Estudio, simulación e implementación de estándares de cifrado simétrico, utilizando MatLab, VHDL y FPGA

María Fernanda Granda, Leandro Reina Pérez

Ingeniería Eléctrica y Electrónica, Escuela Politécnica Nacional, Quito - Ecuador

***Abstract*- El presente documento tiene como finalidad estudiar, simular e implementar los estándares de cifrado simétrico mediante la utilización de Matlab VHDL y FPGA. El estudio inicia con una descripción de los primeros procedimientos que se utilizaron para cifrar la información, luego se revisa en forma detallada los estándares de cifrado S-DES y AES para desarrollarlos en Matlab. Finalmente se realiza la descripción del estándar AES utilizando VHDL para implementarlo en un FPGA XC5VLX110T Virtex 5 de Xilinx.**

I. INTRODUCCIÓN

La necesidad de comunicarnos de forma segura por medios digitales, demanda el desarrollo de aplicaciones computacionalmente seguras, las mismas que están basadas en estándares y algoritmos matemáticos. En este proyecto se hace referencia a los estándares de cifrado simétrico DES y AES, siendo este último uno de los estándares más seguros y uno de los más usados en una amplia gama de comunicaciones.

Adicionalmente se realizó tanto el estudio como la simulación en Matlab, de forma detallada del estándar S-DES, como del AES que es un algoritmo mucho más complejo que el S-DES y el que actualmente es utilizado en aplicaciones que requieren un alto grado de seguridad.

En este trabajo se usa el potencial de los FPGAs y el lenguaje de descripción de hardware VHDL, para la implementación a nivel de hardware del estándar AES, y del cifrado S-DES. Para lo cual se utilizó la tarjeta de entrenamiento XUPV5-LX110T, que posee un chip FPGA XC5VLX110T de la familia Virtex 5 de Xilinx y el software de desarrollo ISE Project Navigator.

II. S-DES

A. *Diseño de un Sistema de Cifrado S-DES utilizando un FPGA*

El diseño de bloques se indica en Fig. 1:

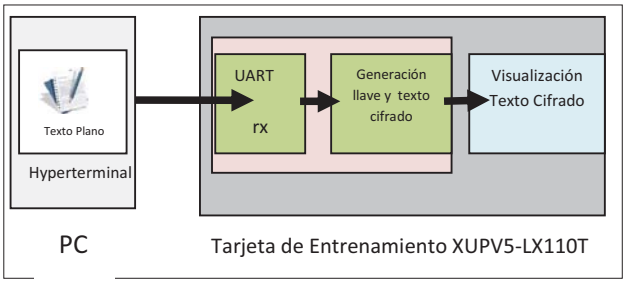

Fig. 1 Diseño de un sistema de cifrado S-DES

El bloque texto plano es el encargado de enviar los datos desde la PC hacia el FPGA, los bits transmitidos se encuentran en formato ASCII, los mismos son empaquetados en una trama RS-232, para enviar estos datos se usa el hyperterminal configurado con los siguientes parámetros: bits por segundo 19200, bits de datos 8, paridad ninguno, bits de parada 1, control de flujo ninguno.

El diagrama RTL del sistema de cifrado S-DES se muestra en Fig.2, contempla las entidades: S-DES y UART, las cuales interactúan de la siguiente manera:

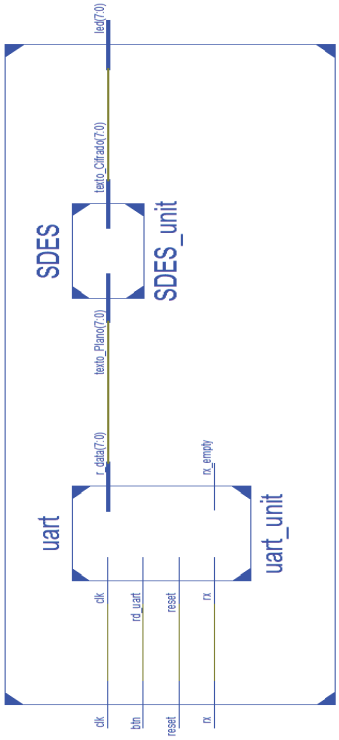

Fig. 2 Diagrama RTL del sistema S-DES

Cuando una trama RS-232 es recibida, el primer módulo en actuar es el UART, el mismo que desempaqueta la trama, tomando únicamente los bits que corresponden al carácter ASCII enviado desde la PC. Luego los ocho bits de datos pasan al módulo S-DES, el cual es el encargado de generar las dos subllaves y realizar el proceso de cifrado, una vez terminado el mismo se presenta el resultado en los leds de la tarjeta de entrenamiento.

EL módulo S-DES, se divide en dos entidades llave expansión y cifrador. En llave expansión se recibe como entrada a 10 bits, formados por la unión de los ocho bits del carácter ASCII y los dos bits menos significativos del mismo. Una vez que se forma la llave estos bits se pasan por la entidad *P10*, la misma que permuta a los bits siguiendo la regla: *K3, K5, K2, K7, K4, K10, K1, K9, K8, K6*, los bits resultantes son pasados por *Shift1*, los cuales son reordenados siguiendo la regla: *P2, P3, P4, P5, P1,P7, P8, P9, P10, P6*, luego estos bits son reordenados de acuerdo a la permutación *P8*, que sigue la regla: *s6, s3, s7, s4, s8, s5, s10, s9*, esta permutación realiza dos cosas: un cambio en el orden de los bits del resultado de *Shift1* y una reducción de los mismos, los ocho bits resultantes constituyen la subllave *K1*, luego el algoritmo indica que se debe tomar los bits resultantes del desplazamiento *Shift1* para realizar un segundo desplazamiento *Shift2*, según la regla: *s3, s4, s5, s1, s2, s8, s9, s10, s6, s7*, finalmente para obtener la subllave *K2*, se debe permutar según *P8*, los bits resultantes de *Shift2*.

