Implementación Hardware del Estandar de Encriptación Avanzado (AES) en una FPGA

Jorge Alberto Celi Méndez, Ing. Ronald Alberto Ponguillo Intriago
Facultad de Ingeniería en Electricidad y Computación
Escuela Superior Politécnica del Litoral
Km. 30.5 Vía Perimetral, 09015863, Guayaquil, Ecuador
jceli@espol.edu.ec, rponguillo@espol.edu.ec

Resumen

Debido a la necesidad de proporcionar una gran seguridad a la información de tipo sensible o privada que se transmite por medios como el internet ya sea en teleconferencias, transacciones de comercio electrónico, transacciones bancarias en línea, etc, es necesaria la utilización de algoritmos de encriptación los cuales pueden ser implementados tanto en software o hardware, dependiendo de los sistemas en los cuales se vaya a utilizar. En este artículo se analiza la implementación en una FPGA del algoritmo AES (Advance Encryption Standar) utilizando el lenguaje de descripción de hardware VHDL, de tal manera que se obtenga un alto rendimiento del FPGA en la realización de los cálculos, finalmente se realizara la evaluación de los resultados de la simulación y del desempeño del FPGA utilizado en la implementación del algoritmo.

Palabras Claves: *AES, FPGA*

Abstract

Due to the need to provide strong security for sensitive information or private type that is transmitted by the internet media as teleconferences, e-commerce, online banking, etc., it is necessary to use encryption algorithms which may be implemented in software or hardware, depending the systems on which it will be used. This article discusses an FPGA implementation of AES (Advance Encryption Standard) using the hardware description language VHDL, so as to obtain a high performance FPGA in performing the calculations, finally out the evaluation of the simulation results and the performance of the FPGA used in the implementation of the algorithm.

Keywords: *AES, FPGA.*

1.- INTRODUCCIÓN

En la actualidad debido al rápido crecimiento de redes que proveen un gran ancho de banda para la transferencia de información a cualquier lugar del mundo, ha hecho que hoy en día gran cantidad de información que viaje a través de las redes, la cual muchas veces es información de tipo sensible que es generada por transacciones financieras, comercio electrónico, video conferencias, etc. Esto demanda una alta seguridad para el almacenamiento y transporte de esta información.

Para proteger la información el Instituto Nacional de Estándares y Tecnología (NIST) designo al algoritmo de encriptación Rijndael como el Estándar de Encriptación Avanzado (AES) en remplazo de TDES, debido a su seguridad, desempeño, eficiencia, flexibilidad y facilidad de implementación. La encriptación es usada para proteger la información mientras es trasmitida entre dos lugares o mientras esta es almacenada en algún medio vulnerable a los ataques de hackers con lo cual se garantiza que la

información llegue a su destino si ningún tipo de alteraciones y sin que esta pueda haber sido revisada por personas no autorizadas a realizarlo.

La implementación del algoritmo AES se encuentra en un IP CORE que fue desarrollado con el lenguaje de descripción de hardware VHDL.

2.- ALGORITMO AES

EL algoritmo AES/Rijndael es un algoritmo de cifrado de bloques simétrico, ya que tanto el remitente como el receptor utilizan la misma clave para encriptar y descencriptar, la cual es conocida como clave publica, este tipo de algoritmos presentan una gran velocidad de encriptación y desencriptación, no aumentan el tamaño del mensaje y es una tecnología muy conocida y difundida, Rijndael trabaja con bloques de datos de 128, 192 o 256 bits para este trabajo se utilizo un bloque de datos de 128 bits tanto para los datos como para la longitud de la clave, los datos y la clave se representan en una matriz de bytes cuyas dimensiones están dadas para la matriz de estado como 4 filas y Nb columnas. Siendo el número de columnas Nb en función del tamaño del bloque de datos.
Nb = tamaño del bloque en bits/32, la clave se presenta igualmente en una matriz similar a la de estado cuya dimensión es 4 filas y Nk columnas. Siendo el número de columnas Nk en función del tamaño de la clave.
Nk = tamaño de la clave en bits/32.
Para el proceso de encriptación se procede a someter la matriz de estado a 4 funciones matemáticas, estas transformaciones se realizan de forma reiterativa por cada ronda, donde el número de rondas denotado por Nr depende de Nb y Nk de acuerdo a la siguiente tabla.

