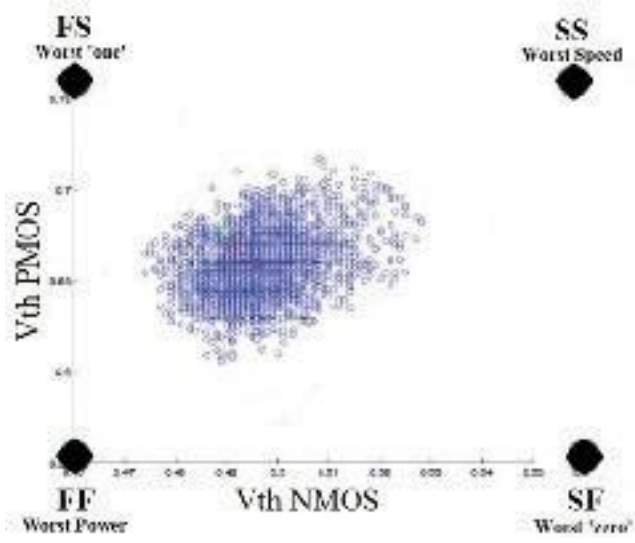

Fig. 5. Representación gráfica de los *corners* de la tecnología y diagrama de dispersión de los valores de tensiones de umbral probables

te de desviaciones de los parámetros de sus valores nominales. Estas desviaciones suelen darse durante el proceso de fabricación del circuito. Por tal razón y para asegurar el trabajo en un rango de frecuencias acorde a la frecuencia máxima esperada para el correcto funcionamiento del circuito, se realizan simulaciones de la estructura del oscilador sobre los *corners* o esquinas estadísticas de la tecnología, que consisten en los valores extremos pero probables para las características de los transistores fabricados en una dada tecnología. Usualmente, estos puntos son caracterizados por la tensión de umbral de los transistores N y P de la tecnología CMOS, conformando un área de desempeño probable como la que se observa en la Figura 5.

El circuito diseñado para cumplir la función de generador de *clock* puede verse en el esquema a nivel de compuertas de la Figura 6, donde se agrega lógica de control que permite cambiar el punto de la cadena de inversores a partir del cual se toma la señal de *clock*. De allí resulta en una frecuencia mayor cuanto menos inversores queden contenidos en la cadena. Esto se logra con un multiplexor de 4 a 1, controlado por las señales s1 y s2 y se le agrega un control de *Integrated Clock Gating* (ICG) y un divisor de frecuencia por 16 para un modo de *debug* de baja velocidad, de aproximadamente 10 MHz. El objetivo del ICG es automatizar el instante en el que se habilita la señal de *clock* para el circuito integrado. Debe tenerse en cuenta que durante el encendido del circuito al oscilador puede tomarle algunos ciclos para alcanzar un estado estacionario de funcionamiento. Las simulaciones de la frecuencia de oscilación pueden verse en la Figura 7 sobre los *corners* de la tecnología para una frecuencia nominal de trabajo de 200 MHz. Allí se aprecia una variación esperable de esta frecuencia de aproximadamente +/- 25% respecto del valor nominal, debido únicamente a las variaciones máximas esperadas del proceso de fabricación del integrado.

Descripción del *hardware*, simulaciones a nivel comportamiento y síntesis en silicio

El circuito integrado comprende dos *hashers* idénticos denominados A y B. En la Figura 8 se