El diagrama RTL del proceso de cifrado se muestra en Fig. 3, consiste en un diseño estructural el mismo que combina una serie de operaciones y permutaciones que cambian el orden de los bits originales, las operaciones son de S-DES, son descritas como entidades.

El proceso de cifrado inicia cuando llegan datos a la entidad IP, la misma que realiza un cambio de posición de los bits originales según la regla: TP_2, TP_6, TP_3, TP_1, TP_4, TP_8, TP_5, TP_7, posteriormente el resultado de IP ingresa a la función Fk, la misma que se encuentra formada por las siguientes entidades internas:

- EP
- SBox0 y SBox1
- P4
- Xor1 y Xor2

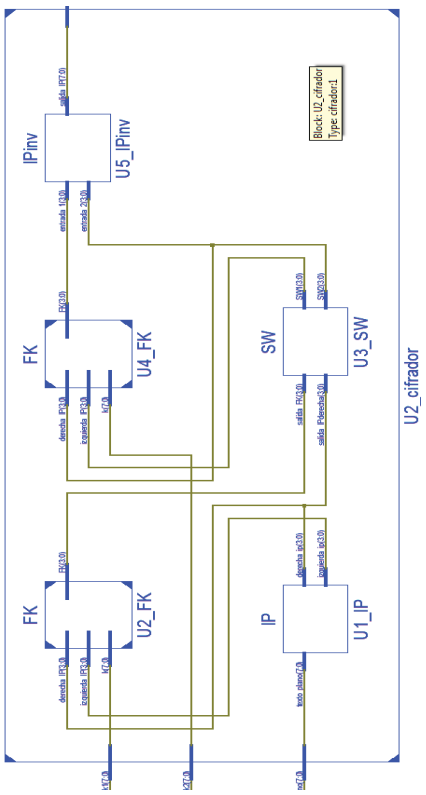

Fig. 3 Diagrama RTL del cifrado S-DES

La entidad EP realiza una expansión (4 a 8 bits) y permutación de los bits de entrada siguiendo la regla:
Pi_4, Pi_1, Pi_2, Pi_3, Pi_2, Pi_3, Pi_4, Pi_1, luego con los 8 bits resultantes se realiza una operación xor entre estos y la llave K1, este resultado se divide en dos partes iguales, los 4 bits de la izquierda ingresan a la entidad SBox0 y los 4 bits de la derecha a SBox1.

La SBox1 y la SBox0 se construyeron de acuerdo a todos los posibles valores que tomarían los bits de entrada, dado que el número de bits que ingresa es 4 se tiene 16 combinaciones posibles, cuyos resultados se muestran en la tabla 1.

TABLA I
SBOX0 Y SBOX1

In	Out
0000	01
0001	11
0010	00
0011	10
0100	11
0101	01
0110	10
0111	00
1000	00
1001	11
1010	10
1011	01
1100	01
1101	11
1110	11
1111	10

In	Out
0000	00
0001	10
0010	01
0011	00
0100	10
0101	01
0110	11
0111	11
1000	11
1001	10
1010	00
1011	01
1100	01
1101	00
1110	00
1111	11

Los bits resultantes de la SBox0 y SBox1, son permutados de acuerdo a P4 que sigue la regla: Si_2, Si_4, Si_3, Si_1, luego con este resultado se realiza la operación xor junto con los bis de la izquierda, obteniéndose el resultado de la función Fk.

Posteriormente se implementa la entidad SW, para realizar un cambio de posición entre ambas mitades del resultado de Fk, luego este resultado nuevamente se pasa por la función Fk y por último por la permutación inicial inversa, dando como resultado el texto cifrado.

III. AES

El estándar AES, se encuentra descrito en la publicación FIPS PUB 197 (Federal Information Processing Standards Publications), que es emitida por el NIST (National Institute of Standards and Technology) de los Estados Unidos de Norteamérica.

En 1997 el NIST realizó un concurso de propuestas para el desarrollo de un nuevo estándar de cifrado avanzado (AES), el mismo que debería cumplir con los siguientes criterios de evaluación:

- Seguridad.
- Eficiencia computacional.
- Requisitos de memoria.
- Idoneidad para hardware y software.

Posteriormente el NIST dio a conocer que Rijndael (desarrollado y presentado por dos criptógrafos belgas: Dr. Joan Daemen y Dr. Vincent Rijimen), sería seleccionado como el nuevo estándar del NIST (AES), por su seguridad y alto rendimiento a nivel de hardware y software.

El tamaño de un bloque en el estándar AES es de 128 bits, tanto para el proceso de cifrado como para el proceso de descifrado. El estándar también soporta bloques de longitudes de llave de 128, 192 y 256 bits.

A. Estado

Un estado es un arreglo de bytes ordenados de forma matricial. Esta distribución o arreglo se toma como base para las operaciones de cifrado y descifrado del estándar AES. Está compuesto de cuatro filas y el número de columnas se denota por Nb y es igual a la longitud del bloque dividido para 32.
Para nuestro caso de estudio la matriz de estado tiene 16 bytes ordenados como: Sf,c = S0,0, S1,0 , S2,0, S3,0, S0,1……..S3,3, esta representación se observa en Fig. 4.

