A Low-Cost and Highly Compact FPGA-Based Encryption/Decryption Architecture for AES Algorithm

Christian Equihua, Esteban Anides, Luis García, Eduardo Vázquez, Gabriel Sánchez, Juan-Gerardo Avalos, and Giovanny Sánchez

***Abstract*— Nowadays, the design of ultra-compact area advanced encryption standard (AES) architectures is highly demanded by the electronics industry since many of these architectures are embedded in portable devices, such as smart phones, tablets, etc., in which the area is critically limited. Until now, many approaches have been proposed to create high-processing and compact architectures. However, the area consumption is still a factor to be improved. In this paper, a highly compact encryption/decryption architecture, which is implemented in a low-cost FPGA, to efficiently simulate the AES algorithm, is proposed. Specifically, an optimized Galois Field Multiplier, which is the most demanding operation in terms of area consumption and processing speed, involved in Mix-Columns and Inverse Mix-Columns transformations, is presented. Therefore, the optimization of the proposed GF (2^8) multiplier by two has allowed to us create an ultra-compact Mix-Columns circuit since this circuit involves large number of multiplications. In addition, the design involves a routing circuit which allowed the proposed architecture to perform encryption or decryption by using common modules. The results demonstrate that the proposed digital circuit expends fewer LUTs and fewer registers when compared with the most compact encryption/decryption architectures reported to date.**

***Index Terms*— Advanced Encryption Standard algorithm, FPGA, encryptor / decryptor, GF (2^8) multiplier.**

I. Introducción

Actualmente, las crecientes tendencias en el uso de diversos dispositivos portátiles, como teléfonos inteligentes, tabletas, etc., han provocado que estos dispositivos sean diseñados con un bajo consumo de energía y área. Además, estos dispositivos contienen un gran número de aplicaciones, especialmente internet de las cosas (del inglés: Internet of Things, abreviado IoT) [1]-[2]. Evidentemente, estos dispositivos cuentan con avanzadas interfaces de transmisión y recepción de datos, lo que permite un intercambio de información entre estos dispositivos de manera eficiente. Para proteger la información en estos dispositivos, comúnmente se utiliza el algoritmo AES (del inglés: Advanced Encryption Standard) debido a sus características como son: fácil implementación y una alta seguridad [3]-[5].

Hasta la fecha, diversos trabajos han demostrado que la implementación del algoritmo AES se puede realizar en software (sistemas de cómputo de propósito general) para aplicaciones donde no se requiere de altas velocidades de procesamiento [6]. Sin embargo, en aplicaciones avanzadas, tales como: sistemas de comunicaciones, sistemas militares, banca electrónica, entre otros, se requiere del desarrollo de sistemas hardware para acelerar la simulación del algoritmo AES y así proteger la información de los usuarios [7]-[11]. Evidentemente, el uso de hardware permite desarrollar sistemas computacionales de alto rendimiento y con una alta seguridad en comparación con las implementaciones en software [7]. Recientemente, se ha incrementado significativamente el uso de dispositivos FPGAs (del inglés: Field Programmable Gate Array) para el desarrollo de sistemas criptográficos AES de altas velocidades de procesamiento, dado que estos dispositivos cuentan con avanzados bloques lógicos y configurables. Por lo tanto, se requiere menos tiempo para realizar prototipos dada su característica de reconfigurabilidad en comparación con los circuitos ASIC (del inglés: Application Specific Integrated Circuit) [12]. Actualmente, los diseños implementados en dispositivos FPGA son considerados como productos finales dado que su integración en los sistemas electrónicos comerciales facilita la reducción del tiempo de desarrollo y de esta manera se puede realizar una fácil actualización a nivel hardware. Por ejemplo, Soltani et al. [7] proponen una arquitectura hardware para simular el algoritmo AES con un alto rendimiento. En este trabajo, los autores proponen un esquema de implementación de la caja-S (del inglés: S-box) utilizando elementos de memoria y elementos combinacionales. Además, los autores diseñan muliplicadores GF (2^8) compactos para minizar el consumo de área. Otra arquitectura hardware fue propuesta por Priya et al. [8] para simular el algortitmo AES eficientemente en términos de velocidad de procesamiento. En este trabajo, los autores utilizan solo circuitos lógicos combinacionales y modifican la estructura de la caja-S para reducir su consumo de área. Sharma et al. [9] propusieron una arquitectura hardware, en la cual la caja-S se restructuro lógicamente para reducir los caminos críticos, los cuales crean circuitos lógicos inestables y estos aparecen cuando sólo se utiliza circuitos lógicos combinacionales en cascada. Farashahi et al. [10] propusieron

C. Equihua is with the Instituto Politécnico Nacional ESIME Culhuacan, Ciudad de México, México (e-mail: cequihuae1900@alumno.ipn.mx).

E. Anides is with the Instituto Politécnico Nacional ESIME Culhuacan, Ciudad de México, México (e-mail: eanidesc1800@alumno.ipn.mx).

L. García is with the Instituto Politécnico Nacional ESIME Culhuacan, Ciudad de México, México (e-mail: jorgeluis102008@hotmail.com).

E. Vázquez is with the Instituto Politécnico Nacional ESIME Culhuacan, Ciudad de México, México (e-mail: edvazquezf@ipn.mx).

G. Sánchez is with the Instituto Politécnico Nacional ESIME Culhuacan, Ciudad de México, México (e-mail: caaann@gmail.com).

J G. Avalos is with the Instituto Politécnico Nacional ESIME Culhuacan, Ciudad de México, México (e-mail: javaloso@ipn.mx). *Corresponding author.*

G. Sánchez is with the Instituto Politécnico Nacional ESIME Culhuacan, Ciudad de México, México (e-mail: gsanchezriv@ipn.mx). *Corresponding author.*

un cifrador AES de alto rendimiento, el cual contiene un número balanceado de registros para eliminar los caminos críticos. Lee et al. [11] propusieron un diseño para simular el algoritmo AES con el mínimo consumo de área. Para lograr el objetivo, los autores utilizaron multiplicaciones matriz-vector constantes. Analizando los trabajos existentes se puede observar que la mayoría de las arquitecturas hardware utilizan elementos lógicos combinacionales para garantizar una gran eficiencia en términos de velocidad de procesamiento y un bajo consumo de área. Sin embargo, el uso de elementos combinacionales en cascada genera caminos críticos, los cuales producen inestabilidad en los circuitos digitales, y por lo tanto reducen criticamente su rendimiento.