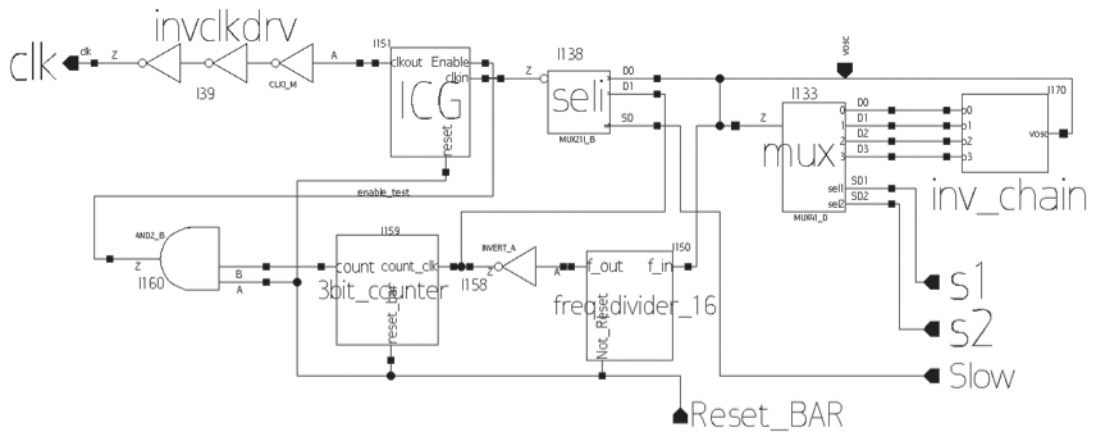

Fig. 6. Diagrama a nivel de compuertas del oscilador configurable

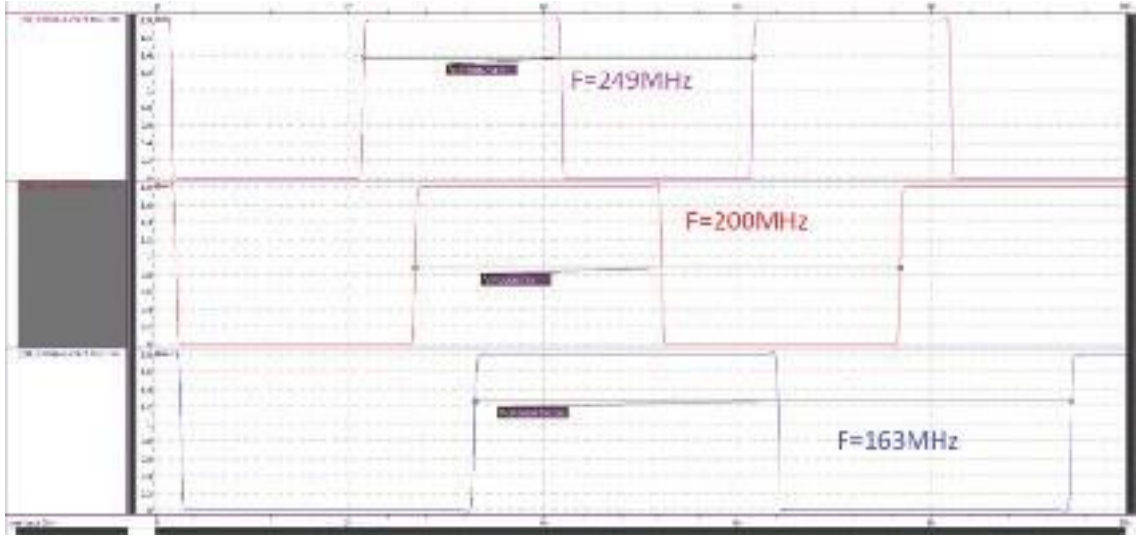

Fig. 7. Frecuencia del oscilador en anillo para condición nominal de 200 MHz, sobre los *corners* de la tecnología (FF, SS y típico)

observa el diagrama en bloques de cada uno de los hashers. El mismo comprende:

. la interfaz UART, encargada de la comunicación con el exterior,
. el *Linear Feedback Shift Register* (LFSR), encargado de la generación de mensajes pseudoaleatorios,
. la unidad de control,
. el motor de encriptación que implementa el algoritmo SHA-256,
. el divisor de la frecuencia de *clock* que permite facilitar la medición externa del *clock*,
. el oscilador propiamente dicho y
. el sincronizador de reset externo.

Todos los bloques a excepción del oscilador fueron íntegramente diseñados y verificados en el lenguaje descriptor de *hardware* Verilog (IEEE, 2001).

Cada uno de los *hashers* tiene tres modos independientes de funcionamiento:

Modo *CALC_HASH:* Luego del encendido del dispositivo, el *host* debe enviar un comando *CALC_HASH (0xDC)* a través de la interfaz de comunicación para que el dispositivo entre en el modo *CALC_HASH*. Seguidamente, el *host* debe enviar los 512 *bits* del mensaje a procesar. Dado que la interfaz serie acepta datos de 8 *bits* de tamaño por cada transacción, el host debe dividir el mensaje en 64 *bytes* ordenados comenzando por el *byte* más significativo (*Most Sgnificant Byte first - MSB first*). Cuando el dispositivo adquiere los 64 *bytes* del mensaje, automáticamente realiza el algoritmo *SHA-256* obteniendo los 256 *bits* de *hash.* Una