$S_{0,0}$	$S_{0,1}$	$S_{0,2}$	$S_{0,3}$
$S_{1,0}$	$S_{1,1}$	$S_{1,2}$	$S_{1,3}$
$S_{2,0}$	$S_{2,1}$	$S_{2,2}$	$S_{2,3}$
$S_{3,0}$	$S_{3,1}$	$S_{3,2}$	$S_{3,3}$

Referencia [4]

Fig. 4 Matriz de estado

Recordando lo anterior, la longitud de la llave puede tener 128, 192 o 256 bits y se representa por Nk, la misma que puede tomar valores de 4, 6 o 8, estos valores reflejan el número de columnas de la llave de cifrado. También se debe considerar que el número de rondas, representado por Nr, depende del tamaño de la llave, es decir que Nr depende del valor de Nk, en la tabla 2, se hace referencia a estos valores.

TABLA II
VALORES Nk, Nr, Nb

	Nk	Nb	Nr
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

Referencia [4]

A continuación se explica el proceso de cifrado, el mismo que consta de dos operaciones: la expansión de llave y el cifrado, las mismas se muestran en Fig. 5.

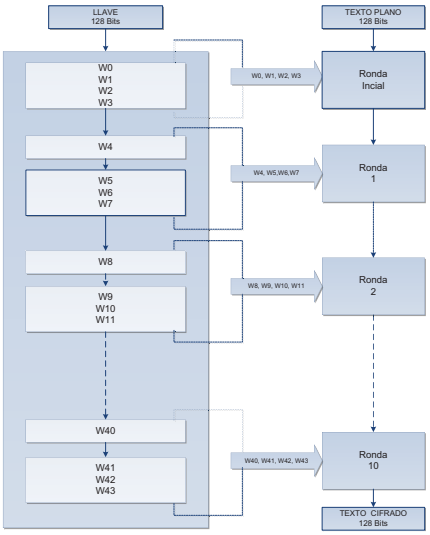

Fig. 5 Diagrama de bloques de AES

B. Expansión de llave

El proceso de expansión, toma la llave de cifrado ingresada por el usuario, realiza una rutina de expansión para generar un total de Nb(Nr + 1) subllaves o palabras, las mismas que intervienen en las rondas del proceso de cifrado del algoritmo. Para expandir la llave, se sigue dos procedimientos, uno para las llaves que son múltiplos de Nk y otro para las llaves que no son múltiplos de Nk.

La generación de cada palabra ***W***, es de la siguiente manera:

- Las palabras W[0], W[1], W[2] y W[3], se generan directamente con los valores de la llave ingresada.
- Las palabras múltiplos de Nk =4, correspondientes a W[4], W[8], W[12], W[16], W[20], W[24], W[28], W[32], W[36], W[40], se generan siguiendo una serie de operaciones, que se van a explicar más adelante.
- Las palabras que nos son múltiplos de Nk =4, W[5], W[6],W[7],W[9]……W[43], se encuentran realizando una operación xor con la palabra que le precede.

1) Palabras múltiplos de W[Nk]: Para encontrar la palabra W[Nk], así como sus múltiplos, las palabras W[nNk] (donde n va desde 1 hasta el número de rondas) pasan por las siguientes transformaciones:

- RotWord
- SubWord
- Rcon [i/Nk]
- Xor con Rcon
- W[i-Nk]
- W[i] = tem XOR w[i-Nk]

La operación RotWord, toma la palabra W[Nk - 1] = [a0, a1, a2, a3] como entrada, realiza una permutación cíclica del primer elemento de la columna y lo desplaza hacia el último elemento de la columna, dando como resultado la palabra WRotWord = [a1, a2, a3, a0]

La transformación SubWord tiene por objetivo romper la linealidad de la información de los datos originales, ya que para construir la misma se toman en cuenta criterios como la no linealidad y complejidad algebraica.

Esta transformación toma como entrada una palabra de cuatro bytes. Estos son sustituidos uno a uno con un valor de la caja-S, la misma que es presentada en la tabla 3.

TABLA III
CAJA S

x \ y	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Referencia [5]

La transformación RCON contiene los 10 valores constantes los mismos que son obtenidos de: [x^(i-1),{00},{00},{00}], cada valor es usado en las diferentes rondas del proceso de cifrado.

Luego se realiza la operación xor, entre el resultado de SubWord y el valor de Rcon [i/Nk] correspondiente al número de ronda. Luego se debe encontrar la palabra w[i-Nk], donde, i corresponde al número de palabra múltiplo de Nk que se está calculando, y el valor de Nk = 4.

Finalmente se realiza la operación W[i] =TEMP XOR W[i-Nk], que consiste en realizar la operación xor con el resultado de la transformación (Xor con Rcon) y el valor de W [i – Nk], encontrado en el paso anterior.

2) Palabras no múltiplos de Nk = 4: Se obtiene realizando un xor, entre la palabra que le precede a la que

se desea encontrar y la palabra W[i-Nk], donde i corresponde a la palabra que se va obtener y Nk = 4.

C. Cifrado

1) Ronda Inicial: Consiste en realizar la operación XOR entre el texto plano y el valor de la llave, ingresados por el usuario, ambos puestos en forma matricial.

2) Ronda Uno: Como se muestra en Fig. 9, la siguiente etapa corresponde a la Ronda 1, proceso que contiene cuatro operaciones:

- SubByte
- ShiftRows
- MixColumns
- Xor

En la operación subbyte, se realiza una sustitución de bytes se usa la caja S que se muestra en la tabla 11, con el mismo procedimiento que ya se empleó en la expansión de la llave.

La operación shiftrows consiste en aplicar el desplazamiento de filas al resultado de la matriz subbyte, como se muestra en Fig. 6.

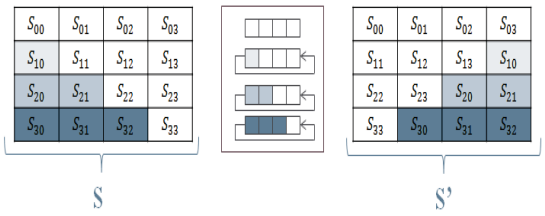

Fig. 6 Permutación Shiftrows

La operación mixcolumns consiste en multiplicar cada columna de la matriz de entrada, por una columna constante que se obtiene en GF(2^8) [X]/(X^4+1).