Clave/Bloque	Nb=4 (128 bits)	Nb=6 (192 bits)	Nb=8 (256 bits)
Nk=4(128 bits)	10	12	14
Nk=6 (192 bits)	12	12	14
Nk=8 (256 bits)	14	14	14

Tabla. 1 Numero de Rondas

Gráficamente el proceso de encriptación con el algoritmo Rijndael se lo puede ver en la Figura. 1

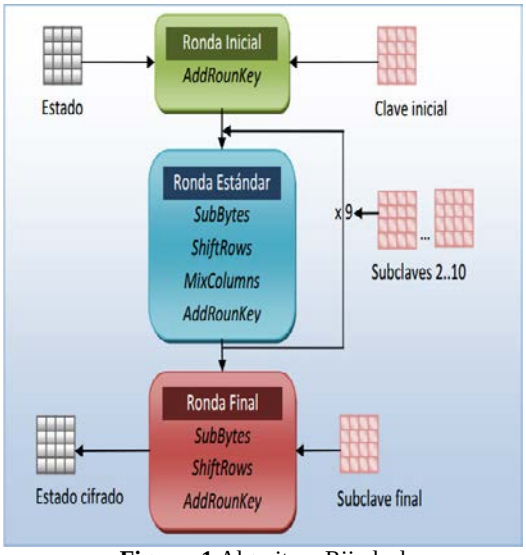

Figura. 1 Algoritmo Rijndael

Principalmente la información a encriptar es mapeada en la matriz de estado la cual se somete a la ronda inicial la cual consiste en la operación AddRoundKey entre la clave inicial y la matriz de estado, a la matriz de estado resultante se le aplican las transformaciones SubBytes, ShiftRows, MixColumns, AddRoundKey, repitiéndose este procedo Nr-1 veces, finalmente a la matriz de estado resultante de estas rondas se le aplica las funciones SubBytes, ShiftRows, AddRounkey en este orden. El resultado de esta ronda final produce el bloque de datos encriptados.

2.1.- SubBytes

En esta función se realiza una substitución no lineal a cada uno de los bytes de la matriz de estado independientemente de esta mediante el uso de una tabla de substitución(S-box) cuya formación se da por una multiplicación inversa en un Campo de Galois (2^8) que es la base matemática de este algoritmo.

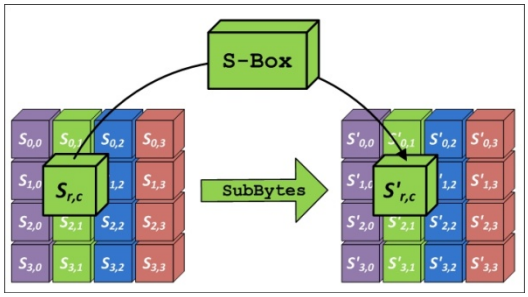

Figura. 2 SubBytes

2.2.- ShiftRows

La transformación que se realiza a la matriz de estado resultante de la función SubBytes es la de rotar hacia la izquierda las filas de esta matriz, cada fila rota un número de posiciones diferentes para la primera fila no existen rotación, para la segunda fila se rota una posición, para la tercera fila dos posiciones y para la cuarta fila tres posiciones, dependiendo del tamaño del bloque de datos el numero de rotaciones varía de acuerdo a la siguiente tabla donde C1, C2, C3 corresponden a las filas 1, 2, y 3 respectivamente.

Tamaño de bloque	C1	C2	C3
128 bits (Nb = 4)	1	2	3
128 bits (Nb = 6)	1	2	3
128 bits (Nb = 6)	1	3	4

Tabla. 2 Número de Rotaciones

El resultado final de la matriz de estado es como la de la Figura. 3

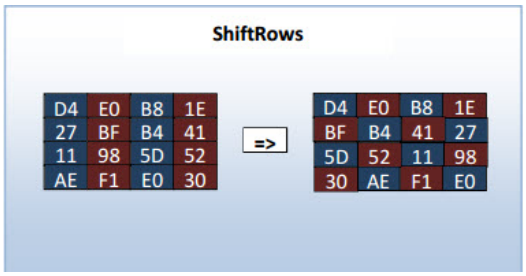

Figura. 3 ShiftRows

2.3.- MixColumns

Esta función toma la matriz resultado de la función ShiftRows y sus columnas son transformadas por medio de una multiplicación usando una matriz fija. La Operación principal de esta función es la multiplicación de las columnas de bytes modulo $x^4 + 1$ por el polinomio $c(x)$ donde matemáticamente $c(x)$ es representado por:

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

La operación estaría representado por:

$$s'(x) = c(x) \otimes s(x)$$

Donde $s'(x)$ representa la matriz de estado resultante de esta transformación.

En la Figura. 4 se muestra gráficamente la operación que se realiza en esta función.