Como se mencionó anteriormente, la mayoría de los dispositivos portátiles emplean circuitos AES para cifrar la información. Por lo tanto, el factor más importante a tener en cuenta en el diseño de avanzados cifradores AES es el consumo de área. Además, en aplicaciones prácticas se requiere tanto del proceso de cifrado como el del descifrado en el mismo dispositivo [13]. Por lo que, el diseño de circuitos cifradores/descifradores AES con el mínimo consumo de area se ha convertido en un gran reto. Cabe señalar que las definiciones de los procesos de cifrado y descifrado AES no son idénticas [14]. Esta característica plantea otro desafío para el desarrollo de arquitecturas compactas de cifrado/descifrado AES, debido a que la implementación de los procesos de cifrado y descifrado por separado implicaría requisitos de área costosos dado que la implementación de cada proceso requiere de un circuito específico. Es por ello, que muy pocos trabajos han presentado arquitecturas de cifrado/descifrado AES embebidas en el mismo dispositivo [15]- [20]. Por ejemplo, Ueno et al. [15] presentaron una arquitectura para simular el algoritmo AES para el cifrado/descifrado de la información. En particular, los autores utilizan técnicas de reordenamiento y retardo con registros, lo que permite al circuito tener caminos críticos cortos. Por otra parte, Rajasekar et al. [16] propusieron una arquitectura compacta para realizar el cifrado/descifrado de la información usando el algoritmo AES. Los autores usan técnicas de diseño digital como: pipeline, reutilización de recursos, intercambio de recursos para minimizar el consumo de área. Visconti et al. [17] presentaron un cifrador/descifrador AES. Kundi et al. [18] presentaron el diseño de un cifrador/descifrador AES usando una estructura simétrica ST-Box. Además, los autores utilizan un bloque de memoria RAM (del inglés: Random Access Memory) para almacenar todas las operaciones de cifrado y descifrado. Por otra parte, Srinivas et al. [19] propusieron el diseño de un cifrador/descifrador AES utilizando principalmente LUTs (del inglés: Look-Up Tables) para conseguir un alto rendimiento y un bajo consumo de área. Analizando los trabajos anteriores, se puede observar que el diseño de arquitecturas compactas, las cuales incluyen los procesos de cifrado y descifrado en el mismo dispositivo es aún un reto a vencer. En este trabajo se presenta un cifrador/descifrador AES ultra-compacto. Específicamente, en este trabajo se optimizó el multiplicador de campo de Galois debido a que es el circuito más demandante en términos de consumo de área y velocidad de procesamiento dado que este circuito es utilizado en el proceso de las transformaciones de columnas mixtas y columnas mixtas inversas, es decir, en operaciones de multiplicación de matrices. Por lo tanto, la optimización del multiplicador GF (2^ 8) propuesto ha permitido crear un circuito con un bajo consumo de área en comparación con los cifradores/descifradores actuales.

II. Algoritmo de cifrado/descrifrado AES

En el año 1998, los científicos Joan Daemen and Vincent Rijmen publicaron el algoritmo Rijndael [20]. En octubre del año 2000 este algoritmo fue el ganador del concurso organizado por la NIST para convertirse en el algoritmo AES. El algoritmo AES es considerado como un algoritmo de cifrado simétrico, es decir, se requiere la misma clave tanto para el descifrado como para el proceso de cifrado. En general, el algoritmo AES es caracterizado por los siguientes aspectos:

- Utiliza llaves de 128, 192 o 256 bits de longitud, las cuales corresponden a tres niveles de cifrado.
- Realiza iteraciones, donde cada iteración se llama ronda. En este caso, el número de rondas depende de la longitud de la llave. Por lo tanto, se requieren de 10, 12 y 14 rondas para las llaves de 128, 192 y 256 bits, respectivamente.
- Trabaja en el campo finito $GF(2^8)$ en donde cada byte es representado por un polinomio de grado menor que ocho. Las operaciones entre coeficientes se realizan en módulo 2. La suma se define como la suma convencional entre polinomios. La multiplicación se realiza como la multiplicación convencional entre polinomios y el polinomio irreducible $m(x) = x^8 + x^4 + x^3 + x + 1$.

En este trabajo, se utiliza el algoritmo AES de 128 bits, como se muestra en la Fig. 1. A continuación, se describen las etapas de cifrado y descifrado del algoritmo AES.

-*Etapa de cifrado.* La Fig.1 muestra la etapa de cifrado. Inicialmente, el texto original es dividido en bloques de 16 bytes, los cuales son organizados en una matriz, la cual es denominada matriz de estado, de 4x4 bytes. Específicamente, los bytes (0-3, 4-7, 8-11 y 12-15) son almacenados en las columnas (0, 1, 2 y 3) de la matriz de estado, respectivamente. A continuación, se inicializa el contador *i* en cero. Este contador permite llevar a cabo la operación XOR entre los elementos de la matriz de estado y las palabras (w[0], w[1], w[2] y w[3]). Posteriormente, y durante nueve rondas, se llevan a cabo las transformaciones *SubBytes*, *ShiftRows*, *MixColumns* y *AddRoundKey*. En la ronda 10, se llevan a cabo las operaciones *SubBytes*, *ShiftRows* y *AddRoundKey*. Finalmente, las columnas (0, 1, 2 y 3) de la matriz de estado, ahora denominada matriz de estado final, se mapean a los bytes (0-3, 4-7, 8-11 y 12-15) del texto cifrado, respectivamente.

- *Etapa de descifrado*. La Fig. 1b muestra la etapa de descifrado. Inicialmente, se establece i=10 para llevar a cabo la operación XOR entre la matriz de estado final y las palabras (w[40], w[41], w[42] y w[43]). Posteriormente, durante nueve rondas se llevan a cabo las transformaciones *InvShiftRows*, *InvSubBytes*, *AddRoundKey* e *InvMixColumns*. Finalmente, la matriz de estado se mapea al texto original.