Para obtener la representación matricial, las columnas se consideran polinomios sobre GF(2^8), y se multiplican modulo ($x^4 + 1$), con un polinomio fijo c(x), donde:

$$c(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}$$

Finalmente se realiza la operación xor, entre el resultado de la operación mixcolumns, y las palabras correspondientes a la ronda1.

3) Rondas 2, 3, 4, 5, 6, 7, 8 y 9: Como se indica en Fig. 9, para la ronda 2, se ejecuta con el dato de entrada correspondiente al resultado de la ronda 1 y las palabras W[8], W[9], W[10], y W[11]. El dato de entrada para la ronda tres será el resultado de la ronda dos y las palabras W[12], W[13], W[14], y W[15]. Lo mismo hasta la ronda nueve.

4) Ronda Diez: Como se indica en Fig. 9, esta ronda realiza todos los pasos de las rondas anteriores, excepto la operación MixColumns. Es decir, la salida de esta ronda es calculada mediante los pasos siguientes:

- SubBytes
- ShiftRows
- Xor

A. Implementación de AES en Matlab

Para el proceso de generación de las 44 subllaves, primero se recupera el valor de la llave que es ingresada en edit text, para calcular las 44 subllaves se utiliza la función fpalabras.

1) Función fpalabras: Todos los valores de la llave son pasados a formato hexadecimal obteniéndose 32 valores en este formato, luego estos son agrupados en grupos de ocho valores que constituyen las primeras 4 subllaves (W[0], W[1], W[2] W[3]).

Para la generación de las palabras múltiplos de Nk, se aplica la función fExpansionKey, la misma que contiene en su interior funciones que realizan las operaciones de generación de las subllaves, esto se observa en Fig. 7.

```
function word = fExpansionKey(wi,Rcon,wiMenosNk)

    wRotWord = fRotWord(wi);
    wSubWord = fSubWord(wRotWord);
    wSubWordBinario = fcambiarABinario(wSubWord);
    Rcon_Binario = fcambiarABinario(Rcon);
    Xor_wSubWordBinario_Rcon_Binario =
xor(wSubWordBinario,Rcon_Binario);
    wiMenosNkBinario = fcambiarABinario(wiMenosNk);
    wordBinario =
xor(wiMenosNkBinario,Xor_wSubWordBinario_Rcon_Binario);
    word = fcambiarAHexadecimal(wordBinario);


return
```

Fig. 7 Llave múltiplo de Nk

Para la generación de las llaves que no son múltiplos de Nk, se utiliza la función FEXpansionKeySimple, la misma que realiza un xor entre la palabra W anterior con la palabra W-Nk, esta función se muestra en Fig. 8.

```
function word = fExpansionKeySimple(wi,wiMenosNk)

    wiBinario = fcambiarABinario(wi);
    wiMenosNkBinario = fcambiarABinario(wiMenosNk);
    Xor_wi_wiMenosNk = xor(wiBinario,wiMenosNkBinario);
    word = fcambiarAHexadecimal(Xor_wi_wiMenosNk);

return
```

Fig. 8 Llave no múltiplo de Nk

2) Cifrado: Para realizar el proceso de cifrado, se convierten los teres ASCII ingresados a formato binario,

luego se cuenta el número de bits, si el número de bits no es igual a 128 o no es múltiplo de 128, se procede a rellenar con bits cero.

Una vez finalizado el relleno, se procede a implementar la ronda inicial por medio de la función fRoundKey, cuyos parámetros de entrada son el texto plano en formato matricial y las primeras cuatro subllaves para realizar una operación xor entre ellas.

Una ronda se implementa mediante la función fRound, la cual lleva en su interior las funciones:

- fAfterSubBytes
- fAfterShiftRows
- fmixColumns
- fRoundKey

La función fAfterSubBytes toma 8 elementos de la matriz de entrada y los intercambia por los elementos de una matriz o caja S.

La función fAfterShiftRows realiza un desplazamiento de las filas de la matriz de entrada como se muestra en FIg. 9.

```
function AfterShiftRows =
fAfterShiftRows(matrizHexadecimal)


        filaUno = matrizHexadecimal(1,:);
        filaDos = [matrizHexadecimal(2,3:8)
matrizHexadecimal(2,1:2)];
        filaTres = [matrizHexadecimal(3,5:8)
matrizHexadecimal(3,1:4)];
        filaCuatro = [matrizHexadecimal(4,7:8)
matrizHexadecimal(4,1:6)];

        AfterShiftRows =
[filaUno;filaDos;filaTres;filaCuatro];


return
```

Fig. 9 Función AfterShiftRows

La función fmixColumns realiza una mezcla de columnas mediante la multiplicación de polinomios.

La función fRoundKey realiza la operación xor entre la matriz resultante de la operación mix columns y la matriz de las 4 subllaves correspondientes.

La función fRoundDiez, constituye la última función para obtener el texto cifrado, está compuesta de las funciones:

- fAfterSubBytes
- fAfterShiftRows
- fRoundKey

C. Implementación de AES utilizando FPGA

La arquitectura planteada para el cifrador AES es presentada en Fig. 14. Aquí se puede observar claramente los distintos módulos o componentes del sistema, y el flujo de bits entre ellos por medio de los buses de datos.

El bloque de color verde representa a la aplicación realizada en MatLab, las flechas de color negro representan a los buses que transportan datos entre los módulos y los bloques de color celeste representan componentes que se encuentran en el interior del chip FPGA Virtex 5.