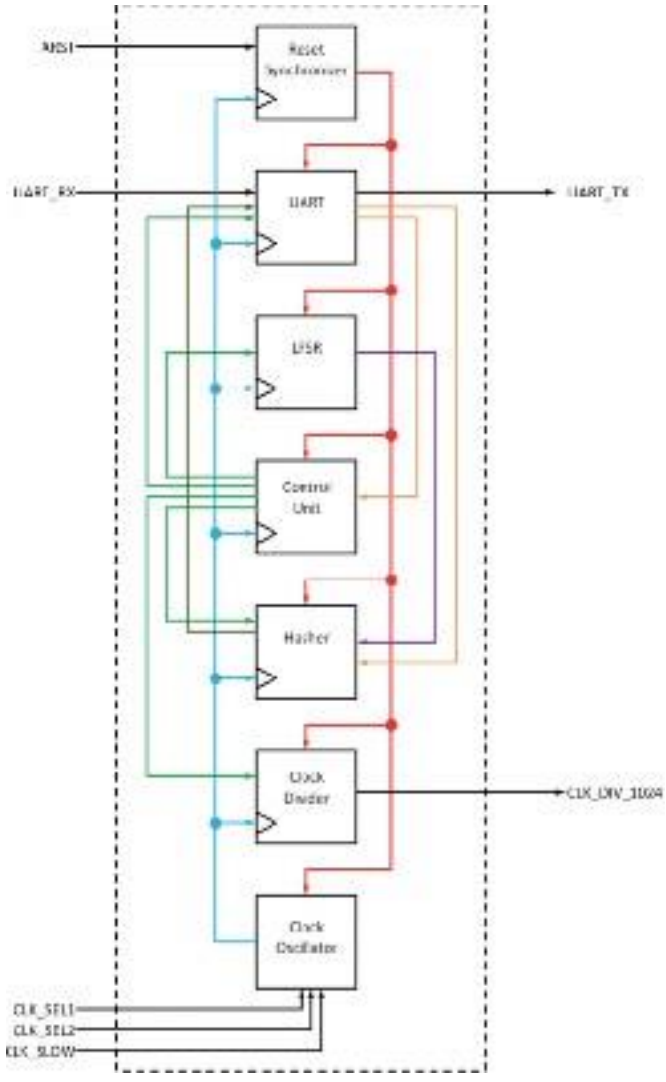

Fig. 8. Diagrama en bloque de cada uno de los *hashers*

vez realizado este cálculo, el dispositivo envía a través de su interfaz serie los 32 *bytes* de *hash* también en formato *MSB first.*

Modo DEBUG: Es similar al modo *CALC_HASH*, pero pensado para la depuración del algoritmo. El *host* debe enviar un comando *ENTER_DEBUG_MODE* (0xDE), seguido de un comando *CALC_HASH* y los 512 bits del mensaje. En cada paso del algoritmo *SHA-256*, el dispositivo transmite por su interfaz serie el estado de los registros internos. Cuando el algoritmo finaliza, el valor del *hash* correspondiente al mensaje es transmitido. Para salir del modo *DEBUG*, el host debe enviar el comando *EXIT_DEBUG_MODE (0xDF)*.
El modo *DEBUG* se logra concatenando los registros internos del *hasher* en una suerte de cadena de registros. Esto es posible dado que cada registro del *hasher* es de 32 *bits* de forma tal que se subdivide cada registro en 4 *bytes* y se lo conecta en cadena con el *byte* siguiente, tal como se observa en la Figura 9.

Para no perder el estado del algoritmo mientras se lee cada registro interno, la cadena de registros es circular, de forma tal de que al finalizar la transmisión de todos los *bytes*, la cadena de registros se encuentra en el estado previo al envío de la información. En ese momento se avanza un paso el algoritmo y se repite la operación. Este proceso se realiza hasta que el algoritmo SHA-256 es completado para el mensaje recibido como dato. De esta forma el *host* puede conocer el estado de todos los registros del *hasher* en cada paso del algoritmo, haciendo posible verificar la correcta funcionalidad del mismo. El orden de la

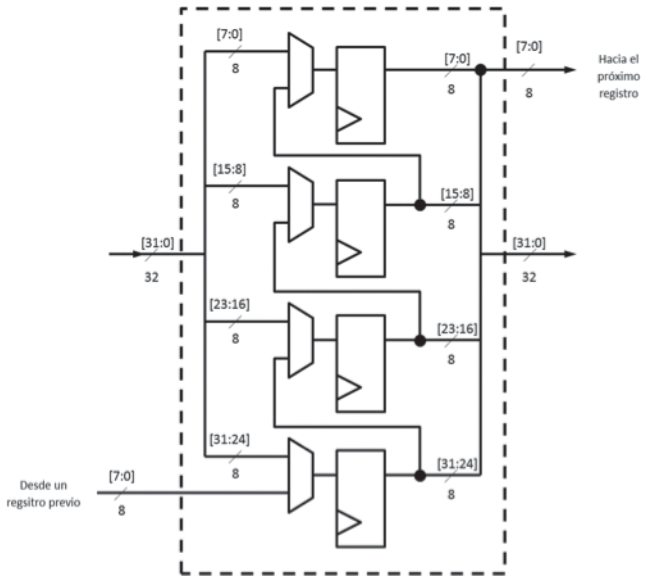

Fig. 9. Conexión interna de cada registro de 32 bits posibilitando la conexión en cadena

Orden	Registro
0 ⋮ 3	W0 [7: 0] ⋮ W0 [31:24]
4 ⋮ 7	W1 [7: 0] ⋮ W1 [31:24]
⋮	⋮
60 ⋮ 63	W15[7: 0] ⋮ W15[31:24]
64 ⋮ 67	A [7: 0] ⋮ A [31:24]
68 ⋮ 71	B [7: 0] ⋮ B [31:24]
⋮	⋮
92 ⋮ 95	H [7: 0] ⋮ H [31:24]

Tabla 1. Orden de la cadena de *debug*

cadena circular está descripto en la Tabla 1.