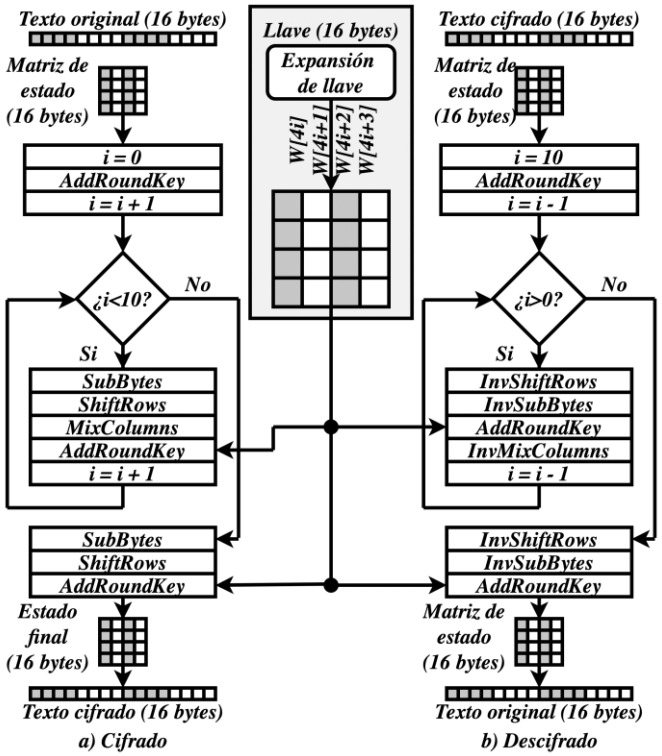

Fig. 1. a) Cifrado y b) descifrado del algoritmo AES-128.

Como se mencionó anteriormente, el algoritmo AES cifra un bloque de datos a través de las transformaciones *AddRoundKey*, *SubBytes*, *ShiftRows* y *MixColumns*, y los descifra a través de *AddRoundKey, InvSubBytes*, *InvShiftRows*, e *InvMixColumns*. A continuación, se explican cada una de estas etapas.

Cifrado:

- *AddRoundKey*. En esta transformación se realiza la operación XOR entre la matriz de estado y la expansión de llave en cada ronda. La suma de bytes a través de la operación XOR es posible debido a que el algoritmo AES trabaja en $GF(2^8)$.

- *SubBytes.* AES define una matriz de búsqueda de 16x16 bytes llamada S-box, la cual consiste en una permutación de los valores del 0 al 255. Para construir esta matriz se asignan los valores 0, 1, …, 255 a la caja-S renglón a renglón y de izquierda a derecha. Después, se mapean todos los bytes en la caja-S a su inverso multiplicativo en $GF(2^8)$. Cabe señalar que esto permite tener una transformación no lineal de los 256 valores, por lo tanto el byte {00} se mapea a sí mismo. Finalmente, se aplica la transformación, como se muestra en (1):

$$b_i' = b_i \oplus b_{(i+4)\ mod\ 8} \oplus b_{(i+5)\ mod\ 8} \oplus b_{(i+6)\ mod\ 8} \oplus b_{(i+7)\ mod\ 8} \oplus c_i \qquad (1)$$

donde b_i y b_i' $(0 \le i < 8)$ representan los ocho bits antes y después de la transformación de cada byte en caja-S y c_i es el i-ésimo bit del byte c con valor igual a {63}.

- *ShiftRows.* En esta transformación, el renglón n de la matriz de estado es desplazado n posiciones a la izquierda en forma circular.

- *MixColumns*. En esta transformación cada columna de la matriz de estado representa un polinomio de cuatro términos con coeficientes en $GF(2^8)$. Cada columna es multiplicada por el polinomio $a(x)= 3x^3 + x^2 + x + 2$ módulo (x^4+1) para obtener el resultado deseado.

Descifrado:

- *AddRoundKey*. En esta etapa se realiza la operación XOR. Esta operación puede llevarse a cabo dado el operador XOR exhibe la propiedad A ⊕ B ⊕ B = A.

- *InvSubBytes*. Para la operación inversa se aplica la transformación, como se muestra en (2):

$$b_i' = b_{(i+2)\ mod\ 8} \oplus b_{(i+5)\ mod\ 8} \oplus b_{(i+7)\ mod\ 8} \oplus d_i \qquad (2)$$

donde d={05}.

- *InvShiftRows.* En esta transformación el renglón n de la matriz de estado es desplazado n posiciones a la derecha en forma circular.

- *InvMixColumns*. Esta es la operación inversa a *MixColumns*, para implementarla se debe multiplicar cada columna de la matriz de estado por $b(x)= \{0B\}\ x^3 + \{0D\}x^2 + \{09\}x + \{0E\}$.

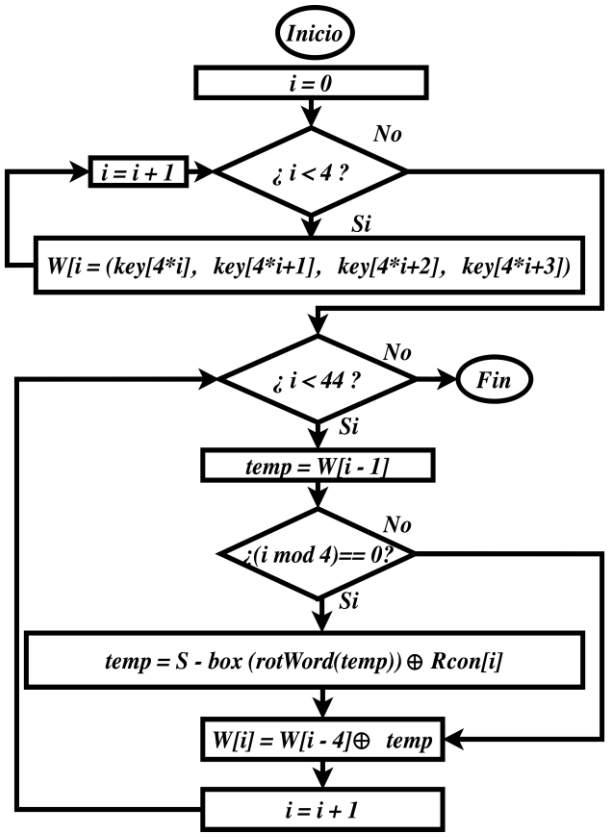

Fig. 2. Expansión de llave en AES-128.