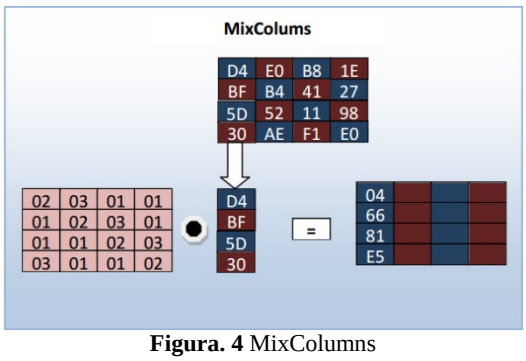

Figura. 4 MixColumns

2.4.-AddRoundkey

La transformación que se realizar en esta función consiste en tomar la matriz resultado de la función MixColumns y

realizar una operación OR-Exclusiva con una de las subclaves generadas a partir de la clave inicial en el proceso de generación de subclaves, la matriz resultado de esta función será la nueva matriz de estado para la siguiente ronda

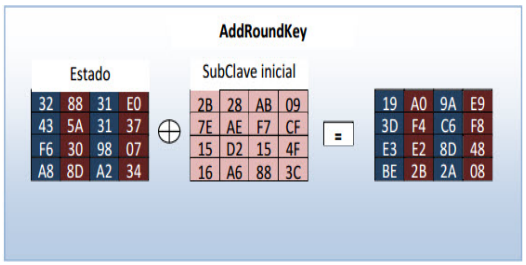

Figura. 5 AddRoundKey

2.5.- Generación de SubClaves

La generación de las subclaves parte de la clave secreta de encriptación y aplica un procesamiento sobre la misma para generar un total de (Nr +1) subclaves, una para cada ronda del algoritmo. El resultado de la expansión nos dará un arreglo lineal, denominado W, de palabras de 4 bytes y con una longitud de $Nb * (Nr + 1)$.

Las primeras palabras de este arreglo contienen la clave de cifrado, ya que estas se mapean en el arreglo W, el resto de palabras se van a generar a partir de estas primeras, como lo indica la Figura. 6

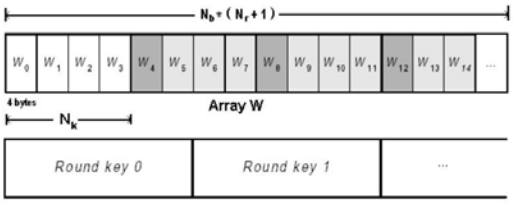

Figura. 6 Generación de Claves

En este proceso de generación de las subclaves interviene la secuencia de las siguientes funciones.

- SubWord
- RotWord
- Rcon

2.5.1.- SubWord

Esta función realiza un transformación igual a la función SubBytes la única diferencia que SubWord se realiza la transformación a 4 bytes y en la función SubBytes esta se aplica a 16 bytes.

2.5.2.- RotWord

La implementación de esta función es similar a la función ShiftRows con la única diferencia que en RotWord solo actúa sobre 4 bytes y no en 16 como la función ShiftRows.

2.5.3.- Rcon

La función Rcon utiliza un arreglo constante del cual cada vez que se va a generar una nueva subclave toma un valor de este arreglo de acuerdo a la vuelta que se está ejecutando y realiza una operación XOR con el resultado de la función RotWord.

3.- Proceso de encriptación

En este proyecto para realizar la encriptación de los datos en la FPGA se realiza el envió tanto del texto como de la clave desde una aplicación que haciendo uso del puerto serial envía estos datos a la tarjeta para su procesamiento, en la FPGA a parte del módulo de encriptación existen otros más que son de gran ayuda tanto para la recepción de los datos como para el envío de estos al computador una vez que se ha realizado la encriptación de los mismos.

Se cuenta con un modulo de reloj el cual provee de una señal o pulso antirrebote, que es activado por un switch de la tarjeta DE2 y es principalmente utilizado para envío de los datos de la tarjeta al computador. En la Figura.6 se muestra la representación de la entidad Reloj.

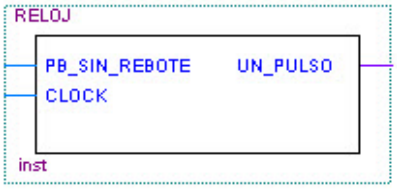

Figura. 7 Modulo Reloj

Para la recepción de los datos en la tarjeta se procedió a reutilizar un core para la lectura y escritura del puerto serial, en la lectura interviene el modulo Uart en el cual se implementa una maquina de estados que realiza un muestreo de los bits que son enviados y estos a su vez son almacenados en arreglos de 127 bits correspondientes al texto y la clave, y luego son enviado al módulo para la realización del proceso de encriptación. La imagen muestra la representación de la entidad Uart.