Modo *FREE_RUNNING*: Este modo está destinado para evaluación de consumo de potencia. En él, el motor de *hashing* toma los 512 bits de mensaje directamente desde el LFSR. Cada vez que el motor de *hash* termina de procesar un mensaje completo, acepta un nuevo dato pseudo-aleatorio generado por el LFSR. Esto hace que el motor de *hashing* opere a su máxima velocidad, pudiendo realizarse una medición de la potencia de consumo en estas condiciones de funcionamiento. Para iniciar el modo *Free running*, el *host* debe enviar a través de la interfaz serie el comando *START_FREE_RUNNING* (0xDA), luego del cual el circuito no envía ninguna información que indique que se encuentra en este modo de operación, siendo el aumento en la corriente de consumo el único testigo de este estado.

Comando	Codificación	Descripción
START_CLK_DIV	0xD8	Inicializa el *pin clk_div*.
STOP_CLK_DIV	0xD9	Detiene el *pin clk_div*.
START_FREE_RUNNING	0xDA	Inicia el modo *FREE_RUNNING*.
STOP_FREE_RUNNING	0xDB	Detiene el modo *FREE_RUNNING*.
CALC_HASH	0xDC	Habilita el cálculo de un *hash* simple.
RST_HASHER	0xDD	Realiza el *soft-reset* del motor de *hashing*.
ENTER_DBG_MODE	0xDE	Inicializa el modo *DEBUG*.
EXIT_DBG_MODE	0xDF	Sale del modo *DEBUG*.

Tabla 2. Listado de comandos y descripciones

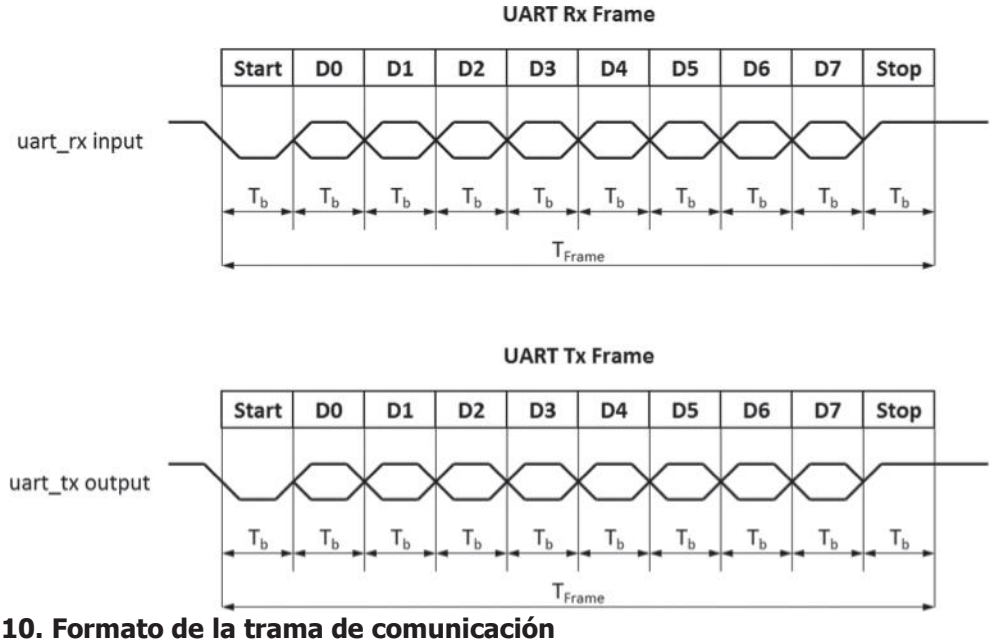

Fig. 10. Formato de la trama de comunicación

Para salir de este modo de operación, el *host* debe enviar el comando *STOP_FREE_RUNNING* (0xDB).

Medición de la frecuencia de *clock*: En el dispositivo se incorporó un *pin* denominado *clk_div*. Cuando el *host* envía el comando *START_CLK_DIV* (0xD8) este *pin* corresponde a la señal de *clock* dividida por 1024, facilitando la medición de la frecuencia de *clock*. Por defecto, luego del encendido del dispositivo, este *pin* se encuentra activo de forma tal de no requerir el envío de ningún comando por parte del *host* para poder medir la frecuencia del *clock*.

Interfaz serie

La interfaz serie corresponde al protocolo UART y se configura su trama con un 1 *bit* de inicio, 8 de datos y 1 de parada, sin *bits* de paridad, como puede observarse en la Figura 10 para Rx y Tx. El tiempo de *bit* T_b no corresponde a ninguna velocidad de transmisión (*baud rate*) estándar y debe ser calculado en función de la frecuencia de reloj del dispositivo de acuerdo a la ecuación (8).

$$T_b = \frac{10^8}{19200 \times f_{clk}[Hz]} \qquad (8)$$

En la Tabla 2 se observa el listado completo de comandos que el dispositivo puede aceptar desde el *host*.