Además de los procesos descritos anteriormente, el algoritmo AES require de una operación denominada expansión de llave

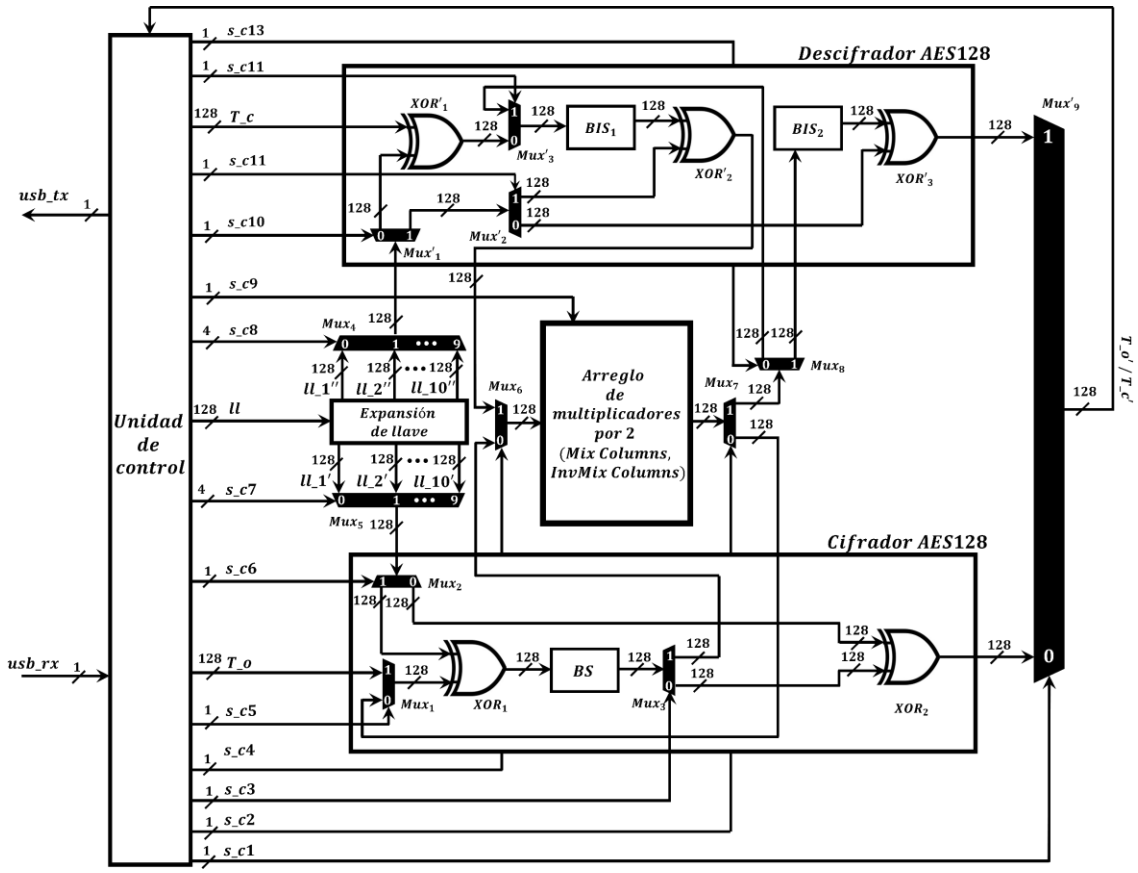

Fig. 3. Diagrama general de la arquitectura de cifrado/descifrado AES128.

(ver Fig. 1). En la Fig. 2 se muestra el diagrama de flujo para llevar a cabo el proceso de expansión de llave. Este algoritmo recibe como entrada los 16 bytes de la llave de cifrado (desde *key*[0] hasta *key*[15]) y entrega como salida las 44 palabras (desde *w*[0] hasta *w*[43]) requeridas en el proceso de cifrado y descifrado en AES-128. La función *rotWord*(*temp*) realiza un corrimiento circular a la izquierda de un byte en la palabra *temp*, cuyo resultado es ingresado a la matriz de la caja-S para hacer la sustitución del byte respectivo. La constante de ronda *Rcon*[*i*] es igual a (*RC*[*i*],0,0,0), en donde los tres últimos bytes son cero y el primer byte está dado por *RC*[1]=1, *RC*[*i*]=*2·RC*[*i*-1], cuya multiplicación está definida en GF(2^8). En AES-128: *RC*[*i*] = 01, 02, 04, 08, 10, 20, 40, 80, 1B y 36 para i = 1, 2, 3, 4, 5, 6, 7, 8, 9 y 10, respectivamente.

III. Arquitectura propuesta de cifrado/descifrado AES128

La arquitectura de cifrado/descifrado AES-128 propuesta está compuesta por una unidad de control, un submódulo cifrador AES-128, un submódulo descifrador AES-128, un módulo de expansión de llave y un arreglo de multiplicadores por 2, como se muestra en la Fig. 3. En general, la unidad de control se encarga de controlar el flujo de datos de entrada y salida a la arquitectura. Esta comunicación se establece mediante el uso del protocolo USB (del inglés: Universal Serial Bus) entre la arquitectura y otros dispositivos portátiles. Además, esta unidad controla el flujo de procesamiento entre módulos ya sea para realizar el cifrado o descifrado AES-128. Específicamente, esta unidad utiliza módulos en común y un sistema de enrutamiento dinámico. Para realizar la implementación de un sistema de enrutamiento dinámico en la arquitectura propuesta, la unidad de control utiliza multiplexores para controlar la conexión y desconexión entre módulos.

El submódulo cifrador AES-128 cifra un bloque de datos a través de las transformaciones *AddRoundKey*, *SubBytes*, *ShiftRows* y *MixColumns*. El submódulo descifrador AES-128 descifra a través de las transformaciones *AddRoundKey, InvSubBytes*, *InvShiftRows*, e *InvMixColumns.* El módulo de expansión de llave recibe 16 bytes de la llave de cifrado y entrega como salida las 44 palabras requeridas en el proceso de cifrado y descifrado en AES-128. Finalmente, el arreglo de multiplicadores por 2 es utilizado para realizar las transformaciones *MixColumns* e *InvMixColumns*. A continuación se describe a detalle el funcionamiento de los siguientes submódulos:

- El **submódulo cifrador AES-128** contiene principalmente un bloque de sustitución *BS* y un par de compuertas lógicas (XOR_1 y XOR_2) (ver Fig. 3). Como se mencionó anteriormente, el algoritmo AES es iterativo. Por lo tanto, para realizar la primera ronda (ver Fig. 1), la unidad de control activa con la señal *s_c5* = 1 la entrada *T_o* del multiplexor Mux_1 para recibir el mensaje original. Posteriormente, se realiza la transformación *AddRoundKey* mediante la operación XOR entre la señal de entrada *T_o* y la llave *ll_1'*. Cabe mencionar que el bloque expansión de llave genera 10 llaves, las cuales son utilizadas durante la ejecución de las 10 rondas. Por lo tanto, la unidad de control proporciona de manera serial una llave al bloque de