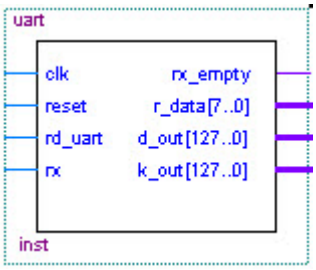

Figura. 8 Modulo Uart

Una vez enviados los datos y la clave al módulo de encriptación se realiza el proceso antes descrito. La Figura. 9 muestra la representación de la entidad aes_encript

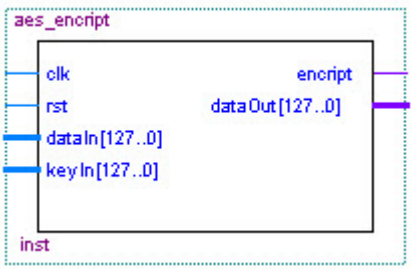

Figura. 9 Módulo Aes_Encript

El resultado de la encriptación es pasado a un módulo en el cual mediante una máquina de estado se realiza el envío de la información generada al módulo uart_send que es el encargado de escribir en el puerto, el módulo tranfer_data utiliza principalmente el pulso generado por el modulo Reloj como señal habilitadora de la máquina de estado y se hace el envío por bytes al siguiente bloque.

La Figura.10 muestra la representación de la entidad tranfer_data.

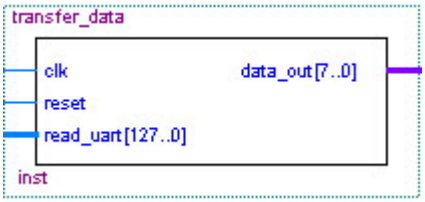

Figura. 10 Módulo Transfert_Data

El siguiente módulo es el que realiza la escritura en el puerto serial de la tarjeta el cual está siendo censando desde la aplicación que está en el computador para realizar la obtención de los datos, este módulo es la parte del core que se reutilizo en este proyecto y el cual realiza la implementación de una máquina de estado al igual que el módulo que realiza la lectura, para esta entidad la señal generada por el módulo Reloj es utilizada para habilitar la escritura de la cadena de bits que se envía desde el modulo transfer_data, la Figura. 12 representa la entidad uar_send.

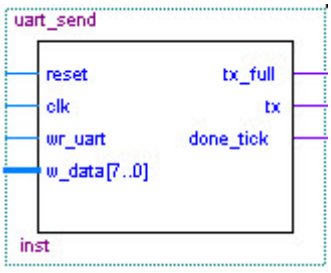

Figura. 12 Modulo Uart_Send

La representación final de todo el diseño se muestra en la Figura. 13

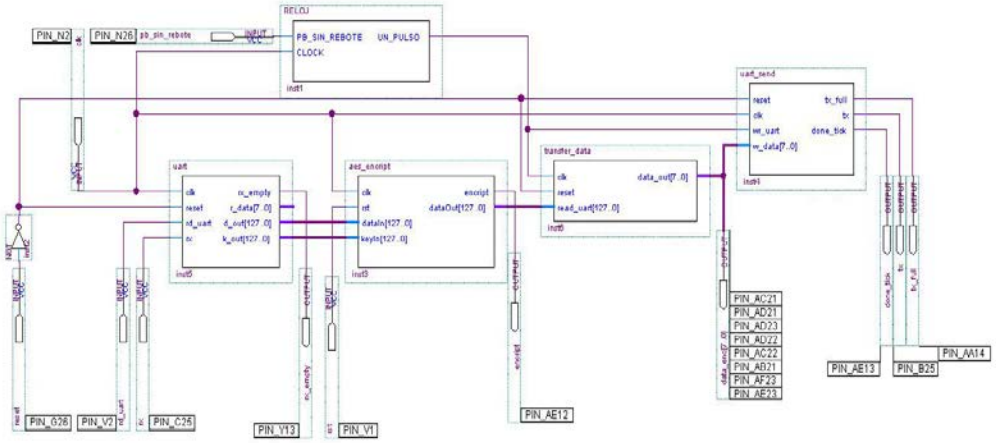

Figura. 13 Esquemático del Proyecto

4.- Simulación y Resultados

En la Figura. 14 se presenta la simulación de la transformación SubBytes en donde la cadena de 128 bits se representa como una matriz 4x4 donde cada uno de los bytes de la cadena son enviado a la función sbox que es en la cual se representa una tabla con 256 valores en los cuales se obtiene un correspondiente valor para cada uno de los bytes. En la Figura. 15 se muestra el resultado de los parámetros de rendimiento al realizar la simulación de esta entidad.