Proceso de síntesis lógica, place and route y resultados obtenidos

La síntesis lógica fue realizada con la herramienta *Synopsys Design Compiler®* (*Synopsys Inc.*, 2014) sobre el proceso de fabricación IBM 180 RF de 180 nm provisto por *MOSIS* (*The MOSIS Service*, 2014). El proceso de *Place and Route*, se divide en dos etapas: la pri-

Resultados	
Máxima frecuencia de operación	285 MHz
Cantidad de Flip-Flop D	1694
Área	0.5mm^2

Tabla 3. Resultados de la etapa de síntesis para cada *hasher*

mera consiste en el emplazamiento de las distintas celdas que conforman el CI, generando así un denominado *floor plan*. La información del mismo es posteriormente utilizada en conjunto con información intrínseca del proceso (niveles de metal, reglas de diseño, etc.) para definir los caminos de interconexión entre las celdas. La herramienta utilizada para este proceso fue *Synopsys IC Compiler® (Synopsys Inc.*, 2014).

Los resultados obtenidos por la etapa de síntesis se detallan en la Tabla 3 para cada uno de los *hashers*.

iPDK en *Open Access*: herramientas del flujo de diseño

Para el desarrollo de este proyecto, fue necesaria la configuración en los servidores de aplicaciones del Laboratorio de Microelectrónica de UTN–FRBA del mencionado *software*, provisto por *Synopsys®* y *Mentor Graphics®,* firmas con las cuales se consiguieron acuerdos académicos para la utilización de estas herramientas. Como paso posterior a la instalación, las herramientas de síntesis, *layout* y extracción de parásitos, provistas por *Synopsys®*, debieron ser configuradas para interactuar con las de verificación (DRC, LVS) provistas por *Mentor Graphics®*. De igual manera, ambos conjuntos de *software* debieron ser adaptados para la utilización de los *kits* de diseño físico interoperables o *Interoperable Physical Design Kit*, (iPDK) en formato *open access*, provistos por IBM, a través de MOSIS. Dicho *kit* corresponde al proceso de fabricación empleado para la síntesis del código verilog (IBM7RF, de 180 nm de longitud de canal). El mismo contiene los parámetros físicos de los componentes para las simulaciones de comportamiento del sistema, así como los diseños de *layout* asociados a los mismos y celdas paramétricas (*pCells*, que generan el *layout* asociado a determinados componentes en forma automática, a partir de las dimensiones que asigna el usuario). Por otro lado, es también el iPDK el que contiene la información de reglas de diseño y extracción, las cuales son utilizadas durante los procesos de DRC, LVS y PEX.

El diagrama de la Figura 11 esquematiza la estructura adoptada para organización de las herramientas: el grueso del *software* está instalado sobre un único servidor (procesador Intel Xeon, 16 GB de memoria RAM, 2 TB de almacenamiento, y utilizando como sistema operativo CentOS 6.5) que realiza todo el procesamiento de datos para el diseño y síntesis de circuitos. Para ello, cada investigador registrado en el laboratorio accede al mismo en forma remota vía *Secure Shell* (SSH) desde cualquier ordenador dentro de UTN-FRBA. Habiendo adoptado esta filosofía de trabajo multiusuario, se gestiona el nivel de acceso a cada usuario del sistema mediante los permisos de UNIX® y se automatiza el *back-up* del proyecto en un repositorio Mercurial en otro equipo.

De igual manera, el PDK provisto por IBM a través de MOSIS, fue organizado en función su contenido. Dado que el mismo es una extensa librería de modelos de componentes, estos fueron clasificados según el *software* para el cual están optimizados (*Synopsys®, Mentor Graphics®, Cadence®*), y se agruparon por otro lado, aquellos modelos que son intrínsecos del proceso. Esta metodología de ordenamiento es de suma practicidad para administrar la instalación del PDK, y nos permitirá eventualmente, organizar otro proceso CMOS en forma similar, **reduciendo en gran medida el tiempo demandado para las tareas de puesta a punto de los scripts de ejecución y administración.**

Las rutinas de administración, ejecución y mantenimiento para la vinculación de todas las herramientas utilizadas fueron escritas en lenguajes *bash* y TCL y posibilitan a su vez, la administración de todos los proyectos a desarrollar en el laboratorio. Las mismas permiten al usuario una interacción mucho más amigable con las aplicaciones de diseño, ya que automatizan la vinculación de cada proyecto en