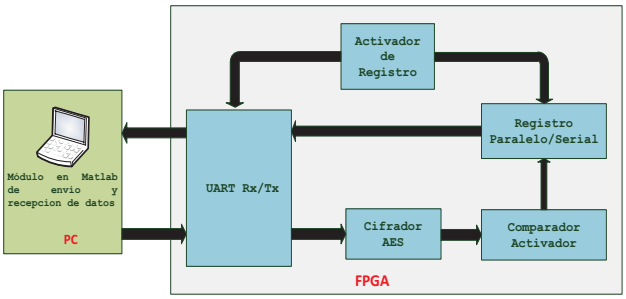

Fig. 10 Arquitectura del cifrador AES

El diagrama RTL del sistema de cifrado contiene cinco entidades y cada una de estas entidades representa un bloque en la arquitectura descrita en Fig.10. Las cinco entidades se unen mediante components en una entidad global denominada aes_total, y esta es la que se sintetiza e implementa en el kit de entrenamiento. El diagrama RTL se muestra en Fig. 11.

1) Bloque UART: Este bloque es el encargado de realizar la transmisión entre el FPGA y la PC. Cuando este bloque recibe un dato desde el PC, lo desentrama y obtiene el byte de datos, adicionalmente la UART posee las siguientes entidades:

- Generador de baudios
- UART de recepción
- Memoria FIFO de recepción de 256 bits
- Memoria FIFO de transmisión de 128 bits
- UART de transmisión

La entidad que genera los baudios permite sincronizar a la UART con la PC, ver [6].

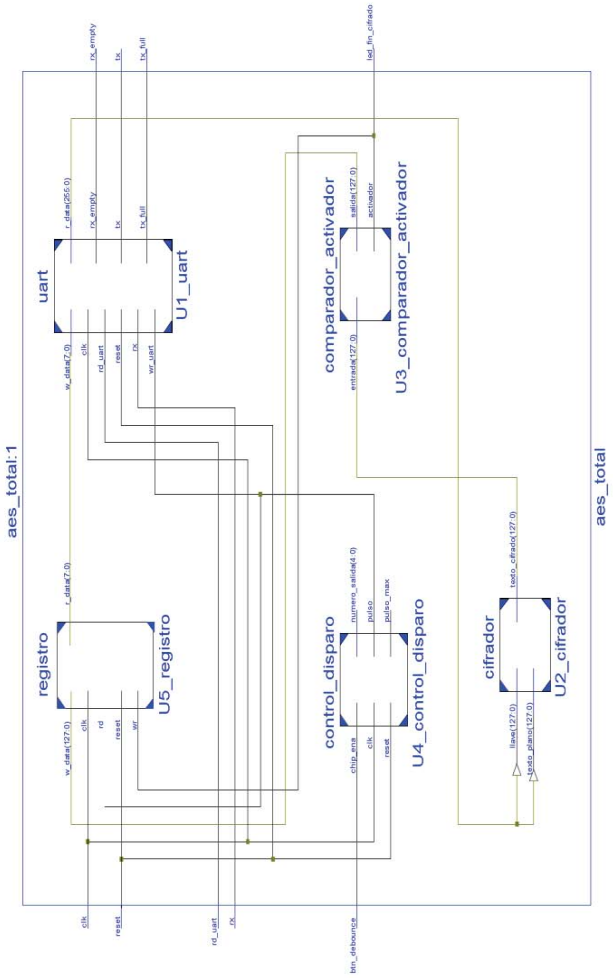

Fig. 11 Diagrama RTL del Cifrador AES

La entidad UART recepción toma la trama (en formato RS-232) enviada por la PC y quita todos aquellos bits que no corresponden a los bits de datos, esta entidad se sincroniza con una señal de control que es enviada por la entidad generador de baudios, ver [6].

Desde la PC se envía por medio de la interfaz en Matlab, una serie de 32 carácteres ASCII que corresponden a la llave y el texto plano, estos son desentramados por la entidad UART recepción y luego son almacenados en la memoria FIFO de 256 bits. La manera en que estos bits se almacenan en la FIFO es de ocho en ocho bits, una vez que la memoria se encuentre llena emite una señal de control y los 256 bits son pasados mediante un bus de datos a la entidad cifrador como se indica en Fig. 11.

La FIFO de transmisión recibe los datos de la entidad registro, su capacidad máxima de almacenamiento es de 128 bits, los mismos que son pasados de ocho en ocho a la entidad UART transmisión, ver [6].

En la UART de transmisión se ensambla la trama RS-232, la cual lleva en el campo datos los bits cifrados, estos son transmitidos a través del puertos serial hacia la PC, ver [6].

2) Bloque cifrador: Este bloque contiene todo el proceso de cifrado que contempla el estándar AES, siguiendo el esquema de Fig. 15, la entidad cifrador recibe de la entidad fifo de recepción el texto plano y la llave, una vez que se efectúan las rondas internas del algoritmo, se devuelve el texto cifrado. Los puertos de entrada y salida correspondientes a la entidad cifrador se detallan en Fig. 12.

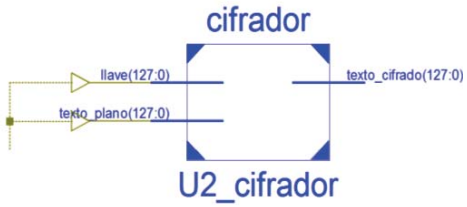

Fig. 12 Diagrama RTL del bloque cifrador

El bloque cifrador se encuentra formado por las siguientes entidades:

- Ronda inicial
- Ronda 1,2,3,…,10
- Llave expansión

En la ronda inicial simplemente se realiza un xor entre la llave y el texto plano.