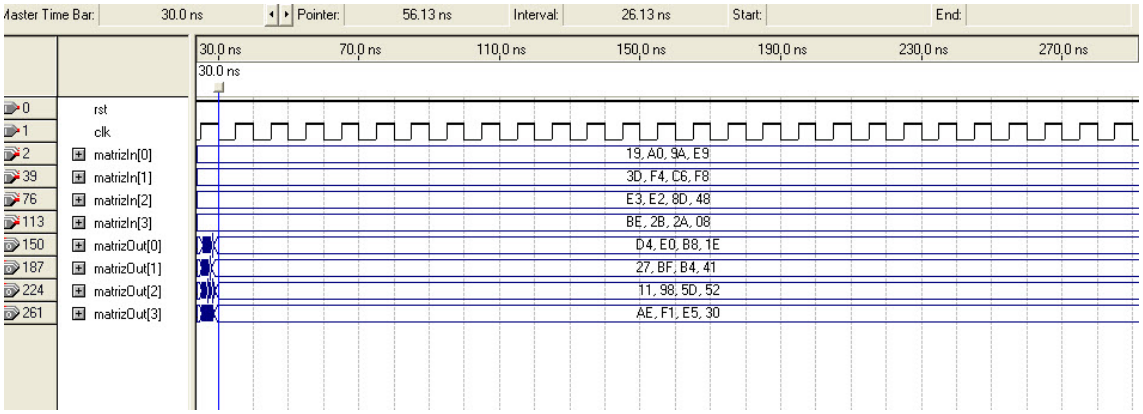

Figura. 14 Simulacion SubBytes

Flow Summary

Flow Status	Successful - Mon Oct 10 01:26:39 2011
Quartus II Version	9.1 Build 222 10/21/2009 SJ Web Edition
Revision Name	subbytes
Top-level Entity Name	subbytes
Family	Cyclone II
Device	EP2C35F672C6
Timing Models	Final
Met timing requirements	Yes
Total logic elements	257 / 33,216 (< 1 %)
Total combinational functions	128 / 33,216 (< 1 %)
Dedicated logic registers	129 / 33,216 (< 1 %)
Total registers	129
Total pins	258 / 475 (54 %)
Total virtual pins	0
Total memory bits	32,768 / 483,840 (7 %)
Embedded Multiplier 9-bit elements	0 / 70 (0 %)
Total PLLs	0 / 4 (0 %)

Figura. 15 Consumo de Recursos de SubBytes

En la Figura. 16 se muestra la simulación de la de la entidad ShiftRows la cual toma el resultado de la función SubBytes y es sometida a un proceso de rotación a la izquierda de sus tres últimas filas, a las cuales se les aplica un desplazamiento de diferentes posiciones para una de las filas. En la Figura. 17 se aprecia los recursos utilizados del FPGA por esta entidad.

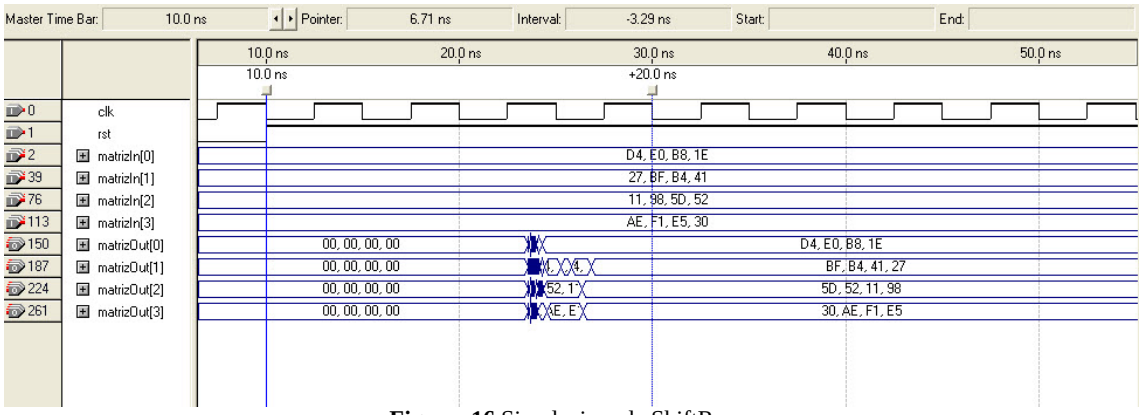

Figura. 16 Simulacion de ShiftRows

Flow Summary

Flow Status	Successful - Wed Oct 12 00:36:01 2011
Quartus II Version	9.1 Build 222 10/21/2009 SJ Web Edition
Revision Name	shiftrows
Top-level Entity Name	shiftrows
Family	Cyclone II
Device	EP2C35F672C6
Timing Models	Final
Met timing requirements	Yes
Total logic elements	256 / 33,216 (< 1 %)
Total combinational functions	0 / 33,216 (0 %)
Dedicated logic registers	256 / 33,216 (< 1 %)
Total registers	256
Total pins	258 / 475 (54 %)
Total virtual pins	0
Total memory bits	0 / 483,840 (0 %)
Embedded Multiplier 9-bit elements	0 / 70 (0 %)
Total PLLs	0 / 4 (0 %)

Figura. 17 Consumo de Recursos de ShiftRows

La simulación de la entidad MixColumns se la puede apreciar en la Figura. 18 la cual muestra el resultado obtenido de realizar la multiplicación de la matriz resultado de la entidad ShiftRows por un polinomio fijo. En la Figura. 19 se puede observar la cantidad de recursos utilizado por esta entidad.