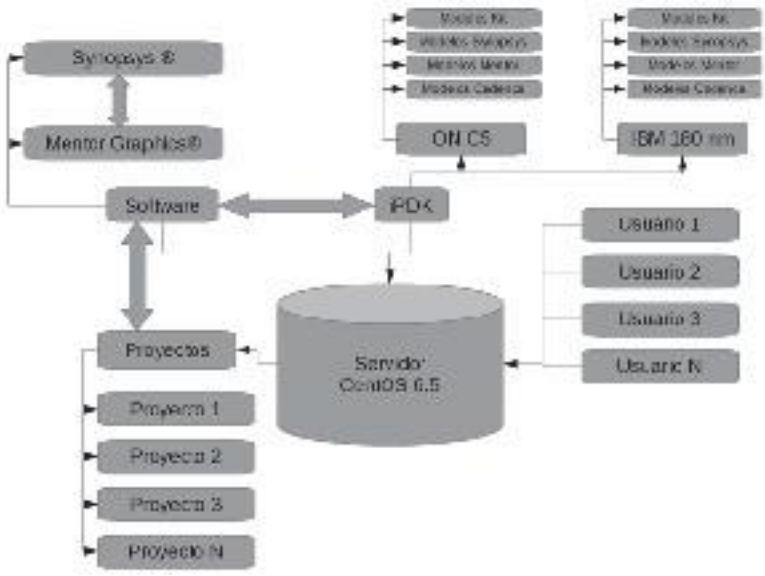

Fig. 11. Estructura Servidor de aplicaciones + iPDK

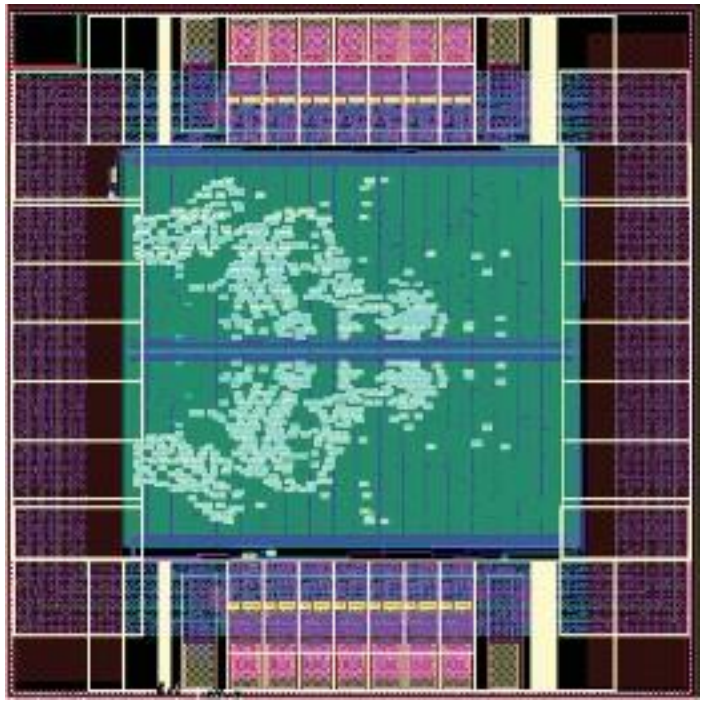

Fig. 12. *Layout* del circuito integrado. Pueden apreciarse los 18 pads de conexión

Fig. 13. Esquema de conexión para la comunicación con el *hasher*

particular con el *software* y las librerías (PDK).

Es importante mencionar, que en paralelo a la puesta a punto del iPDK para 180 nm, se configuró un PDK para 500 nm. Las metodologías desarrolladas planean ser usadas para la configuración de un proceso de 130 nm de IBM, de forma que los tres procesos coexistirían en paralelo en el laboratorio. La puesta a punto de esta estructura (servidor de aplicaciones, instalación del *software*, configuración del PDK, configuración del *software* y su vinculación con el PDK y gestión del acceso remoto) demandó una gran inversión de tiempo y esfuerzos, pero permitirá reducir drásticamente los tiempos empleados para el diseño de circuitos integrados dentro del Laboratorio de FRBA-UTN, y representará una infraestructura disponible en muy pocos establecimientos universitarios dentro del país.

Fabricación y pruebas de funcionamientos

Layout, fabricación y encapsulado

Una vez completado el trabajo de síntesis y verificación, se provee el diseño de las máscaras de litografía a la *foundry* encargada de fabricar el circuito y encapsularlo. Para este diseño se optó por incluir en un mismo *die* (unidad de silicio sobre la cual funciona el diseño) de 1,5 mm x 1,5 mm, dos *hashers* completos con sus respectivos osciladores configurables y *pads* de entrada/salida para la prueba del circuito. En total, cada uno cuenta con 7 *pads* para las líneas de Tx, Rx de la UART embebida, una salida con el valor de *clock* dividido en 1000 veces para su medición y estimación de la velocidad real de funcionamiento y las entradas de control del oscilador configurable, dos para el control fino y una para el valor de f=10 MHz de *debugging*. A cada circuito se le agregan *pads* de alimentación (GND y VDD=1,8 V). En total, el *chip* posee 18 *pins* y se decidió colocarlo en una cápsula del tipo *OCP SOIC* de 20 *pins* - *OCP_SOIC_300_20A* - (*The MOSIS Service*, 2015) de montaje superficial.

La Figura 12 muestra una imagen del *layout* del circuito final, donde pueden verse hacia el centro del *die* los dos *hashers*. Sin embargo, no se distinguen los transistores ni las celdas lógicas debido a la muy alta densidad de transistores por unidad de área, sumado a las líneas de interconexión en 6 niveles metálicos distintos.