La ronda 1 es generada una sola vez, pero es clonada nueve veces correspondientes a las nueve rondas del proceso de cifrado y está compuesta de las entidades:

- subByte
- shiftRows
- mix_columns
- oper_xor

El diagrama RTL de la ronda 1 se muestra en Fig. 13.

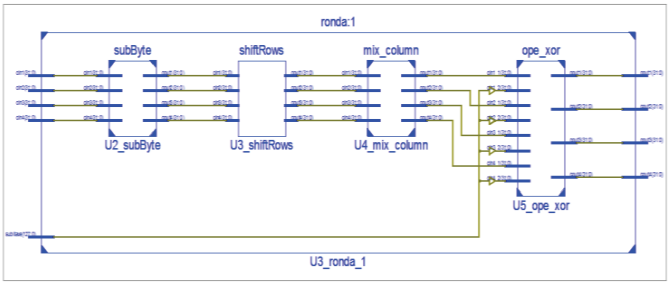

Fig. 13 Diagrama RTL de la ronda 1

La entidad subByte constituye una memoria ROM, ya que en su interior se almacena una matriz con 256 valores escritos en formato hexadecimal, que corresponden a la caja-S. La función de esta entidad es la de sustituir los valores de entrada, por los valores de la caja-S, para ello los valores de entrada sirven como direcciones de memoria para leer la tabla.

Para describir la caja-S, se declaró un tipo de dato definido por el usuario llamado TYPE_MATRIZ como un arreglo de 256 elementos y cada elemento era de tipo STD_LOGIC_VECTOR (7 DOWNTO 0), luego se declaró como valor constante SBOX cuyo tipo de dato era TYPE_MATRIZ; en la constante SBOX se almacenaron los valores de la caja-S.

La entidad shiftRows realiza un desplazamiento de las filas de la matriz de estado. Para representar este desplazamiento en este diseño se realiza el reordenamiento de los elementos de las columnas de la matriz de estado ya que en este bloque se tiene como elementos de entrada columnas.

En la entidad mix_column se debe multiplicar cada columna de la matriz de entrada por una columna constante que se obtiene en GF(2^8) [X]/(X4+1), esta operación en MatLab no resulta muy compleja ya que usa los recursos computacionales de esta herramienta en la multiplicación de matrices y su reducción en GF(2^8). Este proceso en VHDL ya no resulta tan fácil, pues realizar la multiplicación de polinomios y la división para reducir en el campo GF(2^8) utilizaría muchos recurso del dispositivo FPGA, es por esta razón que se utiliza las propiedades de aritmética modular para encontrar la forma de multiplicar matrices. Se obtienen los siguientes resultados que se muestran en Fig. 14:

$$C_0' = (C_0 * \{02\}) \oplus (C_1 * \{03\}) \oplus (C_2) \oplus (C_3)$$
$$C_1' = (C_0) \oplus (C_1 * \{02\}) \oplus (C_2 * \{03\}) \oplus (C_3)$$
$$C_2' = (C_0) \oplus (C_1) \oplus (C_2 * \{02\}) \oplus (C_3 * \{03\})$$
$$C_3' = (C_0 * \{03\}) \oplus (C_1) \oplus (C_2) \oplus (C_3 * \{02\})$$

Fig. 14 Ecuaciones resultantes

Se puede observar en la matriz de resultados que únicamente se debe determinar la multiplicación de cada uno de los valores de la columna de entrada por los valores constantes {02} y {03}; y reagrupar este resultado como se indica en la ecuación anterior.

Finalmente se realiza una operación xor entre el resultado que se obtiene de la entidad mix_column y la llave.

La ronda 10 corresponde a la última ronda en el proceso de cifrado y se encuentra formada por las entidades subByte, shiftRows y la operación xor con la llave.

La entidad llave expansión recibe los datos correspondientes a la llave ingresada por el usuario y genera las 44 palabras, que se implementan en el cifrado.

Las palabras múltiplos de cuatro se ejecutan con una entidad interna que contiene las operaciones AfterRotWord, AfterSubWord, SubBytes, XorRCON, tempXor; mientras que las palabras no múltiplos de cuatro se generan con una operación xor.

La entidad LLAVE_MULTIPLO_4, contiene 4 bloques internos y la interconexión entre ellos. Cada uno de estos bloques corresponde a una operación de transformación de la información de la llave original:

- AfterRotWord
- AfterSubWord
- XorRCON
- tempXor

Esta entidad es generada una vez y clonada diez veces correspondientes a las palabras mencionadas. El diagrama RTL correspondiente a esta entidad se muestra en Fig. 15.

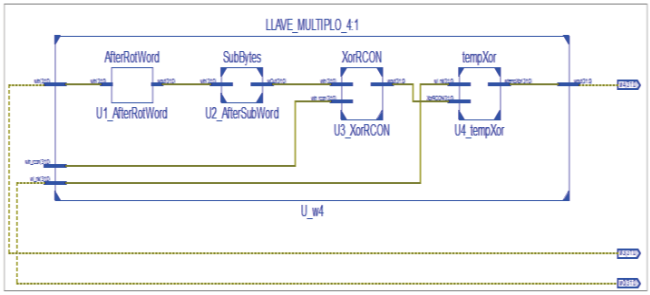

Fig. 15 Diagrama RTL interno de la entidad LLAVE_MULTIPLO_4

La entidad AfterRotWord, se describe como un circuito de desplazamiento, recibe como dato de entrada una palabra correspondiente a W3 de 32 bits. Estos bits en el interior se almacenan en un vector std_logic_vector (31 downto 0), y se los desplaza de acuerdo a la posición en la cual se ubiquen en el interior del vector.

Cabe recalcar que el bit más significativo se ubica en la posición 31 y el menos significativo en la posición cero; según la regla de desplazamiento se debe desplazar el vector en 8 posiciones hacia la izquierda.