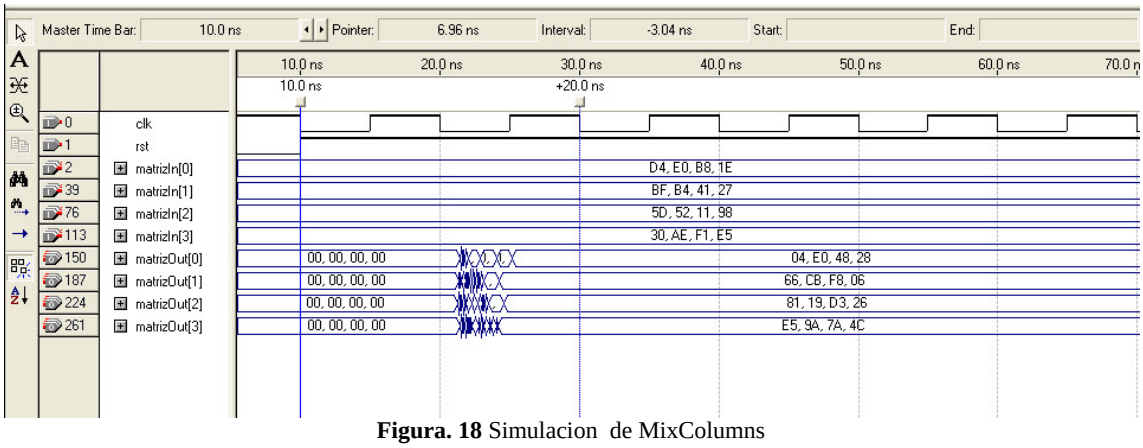

Figura. 18 Simulacion de MixColumns

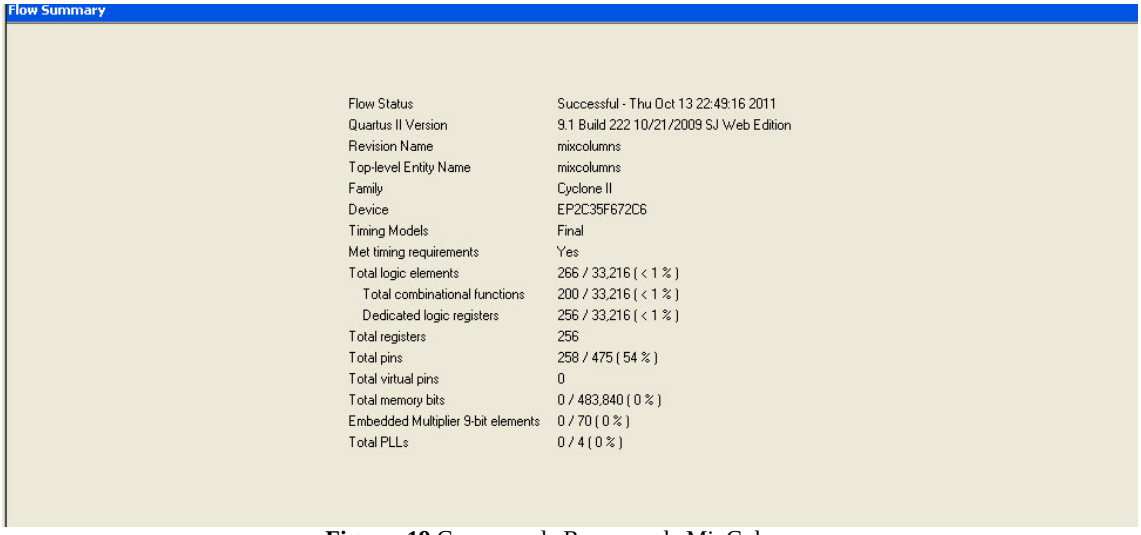

Figura. 19 Consumo de Recursos de MixColumns

En la Figura. 20 se muestra la simulación de la entidad AddRoundKey en la cual se procesa cada byte de la matriz resultado en la entidad MixColunms y se aplica una operación XOR a cada uno de los bytes con los correspondientes bytes de la subclave que es generada por la Entidad KeySchedExpansion. En la Figura. 21 los correspondientes recursos utilizado por esta entidad.