La implementación en silicio del circuito integrado diseñado se realiza gracias al consorcio estadounidense MOSIS (*The MOSIS Service,* 2015), con quien la FRBA-UTN mantiene un acuerdo académico para la fabricación sin costo de circuitos integrados con fines educativos y de investigación. Esto es posible utilizando el tiempo muerto de las *foundries* y por medio de los denominados *Multi Project Waffer* (MPW). El tiempo estimado que demora este proceso ronda los 2 a 4 meses a partir del envío de las máscaras de fabricación a MOSIS. Este tiempo se destina normalmente a la optimización de las estrategias de *test* y a la documentación de la etapa de diseño del proyecto. Como ya se ha comentado, este consorcio es el que el que brinda acceso a los procesos de fabricación, como por ejemplo IBM7RF, proceso de longitud de canal de 180 nm con el cual se sintetizó este circuito.

Testing: Comunicación con el CI y pruebas de funcionamiento

La primera versión del *chip* posee como interfaz de comunicación con el mundo exterior una UART cuya frecuencia de *clock* es derivada de la principal del sistema (oscilador en anillo). Por ello, el *baud rate* de la UART podría no ser estándar (9600BPS; 19200BPS: 115200BPS; etc.) dado que éste surge de una división entera del *clock* del sistema. Esta simplificación en el diseño del *chip* trae aparejadas dos consecuencias: la primera es que dificulta la interfaz de *testing* (por el *baud rate* no estándar), teniendo que colocar una *Field Programmable Gate Array* (FPGA) entre el puerto de comunicación con la PC y el *chip* para compensar esta diferencia; la segunda es la limitación de velocidad que es impuesta por una comunicación asincrónica.

A nivel de *hardware* el *chip* conecta cada uno de sus *pins* a una *Complex Programmable Logic Device (CPLD)* que a su vez se conecta a un puente o *bridge USB-serie* y éste a una PC que trasmitirá los patrones de testeo. La CPLD se colocó para adaptar niveles de tensión y poder solucionar cualquier incompatibilidad entre la UART del *chip* y la UART del *bridge USB-serie*. Además se utiliza para controlar los *pins* de I/O que controlan el *clock* y el modo de funcionamiento.

Los modos de funcionamiento *CALC_HASH y FREE_RUNNING* previamente descriptos, son utilizados en esta etapa de *test*. En el primero, se espera a que se le transmita al *chip*, utilizando la UART, un mensaje patrón a procesar, para luego responder con el resultado del algoritmo para ese mensaje particular. Con este modo se espera verificar el correcto funcionamiento del *chip*. Luego, con el modo *FREE_RUNNING,* se evaluará su consumo sin recibir datos a procesar a través de la UART sino a través de un LFSR. Habilitando los distintos *clocks* del oscilador en anillo es posible medir el consumo para distintas frecuencias de operación.

Conclusiones y perspectivas

El presente trabajo ha sido el puntapié inicial para la colaboración e interacción con el equipo del Prof. Stevens y en la capacitación de nuestro equipo de trabajo en el área del diseño de CI digitales. Desde el punto de vista del diseño, una gran cantidad de tiempo y esfuerzo (más de 2000 horas hombre hasta la fecha) fue invertida en la configuración de un iPDK capaz de otorgar la flexibilidad y productividad necesaria para poder tener un flujo de diseño sólido y que permita plantear objetivos puntuales en tiempos perfectamente estimables. En la actualidad, el Laboratorio de Microelectrónica de la FRBA-UTN, posee la capacidad de diseñar y verificar circuitos integrados analógico-digitales *(mixed-signal)* en tecnologías CMOS-RF de 180 nm provistas por IBM a través de MOSIS, *lo cual es un gran logro entre las entidades educativas del país*. En el futuro cercano, se proyecta la inclusión de un nodo más reciente de fabricación en 130 nm.

El circuito sintetizado proyecta un funcionamiento a una velocidad de *clock* de 200 MHz, pudiendo optimizarse el diseño para una mayor velocidad, objetivo que buscará alcanzarse en la próxima iteración. Cabe destacar que el mismo se toma como caso de estudio para realizar una comparación estimativa entre arquitecturas. El *testing* demuestra ser tan complejo como lo es el diseño del circuito, requiriendo un sistema que pueda correr programas de evaluación a la velocidad de funcionamiento del circuito integrado diseñado.

En el corto plazo, se planea la fabricación de una nueva versión del circuito propuesto para optimizar las metodologías de *testing* y medición del rendimiento del circuito sincrónico, de forma de comparar resultados con estructuras equivalentes diseñadas mediante estrategias asincrónicas. Entretanto, se prevé continuar con la capacitación en el diseño de circuitos integrados asincrónicos para realizar contribuciones a este área del conocimiento que promete, a futuro, importantes resultados a ser tenidos en cuenta no sólo en el ámbito académico sino en la industria de los circuitos de alto rendimiento.