La entidad A fterSubWord, en su interior se encuentra una tabla SBOX con 256 valores escritos en formato hexadecimal. La función de esta entidad es la de sustituir

los valores de entrada provenientes del bloque AfterRotWord por los valores de la tabla SBOX, para ello los valores de entrada sirven como direcciones de memoria para leer la tabla.

La entidad XorRCON, la funcionalidad de este bloque es simple, consiste en realizar una operación XOR entre la palabra W que sale del bloque AfterSubWord y el vector RCON.

La entidad tempXor, de manera similar que el anterior bloque, consiste en realizar una operación XOR entre la salida del bloque XorRCON y la palabra Wi_Nk.

En Fig. 16, se presenta la simulación de la entidad cifrador.

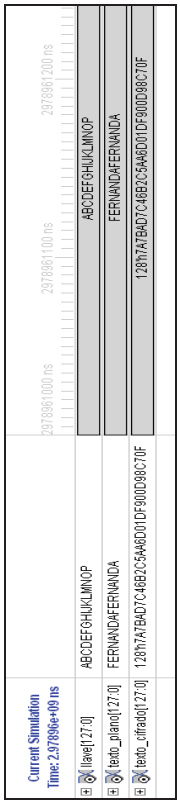

Fig. 16 Ejemplo de figura con buena resolución

3) Bloque comparador activador: El propósito de este bloque es el de comparar el valor de salida de la entidad cifrador; si dicho resultado corresponde a cifrar un texto plano donde todos sus elementos son cero, el pin de salida activador se pone en cero, caso contrario el pin activador se pone en uno y el valor de los pines de entrada se colocan en los pines de salida.

4) Bloque paralelo serial: El propósito de este bloque es el de recibir 128 bits de entrada, almacenarlos en un registro interno para luego devolverlos de 8 en 8 bits. Este bloque está formado principalmente por dos partes: un registro y el control de lectura de ese registro. El registro es un arreglo de 16 palabras, en donde cada palabra es un byte es decir 8 bits, por lo tanto este registro permite almacenar hasta 128 bits, este registro se llena con los 128 bits de entrada cuando el pin de entrada wr se encuentra en 1, el proceso de llenado se realiza de la siguiente manera, los 8 bits más significativos se ubican en el registro (0), los siguientes 8 bits en el registro (1) y así sucesivamente.

El proceso de control de lectura utiliza un contador y este proporciona las direcciones de memoria para leer el registro. Para realizar el contador se utiliza la técnica de transferencia de registros.

5) Bloque control de disparo: Este circuito consiste en un contador módulo 16, es decir envía 16 pulsos a la entidad UART y de esta manera se activa el control de envío de los datos cifrados.

IV. PRUEBAS Y RESULTADOS

Una vez que se ha expuesto el cifrado S-DES, los estándares DES y AES en MatLab como en VHDL; se procede a realizar la verificación de los resultados obtenidos a nivel de software y hardware.

A. S-DES en VHDL

Se ingresan los siguientes datos:
Texto plano →A →01000001
Llave generada en el dispositivo FPGA:
Llave →0100000101
Texto Cifrado →1 1 0 1 0 0 0 1

B. AES en Matlab:

El usuario deberá ingresar un texto plano y la llave en formato ASCII. El proceso de cifrado se ejecutará con el botón Cifrar y es necesario realizar la confirmación de la llave denominada contraseña. En Fig. 17 se muestra este procedimiento.

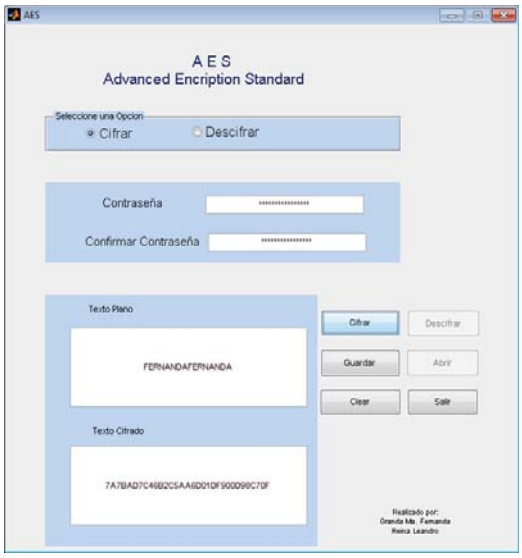

Fig. 17 Ejemplo de cifrado AES en Matlab

D. AES en VHDL

El texto plano y la llave se ingresan en formato ASCII; y para la comprobación del resultado el texto cifrado se muestra en formato hexadecimal, para enviar los datos hacia el FPGA, el usuario deberá seleccionar el puerto serial de la interfaz en Matlab y para enviar los datos deberá pulsar el botón enviar, si desea recibir el texto cifrado deberá pulsar el botón recibir. En Fig. 18 se muestra este procedimiento.

El proceso de verificación de resultados se realizará con la librería cripto de OPENSSL, para lo cual se siguen los siguientes pasos:

1. Se debe crear un archivo .txt, con el texto plano que se desee cifrar.

El nombre del archivo es textoplano.txt y el texto plano a cifrar es:
FERNANDAFERNANDA

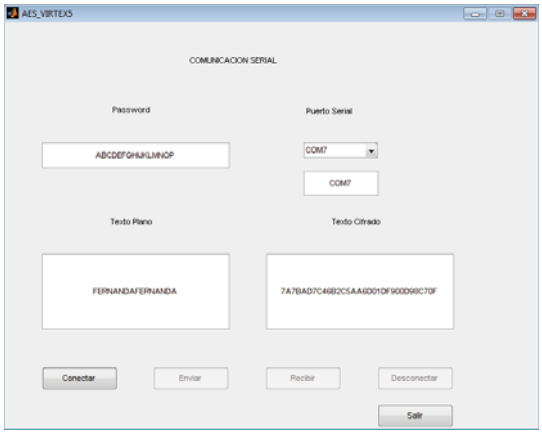

Fig. 18 Ejemplo de cifrado AES en VHDL

2. Para cifrar el archivo textoplano.txt se utiliza el comando:
openssl enc -aes-128-cbc -in textoplano.txt -out textocifrado.enc -nosalt -iv 0000000000000000 -K 41424344454647484 94A4B4C4D4E4F50