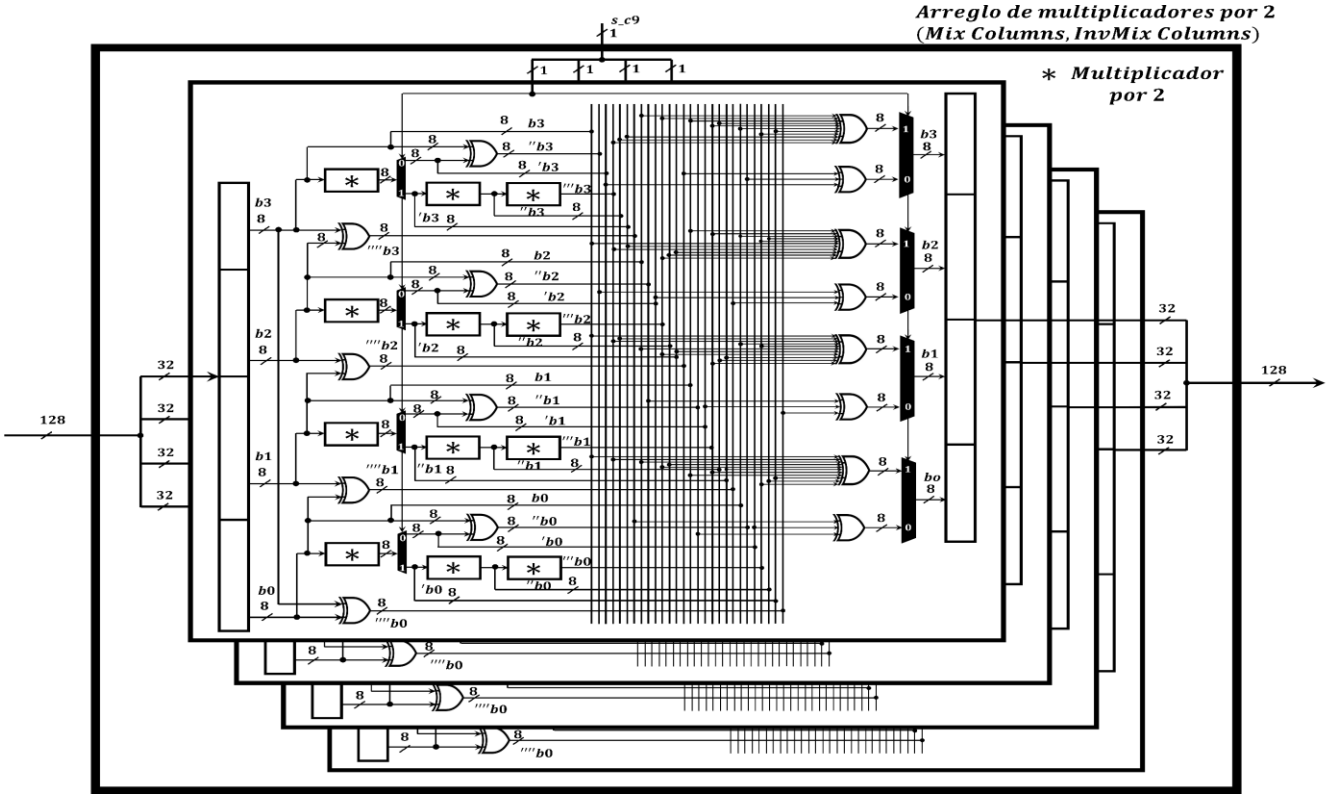

Fig. 4. Diagrama del arreglo de multiplicadores por 2.

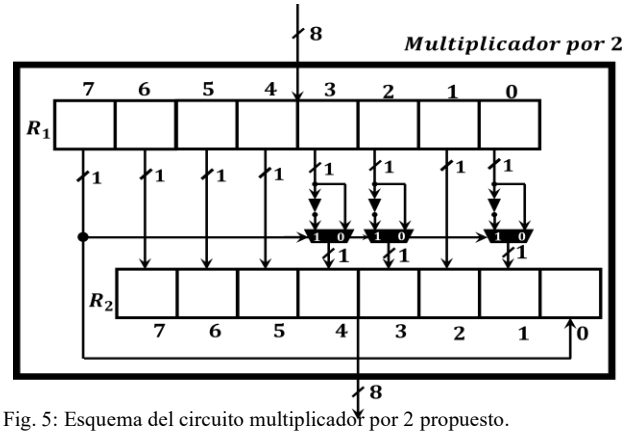

Fig. 5: Esquema del circuito multiplicador por 2 propuesto.

sustitución *BS* mediante el uso de los multiplexores Mux_5 y Mux_2 durante las primeras 9 rondas. En la décima ronda, la señal de salida del multiplexor Mux_2 junto con la señal de salida del Mux_3 se envían a la compuerta XOR_2 para obtener el mensaje cifrado. Específicamente, el bloque de sustitución *BS* realiza la transformación *SubBytes* mediante la matriz caja-S, la cual es implementada usando LUTs, y la transformación *ShiftRows* mediante el uso de registros de corrimiento.

- El **submódulo descifrador AES-128** contiene dos bloques inversos de sustitución BIS_1 *y* BIS_2 y tres compuertas lógicas (*XOR'*$_1$, *XOR'*$_2$ y *XOR'*$_3$). En este caso, inicialmente la compuerta *XOR'*$_1$ realiza la transformación *AddRoundKey* entre la señal de entrada *T_c* y la llave *ll_**10**''*. Esta llave y las llaves *ll_**9**'', ll_**8**'',..., ll_**2**''* son utilizadas durante las siguientes 8 rondas por el bloque inverso de sustitución BIS_1 y la ultima ronda la ejecuta el bloque inverso de sustitución BIS_2. Ambos bloques realizan las transformaciones *InvSubBytes* e *InvShiftRows*.