Las implicancias de esta última idea pueden ser fácilmente contextualizadas cuando se analizan los problemas que enfrenta el diseño sincrónico en la actualidad: los límites cada vez más próximos en el escalamiento de las tecnologías CMOS acarrean incrementos inadmisibles del consumo y la saturación de la capacidad de procesamiento (Iwai, 2015). Por ello, diversas alternativas emergieron en el ámbito académico en la búsqueda de la optimización de los circuitos digitales. Se trató de maximizar su velocidad de procesamiento a un consumo considerablemente menor. Una de las más destacadas es el diseño de circuitos asincrónicos. Esto significa que su funcionamiento no está atado al ritmo de una señalización periódica externa para la realización de operaciones. Es decir, que el circuito con sus estructuras internas, controla el flujo de información entre etapas lógicas mediante señalizaciones internas totalmente asincrónicas. Esto es una suerte de protocolo "*on-demand*" donde el funcionamiento de una estructura se dispara en el momento en que la estructura anterior puso a disposición sus datos. Esta técnica, ampliamente desarrollada actualmente por el profesor Stevens (Stevens, 2013, 2011, 2010) ha reportado ya resultados prometedores en cuanto a consumo y capacidad de procesamiento, tomando siempre como casos de estudio un circuito integrado de aplicación específica como circuito bajo prueba.

Agradecimientos

Los autores de este trabajo agradecen a *Mentor Graphics® y Synopsys Inc. ®* por brindar el acceso al *software* de verificación y diseño a través de sus respectivos acuerdos académicos, sin los cuales el desarrollo aquí presentado no hubiera sido posible.

Glosario

. SHA: *Secure Hash Algorithm* (Algoritmo de Hash Seguro)
. PDK: *Physical Design Kit* (Kit de Diseño Físico)
. iPDK: *Interoperable Physical Design Kit* (Kit de Diseño Físico Interoperable)
. CMOS: *Complementary Metal-Oxide-Semiconductor* (Metal-Óxido-Semiconductor Complementario)
. EDA: *Electronic Design Automation*
. CI: Circuito Integrado
. VLSI: *Very Large Scale Integration* (Alta Escala de Integración) - Corresponde al nivel de integración de dispositivos (transistores MOS) en un circuito integrado.
. DRC: *Design Rule Check* (Chequeo de Reglas de Diseño)
. LVS: *Layout versus Schematic* (Layout vs. Esquemático)
. PEX: *Parasitic Extraction* (Extracción de parásitos)
. UART: *Universal Asynchronous Receiver Transmitte*r (Transmisor-Receptor Universal asincrónico)
. FIFO: *First Input - First Output* (Primero Entra-Primero Sale)
. LFSR: *Linear Feedback Shift Register* (Registro de Desplazamiento Circular)
. FPGA: *Field Programmable Gate Array* (Matriz de Compuertas Programables)
. CPLD: *Complex Programmable Logic Device* (Dispositivo Complejo Lógico Programable)

Referencias

IWAI, H. (2013) "Future of nano CMOS technology", en Proceedings of the Symposium on Microelectronics Technology and Devices (SBMicro '13), pp. 1-10.
IEEE, (2001) "IEEE standard Verilog hardware description language" IEEE Standards (IEEE Std 1364-2001), Septiembre 2001.
STEVENS, K.; GINOSAR, R. y ROTEM, S., (2003) "Relative timing", en IEEE Transactions on very large scale integration (VLSI) systems, Vol. 11, No. 1, Febrero 2003.
STEVENS, K. S.; DAS, S. y MANETAS G., (2013) "Source Asynchronous Signaling Protocol for Asynchronous Handshake Communication Free from Wire Delay Overhead", en IEEE International Symposium on Asynchronous Circuits and Systems, pp. 107-114.
STEVENS, K. S.; GEBHARDT, D. y YOU, J., (2010) "Comparing Energy and Latency of Asynchronous and Synchronous NoCs for Embedded SoCs", en IEEE International Symposium on Network-on-Chip, pp. 115-122.
STEVENS, K. S.; GEBHARDT, D. y YOU, J., (2011) "Design of an Energy-Efficient Asynchronous No Candits Optimization Tools for Heterogeneous SoCs", en IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 30(9), pp. 1387-1399.
NAKAMOTO, S., (2013) "Bitcoin: A Peer-to-Peer Electronic Cash System", en www.bitcoin.org, Research Papers.
MASSIAS, H.; SERRET AVILA X. y QUISQUATER, J., (1999) "Design of a Secure Timestamping Service with minimal Trust Requirement", UCL Crypto Group, Louvain-la-Neuve, Bélgica.
SYNOPSYS INC., (2015) Synopsys Design Compiler, www.synopsys.com California, USA.
MENTOR GRAPHICS, (2015) www.mentor.com, Oregon, USA.
THE MOSIS SERVICE, (2015) www.mosis.com California, USA.