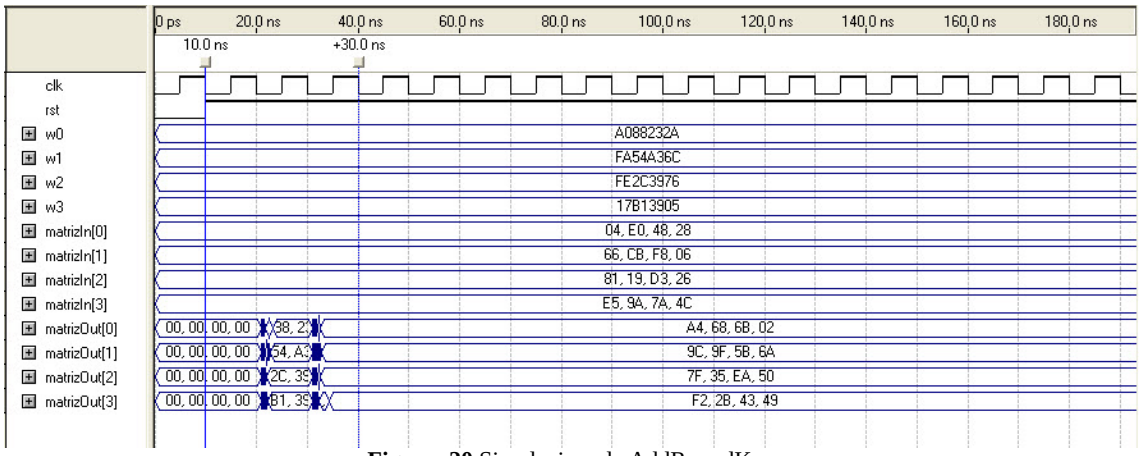

Figura. 20 Simulacion de AddRoundKey

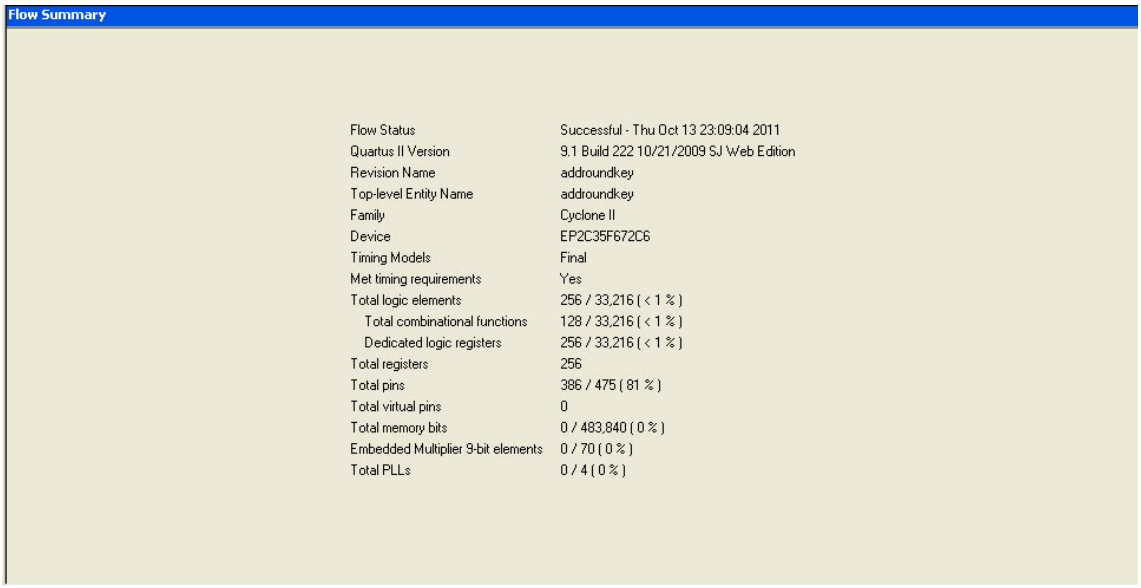

Figura. 21 Consumo de Recursos de AddRoundKey

El proceso de generación de subclaves esta implementado en la entidad KeySchedExpansion este proceso toma como semilla la clave inicial la cual corresponde a la primera subclave y a partir del último byte que forma parte de esta se proceden a generar la demás subclaves mediante el proceso anteriormente descrito. En la Figura. 22 se presenta la simulación de esta entidad y en la Figura. 23 los recursos que son utilizados por la misma.