Como se puede observar en la Fig. 3, la arquitectura propuesta utiliza el arreglo de multiplicadores por 2 para realizar las transformaciones *MixColumns* e *InvMixColumns* en los procesos de cifrado y descrifrado, respectivamente. La Fig. 4 muestra el esquema general del arreglo de multiplicadores por 2. Desde el punto de vista de ingeniería, la multiplicación es una de las operaciones más demandantes en términos de velocidad de procesamiento y consumo de area. En particular, en esta aplicación, se require de un gran número de circuitos multiplicadores para realizar multiplicaciones de matrices, por lo tanto, el consumo de área es significativo.

A. Optimización del Multiplicador por 2

En este trabajo, se optimiza el multiplicador por 2 para disminuir significativamente el consumo de área del cifrador/descifrador AES. Cada multiplicador por dos realiza multiplicación en el campo finito GF(2^8). La Fig. 5 muestra el esquema del multiplicador por 2 propuesto, el cual está compuesto de dos registros de 8 bits, tres multiplexores y tres compuertas NOT. Después de analizar cuidadosamente la operación del polinomio $g(x)=x^8+ x^4+ x^3+ x+1$, nos hemos dado cuenta que el bit más significativo indica el momento en el cual se realizan las operaciones lógicas XOR. En este caso se realiza una operación lógica XOR entre el valor constante de 1 y el valor de un bit del registro R_1. Si el valor del bit del registro R_1 es igual a 1, el resultado (1 XOR 1) = 0. En caso contrario, el resultado (1 XOR bit del registro R1) = (1 XOR 0) = 1. Por lo tanto, se sustituye las operaciones XOR por compuertas lógicas NOT para negar los bits del registro R_1 y el resultado es

almacenado en el registro R_2. Una vez realizadas estas operaciones, el circuito multiplicador por 2 realiza la operación de polinomios de la siguiente manera: Primero, se guardan ocho bits en el registro R_1 de manera paralela. En este circuito utilizamos el valor del bit más significativo para controlar el flujo de datos de los multiplexores. En caso de que el valor del bit más significativo sea 1, el registro *R2* recibe el valor negado de los bits 0, 2, y 3 y el valor de los restantes bits. En caso contrario, sólo se transfieren los bits del registro R_1 (0-6) al registro R_2 (1-7). Finalmente, el bit más significativo de R_1 (7) se coloca en el bit menos significativo de R_2 (0). Cabe destacar que el diseño del multiplicador incluye el uso de registros de corrimiento para evitar la genereación de caminos críticos. Por lo tanto, la arquitectura propuesta puede ser fácilmente escalada para realizar el cómputo de longitudes 192 bits o 256 bits. Además, la eliminación de caminos críticos facilita el enrutamiento e implementación de sistemas de cifrado/descifrado AES en avanzados dispositivos FPGA.

IV. Resultados

La arquitectura de cifrado/descifrado AES-128 propuesta se implementó en un dispositivo FPGA DE0 Cyclone V 5CEBA4F23C7N [21] de bajo costo de la compañía Altera®. La implementación de esta arquitectura requiere de 38 LUTs y 269 Registros. Este consumo de área representa el 0.025%, y 0.3% del total de recursos del FPGA, respectivamente. Por otra parte, la arquitectura propuesta ofrece un rendimiento = 0.64 Gbps, teniendo en cuenta que la tarjeta DE0 Cyclone V tiene un sistema de reloj con una frecuencia de 50 MHz (20 ns por periodo de reloj) y realiza la cifrado/descifrado en 10 ciclos de reloj. Para calcular el rendimiento de la arquitectura se utiliza (3):

$$\text{Rendimiento} = 128 \text{ bit} / (n\_c \times p) \quad (3)$$

Donde: los 128 bits corresponden a la longitud de palabra de las llaves que puede procesar el algoritmo AES, el n_c es el número de ciclos de reloj que requiere el algoritmo AES para realizar el cifrado o descifrado de la información, y p es el período de reloj.

TABLA I
COMPARACIÓN ENTRE PROPUESTAS EN FPGA Y ESTE TRABAJO

	LUTS	Registros + BRAM o ROM	Rendimiento
Este trabajo	38	269 + 0	0.64 Gbps (50 MHz)
Rajasekar et al. [16]	6204	12,408 + 0	0.56 Gbps (190.658 MHz)
Visconti et al. [17]	15,029	4,296 + 0	3.132 Gbit/s (220 MHz)
Kundi *et al* [18]	556	1112 + 2	4.569 Gbps (357 MHz)
Srinivas *et al* [19]	26013	9848 + 8	1.28 Gbps (208.073 MHz)

Para realizar una comparación coherente y concisa entre nuestra propuesta y los trabajos existentes, sólo se tomaron en cuenta los trabajos que proponen un cifrador/descifrador AES. Como se puede observar en la Tabla I, nuestra arquitectura requiere aproximadamente 163, 395, 14, 867 veces menos LUTs y 46, 16, 4, 37 veces menos registros en comparación con los trabajos existentes [16]- [19], respectivamente. Por otra parte, nuestra arquitectura muestra un menor rendimiento en comparación con los trabajos existentes. Sin embargo, estos trabajos utilizan un sistema de reloj de mayor frecuencia. Evidentemente, si se utilizan las frecuencias: 190.658 MHz [16], 220 MHz [17], 357 MHz [18] y 208.073 MHz [19], respectivamente, se obtendría de manera teórica 2.4 Gbs, 2.8 Gbs, 4.5 Gbps y 2.6 Gbps. Estos rendimientos teóricos se obtienen al sustituir los siguientes valores en la ecuación 3 de la siguiente manera:

$\text{Rendimiento}_{190.658\ \text{MHz}}$ [16] = 128 bit / (10 x 5.2 ns) = 2.4 Gbs
$\text{Rendimiento}_{220\ \text{MHz}}$ [17] = 128 bit / (10 x 4.54 ns) = 2.8 Gbs
$\text{Rendimiento}_{357\ \text{MHz}}$ [18] = 128 bit / (10 x 2.8 ns) = 4.5 Gbs
$\text{Rendimiento}_{208.073\ \text{MHz}}$ [19]= 128 bit / (10 x 4.8 ns) = 2.6 Gbs

Por lo tanto, se demuestra de manera teórica que el rendimiento de la arquitectura propuesta es similar y en algunos casos es mayor en comparación con las propuestas existentes [16]-[19], utilizando sus sistemas de reloj. Por otra parte, en caso de que las arquitecturas existentes utilizaran una frecuencia de reloj de 50 MHz, el rendimiento obtenido seria de 0.64 Gbs.