Para observar el texto cifrado en formato hexadecimal se utiliza el siguiente comando:
xxd -l 120 -c 12 textocifrado.enc

V. CONCLUSIONES

Se verifica que el estándar de cifrado AES, posee una estructura óptima y eficiente para ser implementado en hardware, ya que las operaciones que se utilizan para transformar la información no consumen muchos recursos de hardware.

El diseño planteado, es en su mayoría un diseño combinacional es decir que el valor de las salidas dependen del valor de las entradas, la ventaja de este tipo de diseño es que se obtiene mayor rapidez para realizar las operaciones de cifrado, la desventaja de este tipo de diseño es que se utilizan más recursos del FPGA. Otra alternativa de diseño sería el diseño con retroalimentación ya que este nos permite en su mayoría reutilizar el hardware, el problema de este diseño es que los resultados son más lentos ya que el valor de las salidas dependen del valor de las entradas y de un estado anterior, dado que el FPGA utilizado corresponde a la familia Virtex 5, el misma que posee más recursos de hardware se escogió el diseño combinacional.

El bloque que realiza la operación de la mezcla de columnas (mix columns), en un diseño inicial se describió como una operación de multiplicación y reducción de polinomios, dicha solución consumía demasiados recursos de hardware a tal punto que el esquema total de diseño del estándar AES no podía ser realizado, por tal motivo este bloque fue rediseñado.

En un diseño inicial para la implementación de S-DES, se intentó usar el código realizado en MatLab, asimilarlo y convertirlo en VHDL, para lo cual se usa subprogramas (funciones, subrutinas y procesos). Este método no resultó efectivo, ya que se creó código con la sintaxis correcta, pero no sintetizable, ya que lo que se intenta hacer en VHDL, no es programar en un nuevo lenguaje, sino modelar el algoritmo S-DES en hardware. Una vez comprendido el objetivo de VHDL, se usa una descripción estructural del algoritmo, creando un circuito basado en componentes más pequeños, especificando cada una de sus partes y conexiones. La unión de cada

una de estas partes denominadas entidades se realiza mediante components.

Se garantiza que el diseño realizado del estándar AES y el texto cifrado resultante realizado en el FPGA es el correcto, ya que se comprobó el funcionamiento bajo el sistema operativo Ubuntu, mediante la herramienta OPENSSL; cuyos resultados obtenidos son exactamente los mismos que los del sistema implementado físicamente. Adicionalmente se implementó un escenario de pruebas que comprueben el correcto funcionamiento del diseño, las cuales consistieron en ingresar un texto plano en la interfaz gráfica de MatLab, luego estos datos son enviados al dispositivo FPGA, el que realiza el proceso de cifrado y se obtiene el texto codificado como se esperaba.

El diseño jerárquico y por bloques, fue el más adecuado para la realización de este trabajo, ya que se puede implementar el esquema general de forma individual entidad por entidad, de esta manera se prueba el funcionamiento individual de cada una.

VI. REFERENCIAS

[1] (2011) The Vigenère Chiper. [Online]. Disponible: http://www.cs.trincoll.edu/~crypto/historical/vigenere.html

[2] W. Stallings, Fundamentos de Seguridad en Redes, Aplicaciones y Estándares. Segunda Edición, Madrid, 2004.

[3] W. Stallings, Cryptography and Network Security Principles and Practices. Segunda Edición, 1995.

[4] Advanced Encryption Standard (AES), Federal Information Processing Standards Publication 197, 2001.

[5] W. Stallings, Cryptography and Network Security Principles and Practices. Cuarta Edición, 2005.

[6] P. Chu, FPGA Prototyping by VHDL Examples. Primera Edición, 2008.

VII. BIOGRAFÍAS

María Fernanda Granda G, nació en Quito-Ecuador el 23 de Junio de 1983. Realizó sus estudios en la Unidad Educativa Experimental Manuela Cañizares, donde obtuvo el título Bachiller en Físico Matemático. Se graduó en la Escuela Politécnica Nacional como Ingeniero en Electrónica y Redes de Información en 2013.
En el año de 2012 se desempeña como Analista de Telecomunicaciones para el Centro de Soporte de Datos de Movistar. Actualmente desempeña el cargo de Analista de Telecomunicaciones para el Centro de Operaciones de Red de la Corporación Nacional de Telecomunicaciones
Áreas de interés: informática y redes de comunicaciones de datos, FPGA.
(ferchapisca12@gmail.com)

Leandro Reina P, nació en Tulcán-Ecuador el 29 de Junio de 1983. Realizó sus estudios en el Instituto Técnico Superior Bolívar, donde obtuvo el título Se graduó en la Escuela Politécnica Nacional como Ingeniero en Electrónica y Redes de Información en 2013. En el año de 2012 se desempeña como Analista de Telecomunicaciones para el Centro de Soporte de Datos de Movistar.

Actualmente desempeña el cargo de Analista de Core y Plataformas Móviles para la Gerencia de Operación y Mantenimiento de la Corporación Nacional de Telecomunicaciones.
Áreas de interés: telefonía móvil (UMTS, LTE), informática, lenguajes de programación (C#, C, bash, Python), bases de datos, FPGA y VHDL.
(leopiscofer@gmail.com)