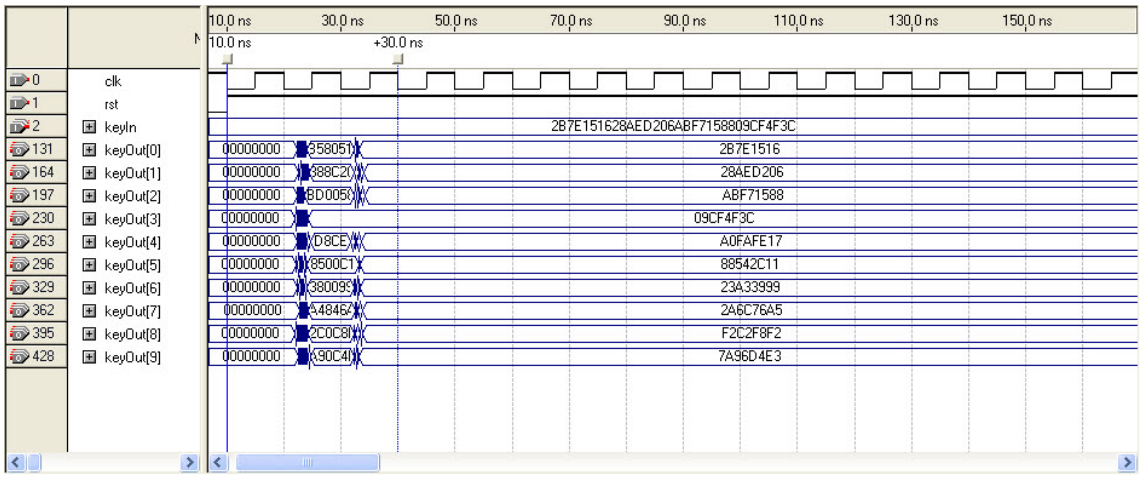

Figura. 22 Simulacion de KeySchedExpansion

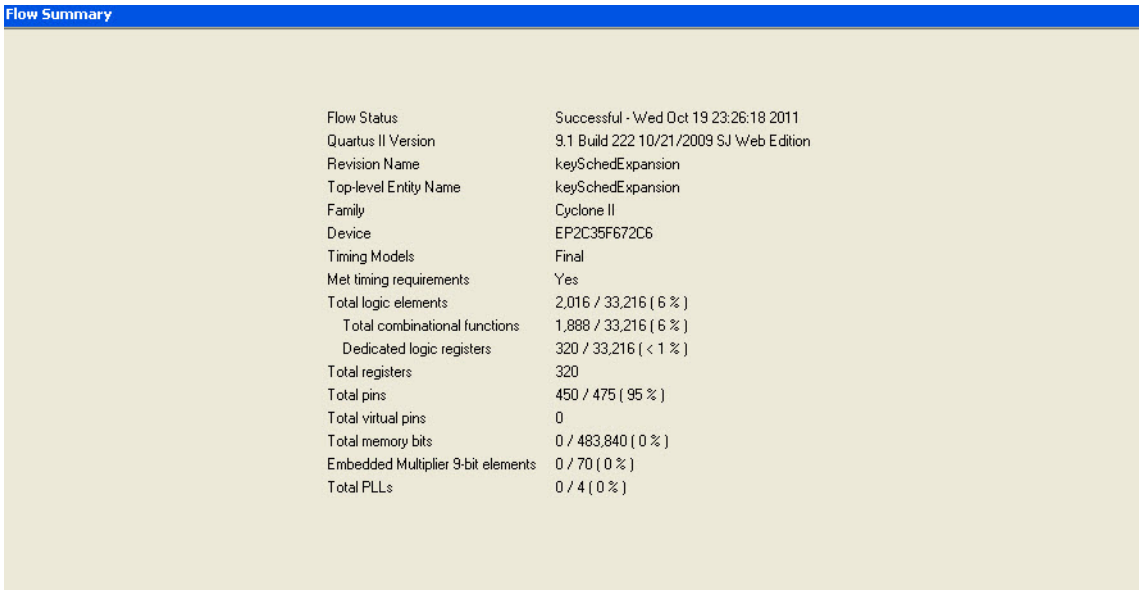

Figura. 23 Consumo de Recursos de KeySchedExpansion

5.- Conclusiones

En la implementación de cada una de una de las funciones que forman parte del algoritmo se logra que se utilicen la menor cantidad de elementos lógicos y bits de memoria de la FPGA.

Analizando el algoritmo de desencriptación AES para realizar su implementación en hardware, se observo que es necesario mucho más código y ciclos a implementar en comparación con otros sistemas de cifrado.

Se comprendió la estructura y el funcionamiento del algoritmo AES lo cual permitió que se realice una síntesis correcta y una implementación exitosa de este algoritmo usando la tecnología FGPA y el lenguaje programación VHDL.

6.- Consideraciones Futuras

Implementar la parte del algoritmo que realiza la desencriptación en hardware

Realizar la implementación del algoritmos de encriptación AES aumentando el tamaño de la clave a 192 y 256 bits

7.- Bibliografía

[1] WIKIPEDIA, AES, disponible en: http://es.wikipedia.org/wiki/Advanced_Encryption_Standard, consultado el: 18/06/2012

[2] Algoritmo de cifrado simétrico AES, disponible en: http://sedici.unlp.edu.ar/handle/10915/4210, consultado el**:** 27/07/2012

[3] Bulk Encryption on GPUs, disponible en: http://developer.amd.com/documentation/articles/pages/BulkEncryptiononGPUs.aspx, consultado el: 26/07/2012

[4] Pong P. Chu FPGA Prototyping by VHDL Examples Xilinx Spartan 3 Version, Editorial Jhon Wiley & Sons, 2008

[5] AES Algoritmo (Rijndael) Information, disponible en: http://csrc.nist.gov/archive/aes/rijndael/wsdindex.html, consultado el: 12/09/2012