Rendimiento [16] = 128 bit / (10 x 20 ns) = 0.64 Gbs
Rendimiento [17] = 128 bit / (10 x 20 ns) = 0.64 Gbs
Rendimiento [18] = 128 bit / (10 x 20 ns) = 0.64 Gbs
Rendimiento [19] = 128 bit / (10 x 20 ns) = 0.64 Gbs

Además, en este trabajo se realizó la implementación del algoritmo AES en una CPU y en una GPU para comparar el rendimiento de nuestra propuesta con respecto a los sistemas digitales actuales. En particular, el algoritmo AES-128 se programó en C y se simuló en un servidor con un procesador Intel® Xeon E5-2630 de 6 núcleos, a 2.6 GHz y con una memoria de 64 GB RAM. De igual forma, el algoritmo AES-128 fue programado en C y simulado en una tarjeta NVIDIA® GeForce GTX 1080 2560 CUDA, a 1.733 GHz con 8GB GDDR5X. Además, se comparó la velocidad de procesamiento de un diseño ASIC [15] con la arquitectura propuesta. La Tabla II muestra el tiempo de procesamiento del algoritmo AES-128 en cada uno de los dispositivos. Como se puede observar, la arquitectura propuesta es 5.7 veces más rápida que el software simulado en el servidor y 3.5 veces más rápido que la simulación en la GPU. Por

TABLA II
COMPARACIÓN EN TÉRMINOS DE VELOCIDAD DE PROCESAMIENTO ENTRE UN SERVIDOR, GPU, DISEÑO ASIC Y LA ARQUITECTURA PROPUESTA

	Frecuencia de reloj	Tiempo de procesamiento
Este trabajo	50 MHz	200 ns
GPU	1.733 GHz	700 ns
Servidor	2.6 GHz	1150 ns
Ueno et al. [15]	787.40 MHz	13.97 ns

lo tanto, la arquitectura propuesta exibe un mejor rendimiento en comparación con el rendimiento de un GPU a pesar de que este último dispositivo contiene más de mil procesadores y un sistema de reloj de mayor frecuencia comparado con el reloj del FPGA. Cabe mencionar que varios factores limitan el rendimiento de la GPU. Estos factores están relacionados con el número de hilos, la transferencia de datos en memoria y el cálculo de operaciones en punto flotante. Por otra parte, la arquitectura ASIC [15] muestra un desempeño superior a la arquitectura propuesta debido a que utiliza un sistema de reloj de mucha mayor frecuencia. Sin embargo, ambas arquitecturas requieren de 10 ciclos de reloj para realizar el cifrado/descifrado.

Para demostrar el ahorro del consumo de área entre el multiplicador por 2 propuesto y el multiplicador por 2 convencional, el cual comúnmente está compuesto de compuertas lógicas XOR [16], se implementaron en un arreglo de 48 unidades. El arreglo de multiplicadores por 2 propuesto requiere de 12 LUTs y 89 registros, mientras que la implementación del arreglo de multiplicadores por 2 convecionales requiere de 2569 LUTs y 133 registros. Como se puede observar, la implementación del arreglo de multiplicadores por 2 propuesto requiere de aproximadamente 214 veces menos LUTs y 1.5 veces menos registros que la implementación del arreglo de multiplicadores por 2 convencionales.

VII. Conclusiones

En este trabajo, se presentó una arquitectura compacta de cifrado y descifrado AES-128 y su implementación en un dispositivo FPGA de bajo costo. En particular, se presenta la optimización del multiplicador por 2 en el campo finito GF(2^8) dado que esta operación es utilizada en arreglos de matrices para la multiplicación de un gran número de elementos. Cabe señalar que el diseño del multiplicador por 2 propuesto utiliza más elementos lógicos secuenciales (registros de corrimiento) que elementos combinacionales (multiplexores) para evitar la generación de caminos críticos dado que estos decrementan significativamente el rendimiento de las aquitecturas hardware y de esta manera se crean circuitos digitales inestables. Además, se propuso un sistema de enrutamiento para habilitar o deshabilitar módulos de procesamiento en común para realizar ambas operaciones (cifrado/descifrado) en el mismo dispositivo y de esta manera conseguir un bajo consumo de área. El uso de estas dos estrategias ha permitido desarrollar una arquitectura de cifrado/descifrado AES muy compacta, lo que permite su fácil integración en los sistemas embebidos actuales y su potencial uso en aplicaciones IoT dado que hoy en día, existe una mayor demanda de diseños digitales óptimos en términos de consumo de área y potencia para integrar estos diseños en dispositivos pórtatiles, en los cuales el área está limitada.

Referencias

[1] M. A. Jan, F. Khan, M. Alam, y M. Usman, “A payload-based mutual authentication scheme for Internet of Things,” Future Generation Computer Systems, vol. 92, pp. 1028–1039, Mar. 2019, doi: 10.1016/j.future.2017.08.035.

[2] D. Bui, D. Puschini, S. Bacles-Min, E. Beigné y X. Tran, "AES Datapath Optimization Strategies for Low-Power Low-Energy Multisecurity-Level Internet-of-Things Applications," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 25, no. 12, pp. 3281-3290, Dec. 2017, doi: 10.1109/TVLSI.2017.2716386.

[3] U. Farooq y M. F. Aslam, “Comparative analysis of different AES implementation techniques for efficient resource usage and better performance of an FPGA”, Journal of King Saud University - Computer and Information Sciences, vol. 29, núm. 3, pp. 295–302, jul. 2017, doi: 10.1016/j.jksuci.2016.01.004.

[4] E. A. Hernandez Diaz, H. M. Perez Meana, y V. M. Silva Garcia, “Encryption of RGB Images by Means of a Novel Cryptosystem using Elliptic Curves and Chaos”, IEEE Latin America Transactions, vol. 18, núm. 08, pp. 1407–1415, ago. 2020, doi: 10.1109/TLA.2020.9111676.

[5] J. L. Corchuelo y S. J. Rueda, "AndroidBLP for Confidentiality Management in Android Environments," in IEEE Latin America Transactions, vol. 15, no. 3, pp. 496-502, March 2017, doi: 10.1109/TLA.2017.7867600.

[6] M. Khan y N. Munir, “A Novel Image Encryption Technique Based on Generalized Advanced Encryption Standard Based on Field of Any Characteristic”, Wireless Personal Communications, vol. 109, núm. 2, pp. 849–867, nov. 2019, doi: 10.1007/s11277-019-06594-6.

[7] A. Soltani y S. Sharifian, “An ultra-high throughput and fully pipelined implementation of AES algorithm on FPGA”, *Microprocessors and Microsystems*, vol. 39, núm. 7, pp. 480–493, oct. 2015.

[8] S. S. Priya, P. Karthigaikumar, N. M. Siva Mangai, y P. Kirti Gaurav Das, “An Efficient Hardware Architecture for High Throughput AES Encryptor Using MUX Based Sub Pipelined S-Box”, Wireless Personal Communications, vol. 94, núm. 4, pp. 2259–2273, jun. 2017, doi: 10.1007/s11277-016-3385-7.

[9] V. K. Sharma, S. Kumar, y K. K. Mahapatra, “Iterative and Fully Pipelined High Throughput Efficient Architectures of AES in FPGA and ASIC”, Journal of Circuits, Systems and Computers, vol. 25, núm. 05, p. 1650049, may 2016, doi: 10.1142/S0218126616500493.

[10] R. R. Farashahi, B. Rashidi, y S. M. Sayedi, “FPGA based fast and high-throughput 2-slow retiming 128-bit AES encryption algorithm”, *Microelectronics Journal*, vol. 45, núm. 8, pp. 1014–1025, ago. 2014.

[11] H. Lee, Y. Paik, J. Jun, Y. Han, y S. W. Kim, “High-throughput low-area design of AES using constant binary matrix-vector multiplication”, Microprocessors and Microsystems, vol. 47, pp. 360–368, nov. 2016, doi: 10.1016/j.micpro.2016.10.003.

[12] V. Nandan y R. Gowri Shankar Rao, “Minimization of digital logic gates and ultra-low power AES encryption core in 180CMOS technology”, Microprocessors and Microsystems, vol. 74, p. 103000, abr. 2020, doi: 10.1016/j.micpro.2020.103000.

[13] S. Shanthi Rekha y P. Saravanan, “Low-Cost AES-128 Implementation for Edge Devices in IoT Applications”, Journal of Circuits, Systems and Computers, vol. 28, núm. 04, p. 1950062, abr. 2019, doi: 10.1142/S0218126619500622.

[14] *Advanced Encryption Standard*, FIPS 197, National Institute of Standards and Technology, nov. 2001.

[15] R. Ueno et al., "High Throughput/Gate AES Hardware Architectures Based on Datapath Compression," in IEEE Transactions on Computers, vol. 69, no. 4, pp. 534-548, 1 April 2020, doi: 10.1109/TC.2019.2957355.

[16] P. Rajasekar y H. Mangalam, “Design and implementation of power and area optimized AES architecture on FPGA for IoT application”, Circuit World, vol. ahead-of-print, núm. ahead-of-print, jun. 2020, doi: 10.1108/CW-04-2019-0039.

[17] P. Visconti, S. Capoccia, E. Venere, R. Velázquez, y R. de Fazio, “10 Clock-Periods Pipelined Implementation of AES-128 Encryption-Decryption Algorithm up to 28 Gbit/s Real Throughput by Xilinx Zynq UltraScale + MPSoC ZCU102 Platform”, Electronics, vol. 9, núm. 10, p. 1665, oct. 2020, doi: 10.3390/electronics9101665.

[18] D.-S. Kundi, A. Aziz, y N. Ikram, “A high performance ST-Box based unified AES encryption/decryption architecture on FPGA”, *Microprocessors and Microsystems*, vol. 41, pp. 37–46, mar. 2016.

[19] N. S. S. Srinivas y Md. Akramuddin, “FPGA based hardware implementation of AES Rijndael algorithm for Encryption and Decryption”, en 2016 *International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*, Chennai, India, mar. 2016, pp. 1769–1776.

[20] J. Daemen y V. Rijmen, *Specification for the Advanced Encryption Standard (AES). Federal Information Processing Standards Publication* 197, 2001.

[21] DE0-CV User Manual, Terasic Inc., Hsinchu City, Taiwan, 2016.

Christian Equihua received the BS degree in Electronics and Telecommunications at "Universidad Politecnica de Texcoco (UPTEX)" in December 2006, currently he is studying the M.Sc. in Microelectronics at ESIME campus Culhuacan from the "Instituto Politecnico Nacional" in Mexico City, Mexico. His current interest is oriented towards information security, specifically crypto-hardware.

Esteban Anides received the BS degree at Universidad Politecnica de Texcoco, Mexico, in 2018. Currently, he is a Master student at the Instituto Politecnico Nacional, Mexico. His research interest are: signal processing, neural networks, adaptive systems, digital filtering, and digital design.

Jorge Luis García received the BS degree at Instituto Politécnico Nacional, Mexico, in 2018. Currently, he is a Master student at the Instituto Politecnico Nacional, Mexico. His research interest is linked to robotics and neuromorphic circuits.

Eduardo Vázquez studied a Bachelor of Computer Science at UAM - Iztapalapa. He obtained his master's and PhD degrees at the Center for Research and Advanced Studies of the National Polytechnic Institute (CINVESTAV, Zacatenco Unit) in 2002 and 2012, respectively. He is currently professor at the Higher School of Mechanical and Electrical Engineering, Culhuacan Unit.

Gabriel Sánchez received the BS degree in Computer Science Engineering and the PhD degree in Electronic and Communications in 1999 and 2005, respectively, from the National Polytechnic Institute, Mexico City. He is a member of the National Researchers System of Mexico. His principal research interest is related to artificial neural networks.

Juan Gerardo Avalos was born in Mexico in 1984. He received the M.Sc. in microelectronics from the National Polytechnic Institute, Mexico, in 2010 and the Ph.D. degree in electronics and communications engineering from the National Polytechnic Institute, Mexico, in 2014. From 2011 to 2012 he was visiting researcher at the Vienna University of Technology, Austria. He is currently working as a Professor in the department of computer engineering, at the National Polytechnic Institute, Mexico.

Giovanny Sánchez received the M.S. degree at Instituto Politecnico Nacional, Mexico, in 2008, and the Ph.D. degree at Universitat Politecnica de Catalunya, Spain, in 2014. His research is focused on developing early auditory neural processing systems, neural-based cryptosystems in neuromorphic hardware, image and audio processing. Currently, he is an Associate Professor in the Instituto Politecnico Nacional, Mexico.